MICROPERIPHERALS
iCOM
6741 VARIEL AVENUE - CANOGA PARK - CALIFORNIA 91303 - PHONE: (213) 348-1391

OPERATOR'S MANUAL

iCOM MICROPERIPHERALS[tm]

FDOS-II FOR SBC/8800/ALTAIR/IMSAI/POLY88

iCOM Microperipherals[tm]

6741 Variel Avenue

Canoga Park, CA 91303

Rev. October 15, 1976

CONTENTS

SECTION 1

UNPACKING AND INSTALLATION

1.1 UNPACKING

Remove the disk drive unit from the shipping box.

Remove the door bracing materials from the drive unit door(s).

Remove the chassis shroud by removing two screws in the upper rear and one screw on the lower rear of each side.

Remove all packing material from inside the unit.

Visually inspect for physical damage.

Insure that all connectors are firmly seated at the rear of each drive unit and on the top of the controller boards.

Replace the chassis shroud.

1.2 INSTALLATION

Insert the supplied interface board into any available slot in the microcomputer's chassis.

Insure that the board is firmly seated.

Plug the interfacing cable, from the rear of the disk drive unit, into the connector at the top of the interface board.

Plug the disk drive unit power cord into any 3-wire, grounded, 117 VAC, 50/60 HZ. outlet.

NOTE: Due to the volatility of magnetic storage media, it is advisable that a back-up copy of the supplied FDOS-II System Diskette be made as soon as possible. In the event that only a one disk drive unit system exists, copies of the files EDIT, ASMB, and EXEC should be made to some other storage media using the dumping command.

1.3 SPECIAL INSTRUCTIONS FOR INSTALLATION OF FDOS-II ON ALTAIR OR IMSAI BUS COMPATIBLE 8080 MICROCOMPUTER.

1.3.1 Initialization (Power Up)

To accomplish normal initialization of the vectors in the interface card's on-board ram, execute at address C3E7 hex. Subsequent entries to the mini-monitor may then be made at address C3E4 hex. Assumes ports 0 and 1 for console device.

1.3.2 If the user desires to operate in a different configuration, it is necessary to establish JMP instructions, in the interface card's on-board RAM, to external user supplied routines which provide input/output capability and a return to monitor vector. These entries must be made prior to booting FDOS-II.

1.3.3

FDOS-II RAM Location	User Supplied Contents	Routine Name	Description
C400 C401 C402	ØC3H CI Address (Lo Byte) CI Address (Hi Byte)	JMP CI	Subroutine to return one (1) character from console keyboard via the A-register, carry bit reset.
C403 C404 C405	ØC3H CO Address (Lo Byte) CO Address (Hi Byte	JMP CO	Subroutine which accepts a character from the C-register and outputs it to the console.
C406 C407 C408	ØC3H RI Address (Lo Byte) RI Address (Hi Byte)	JMP RI	Subroutine to return one (1) byte from the reader device via the A-register. Carry bit must be reset if a byte is returned and carry bit must be set if no input byte is available.
C409 C40A C40B	ØC3H LO Address (Lo Byte) LO Address (Hi Byte)	JMP LO	Subroutine which accepts a character from the C-register and outputs it to the list device.
C40C C40D C40E	ØC3H PO Address (Lo Byte) PO Address (Hi Byte)	JMP PO	Subroutine which accepts a byte from the C-register and outputs it to the punch device.
C40F C410 C411	ØC3H EXIT Address (Lo Byte) EXIT Address (Hi Byte)	JMP EXIT	Entry point to user's monitor program. Used by FDOS-II EXIT command and upon occurrence of fatal errors.
C412 C413 C414	ØC3H ØØ9H ØC1H	JMP RI	Disk read vector.
C415 C416 C417	ØC3H Ø94H ØC1H	JMP WRT	Disk write vector.

1.3.3 (CONT.)

FDOS-II RAM Location	User Supplied Contents	Routine Name	Description
C418	ØC3H	JMP	Asmbl/edit vector.
C419	Ø4ØH	Ø4ØH	
C41A	ØØØH		
C41B	ØC3H	JMP	
C41C	Ø4ØH	Ø4ØH	Exec vector.
C41D	ØØØH		
C41E	ØC3H	JMP	Update vector.
C41F	Ø43H	Ø43H	
C420	ØØØH		

1.3.4 The user may create a permanent copy of FDOS-II which automatically initializes these vectors, by following the instructions below.

1.3.5 Using FDOS-II, assemble the following program and place the object into a file named XX:

```
          ORG     ØC4ØØH
CI        EQU                 ; address of user CI routine
CO        EQU                 ;    "     "   "  CO    "
RI        EQU                 ;    "     "   "  RI    "
LO        EQU                 ;    "     "   "  LO    "
PO        EQU                 ;    "     "   "  PO    "
EXIT      EQU                 ; address of user MONITOR routine
JMP       CI
JMP       CO
JMP       RI
JMP       LO
JMP       PO
JMP       EXIT
JMP       ØC1ØØH
JMP       ØC194H
JMP       Ø4ØH
JMP       Ø4ØH
JMP       Ø43H
END
```

1.3.6 Type the FDOS-II command:

EDIT,EXEC,ZZ

When the prompt @ is output by the editor, enter:

AAAB15ØPAAAS:ØØØØØØØ1FF$ØL1KBE$$ ($ = Escape)

This deletes the end file from EXEC.

1.3.7 Type the FDOS-II command: MERG,YY,ZZ,XX

1.3.8 Place the above diskette into drive unit 1 and a blank diskette into drive unit Ø and type the FDOS-II command:

XGEN,YY:1

1.3.9 The new diskette now contains a copy of FDOS-II which, as it is loading, will initialize the desired RAM locations.

1.3.10 The user may control the execution address of the RUNGO command for purposes of loading and executing programs (such as BASIC or user written programs), by adding the following to the program source code:

ORG ØC418H

JMP XXXX Where XXXX = start of user program.

If re-assembly of the user programs is not practical the technique described in paragraphs 2.1 through 2.4 above may be employed.

1.4 USE OF MINI-MONITOR.

1.4.1 Command Mode

When the prompt character > appears on the console, the mini-monitor is now ready to accept the following commands:

GOTO = GXXXX(cr) Where XXXX is the desired execution address.

Example: GCØØØ(cr) will cause the FDOS loader to execute.

MEMORY DISPLAY/ALTER = MXXXX(cr)

Where XXXX = the ram location to be displayed.

1.4.2 Entry of hex data (2) hex characters for each byte) through the console keyboard will cause the contents of the current ram location to be altered and will cause the contents of the next ram location to be displayed.

1.4.3 To leave the current location un-altered and to display the next locations, depress the space bar.

1.4.4 To terminate the MEMORY DISPLAY/ALTER function, depress carriage return.

1.4.5 TEST MEMORY = TXXXX,YYYY(cr)

Where XXXX is the low ram address and YYYY is the high ram address.

Memory failure will be displayed on the console device as:

XXXX = YY XX

Where XXXX is the address of the failure, YY is the data written and ZZ is the data read.

The memory test is a continuous test which may be interrupted only by depressing CTL-C or by manipulation of the front panel controls.

NOTE: Many FDOS-II failures are attributable to improperly functioning memory. To eliminate the possibility of such a failure recommended procedure is to test all of ram using the TEST MEMORY function prior to loading FDOS-II.

1.5 LOADING FDOS

Power up and executed at address C3E7 hex or user's monitor.

Change vectors if necessary.

Insert systems diskette in drive Ø.

Type GCØØØ(cr) or execute at address CØØØ hex via front panel.

The display of ! (FDOS-II prompt character) on the console device indicates FDOS-II is awaiting command input from the keyboard.

SECTION 2

SYSTEM ORGANIZATION

2.1 FDOS-II SOFTWARE MODULES

FDOS-II consists of the following modules:

FDOS-II Resident Module
FDOS-II Executive
FDOS-II Text Editor
FDOS-II Assembler

2.1.1 FDOS-II Resident Module

The FDOS-II Resident Module is that portion of FDOS-II which is contained in the PROM memory, usually located on the supplied interface board. In addition to containing the disk input/output handler and FDOS-II bootstrap loader, this resident module is available for use by a user program to perform disk read and disk write operations.

2.1.2 FDOS-II Executive

The FDOS-II Executive is transferred from disk memory into the microcomputer's RAM memory when program control is transferred to the FDOS-II Bootstrap Loader contained in the FDOS-II Resident Module. The FDOS-II Executive is in RAM memory, and is awaiting an FDOS-II directive, when it prints the character ! on the console device. The FDOS-II Executive performs all of the command line interpretation, file management, and FDOS-II operational functions.

2.1.3 FDOS-II Text Editor

With the FDOS-II Text Editor, text input is derived from a file on disk and edited text output is stored into a file on disk. The FDOS-II Text Editor is transferred from the disk memory file named EDIT into RAM memory when the FDOS-II editor command is executed. Upon completion of edit operations, the FDOS-II Executive is reloaded into RAM memory automatically.

2.1.4 FDOS-II Assembler

With the FDOS-II Assembler, source program input is derived from a file on disk, and assembled object output is stored into a file on disk. The FDOS-II Assembler is transferred from the disk memory file named ASMB into RAM memory when the FDOS-II assemble command is executed. Upon completion of the assembly operations, the FDOS-II Executive is reloaded into RAM memory automatically.

2.2 FDOS-II DISK LAYOUT

Except for the FDOS-II Resident Module, all FDOS-II programs have been stored on the diskette enclosed with the FDOS-II Software Package. Disk storage space is divided into two or three distinct regions, depending upon whether the diskette is a System Diskette or a User Diskette.

2.2.1 System Diskette

An FDOS-II System Diskette is divided into three distinct regions: file directory area, system area, and user file area. The diskette enclosed with the FDOS-II Software Package is a preloaded FDOS-II System Diskette. On a System Diskette, track Ø is reserved for the file directory, tracks 1-3 are reserved for the storage of the FDOS-II System Executive, and the balance of the disk storage area is available as user file area. It should be noted that the FDOS-II Text Editor and Assembler should reside on a System Diskette within the user file area, as they are on the supplied FDOS-II System Diskette.

2.2.2 User Diskette

An FDOS-II User Diskette is divided into two distinct regions: file directory area, and user file area. On a user diskette, track Ø is reserved for the file directory, and the balance of the disk storage area is available as user file area. Because the User Diskette does not contain the system area, a maximum amount of storage is available for storing user files.

2.3 FDOS-II DISK FILES

2.3.1 Definition

The term "file" applies to any collection of information. Typical examples are files which contain program object information, files which contain program source information, and files which contain user generated data information.

2.3.2 Locations

All disk files are contained within the user file region of a diskette. As disk files are created, an appropriate amount of contiguous disk storage space is reserved for that file, following which begins the next file and so on. The location of a file of information on a diskette as well as other information pertinent to that file, is contained within the file directory area of that diskette. When a file is deleted from a diskette, the disk storage space previously occupied by that file in the user file area, as well as the file directory entry for that file, is made available for later use by a disk packing

technique in which FDOS-II takes all file information, which succeeds the deleted file in the user area, and packs it "down" within the user area, thus filling the storage space gap created by the deletion of a file.

2.3.3 File Names

Each file in the user area of a diskette is accessible by "filename". This filename is stored, along with other information pertinent to that file, in the file directory area of the diskette on which the file is resident. The file name is a string of ASCII characters, which may be from 1 to 5 characters in length. A filename of any number of characters may be entered, however, the FDOS-II System considers only the first five characters to be significant. Examples of valid filenames are as follows:

```
JACK
JOE3
X
#SAM
BLOB5
```

2.3.4 Device Suffixes

File names may be suffixed by a device specifier. The device specifier is a drive unit number preceeded by a colon that separates the specifier from the file name. For example:

```
JOE3:1
#SAM:3
X:Ø
JACK:2
```

A device suffix is used to reference a file which is contained on a diskette loaded into a specific disk drive unit: Ø, 1, 2, or 3. If the device suffix is omitted from the filename, the file is assumed, by FDOS-II, to reside on the diskette which is loaded into drive unit Ø, the System Diskette.

2.4 FILE DIRECTORY

2.4.1 Location of the Directory

Each diskette contains a directory of the files stored on that diskette. The directory is located on sectors 4 thru 26 of track Ø on the diskette, sectors 1 thru 3 being reserved for future FDOS-II usage. Beginning with sector 4, each sector contains 11 file control block (FCB) entries, each FCB being 11 bytes long. Thus a file directory has room to accommodate up to 253 unique files per diskette.

```
Track Ø Sector 1 - Reserved
               2 -     "
               3 -     "
               4 - File Control Block (FCB) 1 thru 11
               5 - FCB's 12 thru 22
               .
               .
               .
               .
              26 - FCB's 243 thru 253
```

2.4.2 Contents of the Directory

Information required about files on a given diskette is kept in the file directory on that diskette. The information specific to one file is contained within that file's 11 byte File Control Block (FCB). The information contained within a file's FCB includes the file name, the file attributes, the length in sectors of the file, and the file's beginning disk track and sector. The 11 byte FCB layout is as follows:

```
BYTES 1-5 ;file name padded with spaces (code 2Ø hex)
BYTE    6 ;file attributes
BYTE    7 ;file's starting track address
BYTE    8 ;file's starting sector address
BYTE 9-1Ø ;file's length in sectors, most significant byte first
BYTE   11 ;(reserved for future FCB expansion)
```

Since all file names on a given diskette are contained in a single directory, each file name must be unique. An attempt to add a file name to the directory when the same file name already exists causes an error indication.

File attributes are characteristics of files that can be set and changed by the user. Those attributes already defined within FDOS-II are as follows:

```
ØØ - user file, no restrictions.
Ø1 - permanent file, cannot be deleted.
8Ø - designates a deleted file.*
FF - designates end of directory.*
```

* These file attributes are automatically manipulated by FDOS-II and, therefore, are unavailable to the user.

A file may have a length of from one sector up to a maximum of 1,975 sectors (252,800 bytes).

2.5 SYSTEM DEVICE

The System Device is always assumed to be disk drive unit Ø. The diskette contained in disk drive unit Ø should always be a system diskette (see section 2.2.1). Disk drive units 1, 2, and 3 may contain either a system or a user diskette.

The System Device is used as the Bootstrap Device. That is to say, whenever FDOS-II attempts to bring the FDOS-II Executive, FDOS-II Text Editor, or FDOS-II Assembler from disk memory into RAM memory, it assumes that these modules are contained on the system diskette presently contained disk drive unit Ø.

The System Device is also considered to be the Default Directory Device. Thus, whenever a device suffix is omitted from an FDOS-II command or from a file name, disk drive unit Ø is assumed (see section 2.3.4).

SECTION 3

FDOS-II OPERATION

3.1 STARTING FDOS-II

To start FDOS-II, follow the microcomputer manufacturer's recommended start-up procedure for their resident debug or monitor program. Insert a system diskette into the system device and close the door (see sections 2.2.1 and 2.5). Using the microcomputer's debug or monitor commands, transfer program control to the FDOS-II Bootstrap Loader, which is the first memory location of the FDOS-II Resident Module (see appendix B).

When an exclamation mark (!) is printed on the console device, FDOS-II is awaiting command directives. An FDOS-II command directive is available which will return user control to the microcomputer's debug or monitor program (see section 5).

Thus the operator can go from the microcomputer's debug or monitor program to FDOS-II, vice versa, at will.

3.2 FDOS-II COMMAND LINE

The basic FDOS-II command line is simply an FDOS-II command directive followed by the operands, if any, required by that command directive. The command directive must be separated from the first operand by a comma (,), and each operand must be separated from another by a comma (,). For example:

```
!ASMB,AL,BOB,3↵
```

The FDOS-II command line must be terminated by a carriage return (↵). FDOS-II does not attempt to interpret nor execute any command directive until the command line is terminated.

Prior to terminating a command line, previously typed characters in the command line may be deleted from the command line by pressing the RUBOUT key on the console device. Each time the RUBOUT key is depressed, the last character existing in the command line is deleted and echoed onto the console device, as verification to the operator which character was deleted. For example:

```
          (RUBOUT key depressed)
              |   |||       |
!ASMQQB,XL,,LXAL,BOB,993↵
```

is the same as

```
!ASMB,AL,BOB,3↵
```

Prior to terminating a command line the entire command line entry may be deleted by pressing the BREAK key on the console device. FDOS-II will respond by printing !.

3.3 FDOS-II ERROR MESSAGES

When the user issues a command that contains an error, an appropriate error message will be typed out. The error messages are as follows:

FORMAT ERROR - the command line format was incorrect and command execution could not proceed.

NO SUCH FILE - a filename, from which information was to be taken, does not exist in the file directory of the specified diskette.

DUPL NAME - an attempt was made to cause an entry to be made, in the file directory, with a duplicate name to one which already exists in that directory.

NO ROOM - FDOS-II was requested to allocate more disk space to a file than was remaining on the specified diskette, or a 254'th file directory entry was attempted.

DISK NOT READY - the referenced disk drive unit is not ready. Either no diskette has been inserted, the drive unit door is not closed, or the diskette is not yet up to speed.

MEDIA ERROR - FDOS-II has been unable to write to the specified media. A copy of this media should be made to recover all but the inaccessible regions.

There are three error messages which may eminate from the FDOS-II Resident Module. For brevity sake, the error message is a single digit, or a ?, followed by a return to the microcomputer's debug or monitor program. They are as follows:

? - a checksum error was incurred while loading an object file from disk.

1 - unable to read from the diskette media

2 - an attempt was made to write more information to a file than there was disk space allocated to that file.

3 - the referenced disk drive unit has become "not-ready" (see DISK NOT READY error message).

SECTION 4

FDOS-II RESIDENT MODULE

4.1 DISK INPUT/OUTPUT

Provisions have been made in the FDOS Resident Module (see appendix B) to enable the programmer to develop user oriented programs which utilize the FD360 as a peripheral mass storage device outside of the FDOS-II environment. Contained within the module is a disk read (RI) routine and a disk write (WRT) routine which provide byte oriented input and output capabilities, respectively, to the user.

In order to use the disk input and output routines RI and WRT, it is the programmer's responsibility to first set up pointers to the area on disk which is to be accessed. This is known as "opening" a disk file. Once a disk file has been opened, RI and WRT may be called any number of times in much the same fashion as the user would call the console input and console output routines in the microcomputer's debug or monitor program. The driver handles all maintenance of the file pointers once the file has been opened. It should be noted that only one input file and one output file may be opened at any given time.

The following RAM memory locations are used by the RI and WRT routines. Refer to Appendix B for the actual memory addresses of the locations.

Location	Description
ISIZE	Input file's size in sectors (2 bytes)
ITRK	Input file's beginning track address
ISCTR	Input file's beginning unit & sector address
ICNTR	Controller's read buffer counter
OSIZE	Output file's size in sectors (2 bytes)
OTRK	Output file's beginning track address
OSCTR	Output file's beginning unit & sector address
OCNTR	Controller's write buffer counter

4.1.1 Disk Input

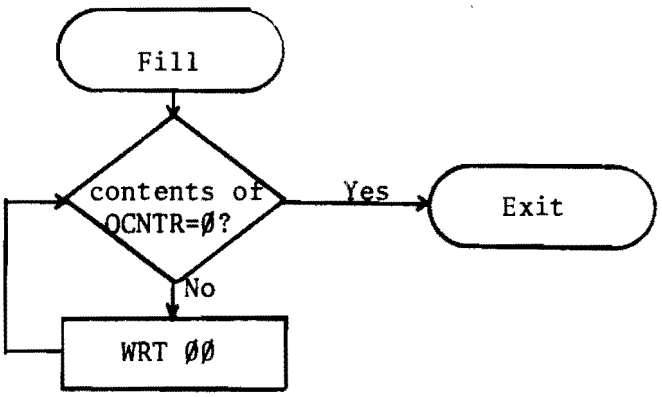
To open an input file, the user simply stores the appropriate input file information into locations ISIZE, ITRK, ISCTR, and ICNTR. Then each call to RI will return the next byte of data, from the disk, into the same register that the microcomputer's debug or monitor program would normally return a console input data byte. If no more data exists (i.e. the input file size ISIZE has reached Ø) the carry bit is returned as a "1", otherwise the carry bit is returned as a "Ø". The contents of ISIZE should be set to the number of sectors +1 that are to be read before RI is to return an end-of-file indication (carry bit set). If the programmer is going to perform his own end-of-file monitoring, the

file size may be set to some arbitrarily large number (i.e. FFFF). The contents of ITRK should be set to the track number (ØØ-4C) from which input data is to begin being read. The contents of ISCTR should be set to contain the drive unit number (ØØ-11) in bits 6 & 7, and the sector-1 (i.e. ØØ-19 hex) from which input data is to begin being read. The contents of ICNTR should be set to ØØ. Each call to RI will bring in the next sequential data byte from the disk. As a sector (128 bytes) of data is read, RI increments the disk address (ITRK and ISCTR) and decrements the input size (ISIZE). Any sector containing a DD mark is ignored, but it is computed in the input size.

4.1.2 Disk Output

To open an output file, the user simply stores the appropriate output file information into locations OSIZE, OTRK, OSCTR, and OCNTR. Then each call to WRT will output, to disk, the byte contained in the same register from which the microcomputer's debug or monitor program would normally output a console data byte. The contents of OSIZE should be set to the number of sectors that are allowed to be written before WRT terminates by printing error message 3 onto the console (see section 3.3). If the programmer is going to perform his own maximum file size monitoring, the file size may be set to some arbitrarily large number (i.e. FFFF). The contents of OTRK should be set to the track number (ØØ-4C) to which output data is to begin being written. The contents of OSCTR should be set to contain the drive unit number (ØØ-11) in bits 6 & 7, and the sector (Ø1-1A) to which output data is to begin being written. The contents of OCNTR should be set to ØØ. Each call to WRT will output one byte to disk. When 128 bytes have been sent to the disk, WRT writes that data onto the disk and increments the disk address (OTRK and OSCTR) and decrements the output file size (OSIZE). WRT verifies each sector it has written and if, after 5 attempts, it is unable to write a sector, it writes a DD mark to that sector and advances to the next contiguous disk address and attempts the disk write again. OSIZE is also decremented for each sector written with a DD mark.

When the user has written all his data to the disk, using WRT, it is possible that a partial sector of data still remains in the controller's write buffer. To insure that all data has been written onto the media, the programmer should continue to output a pad character (i.e. ØØ) until the write buffer reaches 128 bytes and WRT writes it to the disk. A flow chart of such a fill routine is as follows:



4.2 DISK SECTORING

It should be noted that RI and WRT utilize a logical/physical technique of disk addressing. Sectors on a diskette are physically adjacent and contiguous from 1-26 (Ø1-1A). After accessing physical sector 1, an entire revolution of the disk must occur if physical sector 2 cannot be accessed immediately. To overcome these rotational delays, RI and WRT translate the requested sector address (logical sector) into some other sector address (physical sector) which is then used by RI and WRT. For example, if sector 2 is requested, physical sector 1Ø (ØA) is the area on disk actually accessed; if sector 2Ø (14) is requested, physical sector 16 (1Ø) is the area on disk actually requested. This entire technique is normally transparent to the user if he remains under the control of RI and WRT. Of course, if desired, the contents of TBL may be altered, even to the point of providing a 1:1 translation of logical: physical sectoring.

SECTION 5

FDOS-II DIRECTIVES

5.1 FDOS-II COMMANDS

When an exclamation mark (!) is printed on the console device, FDOS-II is awaiting any one of the following directives. These commands are listed in alphabetical order followed by a summary list of the commands for quick reference.

Name: ASMB

Format: ASMB,sourcefilename,objectfilename,passoption)

Purpose: To assemble the contents of the source file and to direct the assembled object output, if any, to the object output file and the assembled listing, if any, to the list device or to a disk file.

Comments: All three operands must be specified. If no object or listing file is to be created, any dummy file name (i.e. X or Y or Z etc.) may be entered in this operand field since no file directory entry will be created.

The pass option operand field may contain the number 2, 3, 4, or 5.

2 = only an assembly listing is generated to the list device.

3 = only an object output is generated to the output object file.

4 = both an assembly listing and an object output are produced.

5 = only an assembly listing is generated to the disk file objectfilename.

Example: ASMB,JOES,JOEO,3)

Produce an object file named JOEO from the source file named JOES.

Name: BUILD

Format: BUILD,newfilename↵

Purpose: To construct a new source file, using the FDOS-II editor, on the diskette from the console keyboard.

Comments: The BUILD directive is functionally equivalent to the EDIT directive except that FDOS-II assumes no pre-existent input file. The operator should use the editor I (insert) command, to enter data from the console keyboard, and the editor E (end) command to terminate the operation and return to FDOS-II.

Example: BUILD,SAM↵

Produce a new file SAM from the console keyboard.

Name: CHGAT

Format: CHGAT,filename,newattributes)

Purpose: To change the present attributes of the designated file to those specified in the new attributes operand.

Comments: (see section 2.4.2)

Examples: CHGAT,MAIN,1)

Set the attributes of file MAIN to Ø1, thus setting it as a permanent, non-deletable, file.

```
CHGAT,MAIN,Ø)
CHGAT,MAIN)
```

Set the attributes of file MAIN to ØØ, thus placing no restrictions on its use or access.

Name: COPY

Format: COPY↵

Purpose: To copy the contents of the diskette in drive unit Ø onto the diskette in drive unit 1.

Comments: This is a one-for-one image copy; therefore, the contents of either diskette need not be of FDOS-II format.

If any sector of the source diskette is determined bad after 5 read tries, the last data read from that sector, whether good or bad, is written to the new diskette.

Name: CREAT

Format: CREAT,newfilename,newfilesize↵

Purpose: To create a user designated file directory entry with the specified file name and file size in sectors.

Comments: The file size is specified in hexadecimal with a minimum size of 1 sector.

The designated disk space is allocated to this file, and this new file is then treated as any other user or FDOS-II created file entry.

Example: CREAT,JACK,1F↵

Creates a new file directory entry with the file name JACK, attributes of ØØ, and an allocated disk space of 31 (1F hex) sectors.

Name: DELET

Format: DELET:unitnumber,filename1,filename2,.....,filenamen↵

Purpose: To delete the designated, non-permanent, files from the diskette, in the specified drive unit, and then to repack the contents of that diskette's user file area and file directory area, thus making the disk space available for additional files.

Comments: The file names need not be in any specific order.

The unit number refers to the drive unit in which the diskette, with the specified files to be deleted, is loaded. The unit number may be Ø, 1, 2, or 3. If the unit number is omitted, Ø is assumed.

Examples: DELET:2,JOE1,JOE7,AL,SAM,JACK↵

Deletes the specified files from the diskette loaded into drive unit 2.

DELET:Ø,JOE1,JOE7,AL,SAM,JACK↵
DELET,JOE1,JOE7,AL,SAM,JACK↵

Deletes the specified files from the diskette loaded into drive unit Ø.

Name: DUMP

Format: DUMP,filename)

Purpose: To dump the contents of the specified file to the disignated punch device.

Comments: Leader and trailer (blank) paper tape is produced when applicable.

Example: DUMP,MAIN)

Transfers the contents of file MAIN to the punch output device.

Name: EDIT

Format: EDIT,inputfilename,newoutputfilename↵

Purpose: To enable editing of the contents of the input file, using the FDOS-II Text Editor. Edited data is stored into the new output file.

Comments: Data to be edited is brought from the disk input file into the text editor's RAM buffer by using the editor's A command. Edited data is transferred from the text editor's RAM buffer to the disk output file by using the editor's P command. The edit operation is terminated, the file directory updated, and control returned to FDOS-II when the editor's E command is executed.

Example: EDIT,BOB1,BOB2↵

Establishes a new file BOB2 which will receive the data edited from the contents of the existing file BOB1.

Name: EXIT

Format: EXIT↵

Purpose: To return control back to the microcomputer's debug or monitor program.

Name: HOME

Format: HOME,unitnumber↵

Purpose: To position the disk head, on the specified drive unit, to track Ø.

Comments: The unit number may be Ø, 1, 2, or 3. If the unit number is omitted, Ø is assumed.

Examples: HOME,2↵

Returns the disk head, on drive unit 2, to track Ø.

HOME↵
HOME,↵
HOME,Ø↵

Reutrns the disk head, on drive unit Ø, to track Ø.

Name: INIT

Format: INIT,unitnumber↵

Purpose: To initialize the file directory area on the specified drive unit.

Comments: The unit number may be 1, 2, 3, or FF, where FF specified drive unit Ø.

All existing files, permanent or not, are cleared from the specified file directory.

This command must be used to prepare any non-FDOS-II diskette for use by FDOS-II.

The resultant of this command is a User Diskette (see section 2.2.2). Caution should be observed if using this command on a System Diskette.

Examples: INIT,1↵

Initializes the file area of the diskette in drive unit 1.

INIT,FF↵

Initializes the file area of the diskette in drive unit Ø.

Name: LIST

Format: LIST,unitnumber↵

Purpose: To print out the contents of the file directory on the diskette in the specified drive unit. Lists the filenames, attributes, file's starting track and sector, and the file's size in sectors.

Comments: The unit number may be Ø, 1, 2, or 3. If the unit number is omitted, Ø is assumed.

Examples: LIST,1↵

List's the file directory of the diskette in drive unit 1.

LIST↵
LIST,↵
LIST,Ø↵

List's the file directory of the diskette in drive unit Ø.

Name: LOAD

Format: LOAD,newfilename↵

Purpose: To create the specified file entry and to transfer the contents of the reader input device into that file.

Name: MERGE

Format: MERGE,newfilename,filename1,filename2,.....,filenamen↵

Purpose: To create a new file whose contents is the concatenation of the contents of the specified files, in the order in which they appear in the command.

Comments: The existing files are unaffected.

Examples: MERGE,MAIN,SUB1,SUB2,SUB3↵

Creates the new file MAIN with the contents of files SUB1, SUB2, and SUB3, in that order.

MERGE,MAINC,MAIN↵

Copies the contents of file MAIN into a new file MAINC.

Name: PRINT

Format: PRINT,filename↵

Purpose: To print the contents of the specified file to the designated list device.

Name: RENAM

Format: RENAM,oldfilename,newfilename↵

Purpose: To modify the specified file's file directory entry by replacing its existing file name with a new file name.

Comments: Only the file name area of the file's file directory entry is affected.

Example: RENAM,MAIN5,MAIN↵

Renames the file MAIN5 with the name MAIN.

Name: RUN

Format: RUN,objectfilename,offsetbias↵

Purpose: To load the contents of the object file into RAM memory for execution. The data is loaded into memory at locations which are the sum of the memory address specified in the object file plus the offset bias.

Comments: The offset bias address is specified in hexadecimal. If omitted, the offset bias is equal to Ø.

Following the loading of the object file, control will return to the microcomputer's debug or monitor program, if no auto-start address exists in the object file, or to the specified auto-start address if it exists.

Examples: RUN,MAIN↵
RUN,MAIN,↵
RUN,MAIN,Ø↵

Loads the contents of the object file MAIN into RAM memory with an offset bias of Ø.

Run,MAIN,FØØØ↵

Loads the contents of the object file MAIN into RAM memory with an offset bias of FØØØ hex.

Name: RUNGO (FDOS-II/MDS Only)

Format: RUNGO, hexobjectfilename,inputfilename,outputfilename,n↵

Purpose: To load the contents of the object file into RAM memory for execution. In addition, inputfilename and outputfilename are opened and the number n is placed in location PASS. After loading program control is transferred to memory location ASMB.

Comments: Any or all of the last three fields may be omitted. If an omitted field is followed by a supplied filed, the correct number of commas must exist in the command line.

n may be any hex number from Ø to FF.

By default, inputflie parameters are indeterminate, outputfile parameters are track=76 sector=1 size=1, and n parameter is Ø.

Examples: To update the directory following outputs to outputfilename, do a JUMP to location UPDAT in the FDOS PROM driver.

RUNGO,MAIN↵

Loads the contents of the object file MAIN into RAM memory and transfers program control to memory location ASMB.

RUNGO,ICE8Ø,LOADF,SAVEF↵

Opens the input file LOADF, creates and opens the output file SAVEF, loads the contents of file ICE8Ø into RAM memory, and transfers program control to memory location ASMB.

RUNGO,TRY,,,7↵

Sets memory location PASS to 7, then loads the contents of file TRY into RAM memory and transfers program control to memory location ASMB.

Notes: To "rewind" the input file, the user's program should perform a CALL RESTR, where RESTR is in the FDOS-II Resident (see Appendix).

Following completion of output to the output file, the user should terminate with a JMP UPDAT, where UPDAT is in the FDOS-II Resident (dee Appendix). This JMP loads the FDOS-II Exec into RAM and updates the output file's directory entry.

Name: VIEW

Format: VIEW,filename,linesperframe,firstline↵

Purpose: To display, onto the console device, the contents of the specified file one frame at a time. The number of lines per displayed frame, if not specified, is 14 by default. The first line displayed is line 1, if not specified otherwise.

Comments: "Lines per frame" and/or "first line number" may be omitted, and if so, are assumed to be 14 and 1 respectively. All numbers are in hex.

When in the VIEW command, the following four keys may be used:

N Causes the next frame to be displayed.

P Causes the previous frame to be displayed.

F Causes the first frame to be displayed (i.e. that frame whose first line is "first line").

B Causes the beginning frame to be displayed (i.e. that frame whose first line is 1).

CR (carriage return) Returns to FDOS-II executive.

Name: XGEN

Format: XGEN↵

Purpose: To generate the system region of a System Diskette (see section 2.2.1) in drive unit Ø from the copy of the FDOS-II Executive which is loaded into the reader input device.

Comments: This command is used primarily to generate new System Diskettes as new versions of the FDOS-II Executive become available or when no System Diskette exists.

If no system diskette exists, one can be generated as follows:

1. Load the copy of the FDOS-II Executive into RAM memory and execute it at memory location 2Ø hex.

2. Insert a new diskette into drive unit Ø.

3. Place a copy of the FDOS-II Executive into the reader input device and type

   XGEN↵

4. Using the LOAD command, transfer copies of the FDOS-II Text Editor and FDOS-II Assembler from the reader input device to files EDIT and ASMB respectively.

SUMMARY

OF

FDOS-II COMMANDS

ASMB,sourcefilename,destinationfilename,p

assembles the contents of the source file and directs the object to the destination file. p is the pass number which determines whether the assembly should produce a listing only, object only, or both.

BUILD,destinationfilename

enables the user to build a new source file onto the diskette from the console keyboard.

CHGAT,filename,newattributes

changes the present attributes of the designated file to those specified in the new attributes filed.

COPY

copies the contents of the diskette in drive unit "0" onto the diskette in drive unit "1".

CREAT,filename,size

creates the designated filename in the directory and allocates disk space equal to size.

DELET:u, filename1,filename2,.....,filenamen

deletes the designated files from the diskette in drive unit u, and then repacks the contents of that diskette, making the disk space available for additional files.

DUMP,filename

dumps the contents of the file to the punch output storage device.

EDIT,inputfilename,outputfilename

enables editing of the input file's contents. Edited data is stored into the output file.

EXIT

returns to the microcomputer system monitor.

HOME,u

positions the disk head on drive unit "u" to track 0.

INIT,u

initializes the file directory on the diskette in drive unit "u". Clears any existing user files on that diskette.

LIST,u

lists the contents of the file directory on the diskette in drive unit u. Lists the filenames, attributes, and file sizes in sectors.

LOAD,destinationfilename

loads the contents of the reader device into the specified file on diskette.

MERGE,newfilename,filename1,filename2,.....,filenamen

creates a new file which is a concatenation of filenames 1-n, in that order.

PRINT,filename

prints the contents of the file on the list output device.

REMAN,oldfilename,newfilename

renames the old file with the new filename.

RUN,filename,offsetbias

loads the contents of the file into RAM for execution.

RUNGO,hexobjectfilename,inputfilename,outputfilename,n

sets up the specified input, output, and n parameters, if given; loads the contents of the hex object file into RAM for execution; and then does a branch to location.

VIEW,filename,linesperframe, firstline

displays the contents of the specified file one frame at a time.

XGEN

enables system generation of other iCOM FDOS versions as might become available in the future.

APPENDIX A

FOR

FDOS-II/SBC

OPERATOR'S MANUAL

```
;
;
; ICOM, INC FD360 DIAGNOSTIC FOR GENERALIZED 8080 BASED SYSTEMS
;
; LOAD INTO RAM MEMORY AND START AT LOCATION 4000H for SBC-
  80/10 and 100H for Altair/IMSAI and 2000H for Poly 88
; LOAD A SCRATCH DISKETTE INTO THE DRIVE UNIT TO
;    BE TESTED

; TYPE THE DESIRED TEST TO BE PERFORMED

; CONTINUOUS TESTS MAY BE MANUALLY ABORTED

;  OR BY PRESSING "CTL-C"


; U=UNIT NUMBER  0(OR NOTHING), 1, 2, OR 3
; T=TRACK
; S=SECTOR

; A      -CLEAR DRIVE ELECTRONICS
; BU,T   -SEEK TO TRACK
; DU,S   -READ TO BUFFER FROM PRESENT TRACK
; FU,S   -WRITE FROM BUFFER TO PRESENT TRACK
; GU,S   -RD/WRT (BFR) CONTINUOUS ON PRESENT TRACK
; HU     -TRK0 TO TRK76 LOOP
; I      -UNIT SELECT TEST
; JU     -SEEK TEST ONCE(2 MIN)
; KU     -SEEK TEST CONTINUOUS
; LU     -SEEK TEST READ ONLY
; MU     -DD MARK TEST ONCE
; N      -RETURN TO MONITOR

; LIST OF ERRORS

; 01 - CRC ERROR ON READ 5 TIMES - 01(TRK)(UNIT/SCTR)
; 02 - CRC ERROR ON WRITE 5 TIMES - 02(TRK)(UNIT/SCTR)
; 03 - RD/WRT DATA ERROR - (REC'D)(EXP'D)(BYTE#)
; 04 - UNIT SELECT ERROR - (REC'D)(EXP'D)
; 05 - SEEK ERROR - (REC'D)(EXP'D)(TRK)(SCTR)
; 06 - DD MARK ERROR - (SCTR)
; 07 - DD MARK ERROR ON RD/WRT



; BUFFER = 1000H - 107FH

; 8080 FD360 DIAG VER. 1.0
```

```
E800            ORG     0E800H
                ;*****
                ;
                ;RESIDENT FDOSII FOR SBC-80/10 VERSION 0.1
                ;
                ;*****
                ;
                ;
0007            DATAO   EQU     7
0007            DATAI   EQU     7
0006            CNTRL   EQU     6
                ;
                ;
3CE0            BASE    EQU     3CE0H

3CE0            PASS    EQU     BASE
3CE1            OFILE   EQU     BASE+1
3CE2            OUNIT   EQU     BASE+2
3CE3            IUNIT   EQU     BASE+3
3CE4            ISIZE   EQU     BASE+4
3CE6            ITRK    EQU     BASE+6
3CE7            ISCTR   EQU     BASE+7
3CE8            ICNTR   EQU     BASE+8
3CE9            OSIZE   EQU     BASE+9
3CEB            OTRK    EQU     BASE+11
3CEC            OSCTR   EQU     BASE+12
3CED            OCNTR   EQU     BASE+13
3CDF            TITRK   EQU     BASE-1
3CEE            TISZE   EQU     BASE+14

3CDD            STACK   EQU     3CDDH
4000            ASMB    EQU     4000H
                ;
                ;
                ;
                ;
                ;
                ;ENTRY POINT WHEN Q IS TYPED
                ;LOADS FDOS AND BRANCHES TO FDOS S.A.

E800 C315E8             JMP     FDOS

E803 C3FD03     CI:     JMP     3FDH    ;KEYBOARD INPUT VECTOR

E806 C3FA03     CO:     JMP     3FAH    ;CONSOLE OUTPUT VECTOR

E809 C30004     RDRIN:  JMP     400H    ;READER INPUT VECTOR

E80C C3FA03     LO:     JMP     03FAH   ;LIST OUTPUT VECTOR
```

```
E80F C30304   PO:     JMP     0403H   ;PUNCH OUTPUT VECTOR

E812 CF       MNTR:   RST     1
E813 00               NOP
E814 00               NOP

E815 31DD3C   FDOS:   LXI     SP,STACK
E818 CD5EE8           CALL    FDOS1
E81B C30040           JMP     4000H
              ;
E81E C354E8   RSTV:   JMP     RESET
E821 C3E8E9   XUSV:   JMP     XUS
E824 C3F7E9   XXUSV:  JMP     XXUS
E827 C3FFE9   SEEKV:  JMP     SEEK+1
E82A C30BEA   RFLGV:  JMP     RFLAG
E82D C30DEA   LOOPV:  JMP     LOOP
E830 C37AE8   RSTRV:  JMP     RESTR
              ;
              ;
              ;
              ;
              ;
E833 C311E9   RIV:    JMP     RI
              ;
E836 C39CE9   WRTV:   JMP     WRT
              ;
              ;
E839 C340EA   PASSV:  JMP     IPASS   ;ASMB INTERPASS FNC

E83C CD93E8   ASSEM:  CALL    REDX
E83F CD7AE8           CALL    RESTR
E842 C30040           JMP     ASMB


              ;
E845 31DD3C   UPDAT:  LXI     SP,STACK
E848 CD5EE8           CALL    FDOS1
E84B C30340           JMP     4003H
              ;
              ;
E84E CD93E8   PROG:   CALL    REDX
E851 C312E8           JMP     MNTR
              ;
              ;

E854 3E81     RESET:  MVI     A,81H
E856 CD0DEA           CALL    LOOP
E859 3E0D             MVI     A,0DH
E85B C30DEA           JMP     LOOP
```

```
                        ;
E85E CD54E8   FDOS1:    CALL    RESET
E861 210000             LXI     H,0     ;SET BIAS=0
E864 E5                 PUSH    H
E865 216900             LXI     H,105
E868 22E43C             SHLD    ISIZE
E86B 21E63C             LXI     H,ITRK
E86E 3601               MVI     M,1     ;TRACK=1
E870 2C                 INR     L
E871 3600               MVI     M,0     ;SECTOR=0
E873 2C                 INR     L       ;READ BFR EMPTY
E874 3600               MVI     M,0
E876 CD93E8             CALL    REDX
E879 C9                 RET             ;GO TO FDOS
                        ;
                        ;
                        ;
E87A 2AEE3C   RESTR:    LHLD    TISZE   ;RESTORE IFILE POINTERS
E87D 22E43C             SHLD    ISIZE
E880 3ADF3C             LDA     TITRK
E883 32E63C             STA     ITRK
E886 3AE33C             LDA     IUNIT
E889 0F                 RRC
E88A 0F                 RRC
E88B 32E73C             STA     ISCTR
E88E 97                 SUB     A
E88F 32E83C             STA     ICNTR
E892 C9                 RET
                        ;
                        ;
                        ;SUBROUTINE TO READ A HEX FILE INTO MEMORY
                        ;STARTS WITH ROUTINE REDO, USES ALL REGISTERS
                        ;
                        ;
E893 E1       REDX:     POP     H       ;SWAP BIAS & RETURN
E894 E3                 XTHL
E895 E5                 PUSH    H
E896 E1       REDO:     POP     H       ;GET BIAS
E897 E5                 PUSH    H
E898 CD08E9             CALL    RIX     ;GET CHAR INTO A
E89B 063A               MVI     B,':'
E89D 90                 SUB     B
E89E C296E8             JNZ     REDO
E8A1 57                 MOV     D,A
E8A2 CDD7E8             CALL    BYTE
E8A5 CAC8E8             JZ      RED2
E8A8 5F                 MOV     E,A
E8A9 CDD7E8             CALL    BYTE
E8AC F5                 PUSH    PSW
```

```
E8AD CDD7E8          CALL    BYTE
E8B0 C1              POP     B
E8B1 4F              MOV     C,A
E8B2 09              DAD     B
E8B3 CDD7E8          CALL    BYTE
E8B6 CDD7E8  RED1:   CALL    BYTE
E8B9 77              MOV     M,A
E8BA 23              INX     H
E8BB 1D              DCR     E
E8BC C2B6E8          JNZ     RED1
E8BF CDD7E8          CALL    BYTE
E8C2 C2EEE8          JNZ     LER
E8C5 C396E8          JMP     RED0
E8C8 CDD7E8  RED2:   CALL    BYTE
E8CB 67              MOV     H,A
E8CC CDD7E8          CALL    BYTE
E8CF 6F              MOV     L,A
E8D0 B4              ORA     H
E8D1 CAD5E8          JZ      RED3
E8D4 E9              PCHL
E8D5 E1      RED3:   POP     H
E8D6 C9              RET
                ;
                ;
                ;
E8D7 CD08E9  BYTE:   CALL    RIX
E8DA CDF6E8          CALL    NBL
E8DD 07              RLC
E8DE 07              RLC
E8DF 07              RLC
E8E0 07              RLC
E8E1 4F              MOV     C,A
E8E2 CD08E9          CALL    RIX
E8E5 CDF6E8          CALL    NBL
E8E8 B1              ORA     C
E8E9 4F              MOV     C,A
E8EA 82              ADD     D
E8EB 57              MOV     D,A
E8EC 79              MOV     A,C
E8ED C9              RET
                ;
                ;
                ;
                ;SUBROUTINE IF LOAD ERROR OCCURRED

E8EE 0E3F    LER:    MVI     C,'?'
E8F0 CD06E8          CALL    CO
E8F3 C312E8          JMP     MNTR
```

```
                ;SUBROUTINE TO CONVERT TWO HEX CHARACTERS
                ;   TO ONE BYTE

E8F6 D630       NBL:    SUI     '0'
E8F8 D8                 RC
E8F9 C6E9               ADI     0E9H
E8FB D8                 RC
E8FC C606               ADI     6
E8FE F204E9             JP      NIO
E901 C607               ADI     7
E903 D8                 RC
E904 C60A       NIO:    ADI     10
E906 B7                 ORA     A
E907 C9                 RET


                ;
                ;
                ;
                ;
                ;
                ;SUBROUTNE TO READ A BYTE FROM DISK
                ;PLACES CHAR IN A-REG.  ENTRY AT RI READS 8 BITS,
                ;ENTRY AT RIX READS 7 BITS.
                ;
                ;
E908 CD11E9     RIX:    CALL    RI
E90B DA12E8             JC      MNTR
E90E E67F               ANI     7FH
E910 C9                 RET
                ;
                ;
E911 C5         RI:     PUSH    B       ;SAVE REG D-L
E912 E5                 PUSH    H
E913 21E83C             LXI     H,ICNTR
E916 7E                 MOV     A,M
E917 A7                 ANA     A       ;CNT=0?
E918 C276E9             JNZ     RI10    ;NO
E91B 2EE7       RI5:    MVI     L,ISCTR AND 0FFH        ;YES-INCR D.A.
E91D CD8AE9             CALL    INCDA
E920 2AE43C             LHLD    ISIZE
E923 2B                 DCX     H
E924 22E43C             SHLD    ISIZE
E927 7D                 MOV     A,L
E928 A7                 ANA     A
E929 C23EE9             JNZ     RI3
E92C 7C                 MOV     A,H
E92D A7                 ANA     A
E92E C23EE9             JNZ     RI3
E931 23                 INX     H
```

```
E932 22E43C          SHLD    ISIZE
E935 21E83C          LXI     H,ICNTR
E938 3600            MVI     M,0
E93A 37              STC             ;SET EOF
E93B E1        RI2:  POP     H       ;RESTORE D-L
E93C C1              POP     B
E93D C9              RET
                ;
E93E 21E73C    RI3:  LXI     H,ISCTR ;XMIT UNIT/SECTOR
E941 CDE8E9          CALL    XUS
E944 CD30EA          CALL    CHK     ;MAKE SURE A DISK
E947 2C              INR     L       ;SET CNTR=128
E948 3680            MVI     M,128
E94A 0E05            MVI     C,5     ;SET TRY CNT=5
E94C 2EE6            MVI     L,ITRK AND OFFH ;SEEK TRACK
E94E CDFEE9          CALL    SEEK
E951 3E03      RI6:  MVI     A,3     ;READ DATA
E953 CD0DEA          CALL    LOOP
E956 DB07            IN      DATAI   ;DD MARK?
E958 E680            ANI     80H
E95A CA63E9          JZ      RI4     ;NO
E95D CD0BEA          CALL    RFLAG
E960 C31BE9          JMP     RI5
E963 DB07      RI4:  IN      DATAI   ;CRC ERROR?
E965 E608            ANI     8H
E967 CA76E9          JZ      RI10    ;NO
E96A CD0BEA          CALL    RFLAG
E96D 0D              DCR     C       ;DECR CNTR
E96E C251E9          JNZ     RI6
E971 3E01            MVI     A,1
E973 C337EA          JMP     CHK1
                ;
E976 3E40      RI10: MVI     A,40H   ;READ BYTE INTO A
E978 D306            OUT     CNTRL
E97A DB07            IN      DATAI
E97C 4F              MOV     C,A
E97D 3E41            MVI     A,41H   ;STROBE BUFFER
E97F CD0DEA          CALL    LOOP
E982 2EE8            MVI     L,ICNTR AND OFFH        ;DECR   READ COUNTER
E984 35              DCR     M
E985 79              MOV     A,C
E986 B7              ORA     A
E987 C33BE9          JMP     RI2
                ;
                ;
                ;
                ;ROUTINE TO INCREMENT DISK ADDRESS
                ;
E98A 34        INCDA: INR    M
E98B 7E              MOV     A,M
```

```
E98C E61F            ANI     1FH
E98E FE13            CPI     27
E990 CA95E9          JZ      INCDB
E993 2D              DCR     L
E994 C9              RET
E995 7E      INCDB:  MOV     A,M
E996 E6C1            ANI     0C1H
E998 77              MOV     M,A
E999 2D              DCR     L
E99A 34              INR     M
E99B C9              RET
             ;
             ;
             ;SUBROUTINE TO WRITE A BYTE TO DISK
             ;EXPECTS CHAR TO BE IN C-REG
             ;
E99C 79      WRT:    MOV     A,C
E99D E5              PUSH    H
E99E D307            OUT     DATAO   ;OUTPUT HAR
E9A0 3E31            MVI     A,31H
E9A2 CD0DEA          CALL    LOOP
E9A5 21ED3C          LXI     H,OCNTR
E9A8 34              INR     M       ;INCREMENT BFR CNT
E9A9 7E              MOV     A,M
E9AA FE80            CPI     128     ;=128?
E9AC C2E6E9          JNZ     WRT4    ;NO
E9AF 3600            MVI     M,0     ;CLEAR COUNT
E9B1 21EC3C  WRT1:   LXI     H,OSCTR ;XMIT UNIT/SECTOR
E9B4 CDE3E9          CALL    XUS
E9B7 CD30EA          CALL    CHK     ;MAKE SURE A DISK
E9BA 0E05            MVI     C,5     ;SET TRY CNT=5
E9BC 2D              DCR     L       ;SEEK TRACK
E9BD CDFEE9          CALL    SEEK
E9C0 3E05    WRT2:   MVI     A,5     ;WRITE DATA
E9C2 CD0DEA          CALL    LOOP
E9C5 3E07            MVI     A,7     ;READ FOR CRC
E9C7 CD0DEA          CALL    LOOP
E9CA DB07            IN      DATAI   ;CRC ERROR?
E9CC E608            ANI     8H
E9CE CAE3E9          JZ      WRT3    ;NO
E9D1 CD0BEA          CALL    RFLAG
E9D4 0D              DCR     C       ;DECR TRY CNT
E9D5 C2C0E9          JNZ     WRT2    ;TRY AGAIN
E9D8 3E0F            MVI     A,0FH   ;WRITE AS DD
E9DA CD0DEA          CALL    LOOP
E9DD CD19EA          CALL    WRTN    ;INCREMENT DA & CHK SIZE
E9E0 C3B1E9          JMP     WRT1
E9E3 CD19EA  WRT3:   CALL    WRTN    ;INCREMENT DA & CHK SIZE
E9E6 E1      WRT4:   POP     H       ;RESTORE D-L
```

```
E9E7 C9             RET
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO TRANSMIT UNIT/SECTOR BYTE
                    ;
E9E8 7E       XUS:  MOV   A,M
E9E9 E61F           ANI   1FH     ;EXTRACT LOG SECTOR
E9EB E5             PUSH  H
E9EC 2166EA         LXI   H,TBL-1 ;GET TABLE PNTR
E9EF 85             ADD   L       ;MAKE SECTOR PNTR
E9F0 6F             MOV   L,A
E9F1 4E             MOV   C,M     ;GET PHYS SECTOR
E9F2 E1             POP   H
E9F3 7E             MOV   A,M
E9F4 E6C0           ANI   0C0H
E9F6 B1             ORA   C       ;MERGE UNIT & PHYS SCTR
E9F7 D307     XXUS: OUT   DATAO
E9F9 3E21           MVI   A,21H
E9FB C30DEA         JMP   LOOP
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO SEEK TRACK IN A
                    ;
E9FE 7E       SEEK: MOV   A,M
E9FF D307           OUT   DATAO
EA01 3E11           MVI   A,11H
EA03 CD0DEA         CALL  LOOP
EA06 3E09           MVI   A,09
EA08 C30DEA         JMP   LOOP
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO RESET FLAG
                    ;
EA0B 3E0B     RFLAG: MVI    A,0BH
                    ;SUBROUTINE TO ISSUE CMD & LOOP ON BUSY
                    ;
EA0D D306     LOOP: OUT     CNTRL
EA0F 97             SUB   A
EA10 D306           OUT   CNTRL
EA12 DB07     LOOP1: IN     DATAI
EA14 1F             RAR
EA15 DA12EA         JC    LOOP1
EA18 C9             RET
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO INCR DISK ADDR & CHK OFILE SIZE
```

```
                        ;
EA19 2EEC       WRTN:   MVI     L,OSCTR AND 0FFH
EA1B CD8AE9             CALL    INCDA
EA1E 2AE93C     WRTN2:  LHLD    OSIZE
EA21 2B                 DCX     H
EA22 22E93C             SHLD    OSIZE
EA25 7D                 MOV     A,L
EA26 A7                 ANA     A
EA27 C0                 RNZ
EA28 7C                 MOV     A,H
EA29 A7                 ANA     A
EA2A C0                 RNZ
EA2B 3E02               MVI     A,2
EA2D C337EA             JMP     CHK1
                        ;
                        ;
                        ;
                        ;SUBROUTINE TO CHECK IF A DISK, ELSE ERR03
                        ;
EA30 DB07       CHK:    IN      DATAI
EA32 E620               ANI     20H
EA34 C8                 RZ
EA35 3E03               MVI     A,3
                        ;ROUTINE TO PRINT ERR(E)
EA37 F630       CHK1:   ORI     30H     ;CONVERT TO ASCII
EA39 4F                 MOV     C,A
EA3A CD06E8             CALL    CO
EA3D C312E8             JMP     MNTR
                        ;
                        ;
                        ;
                        ;INTERPASS FUNCTIONS
                        ;IF BIT 0 OF (PASS) IS EQUAL TO 1, THEN BIT 0 OF
                        ;   (PASS) IS SET TO 0 AND 31H, ASCII 1, IS RETURNED IN
                        ;   A-REG.  IF BIT 0 OF (PASS) IS EQUAL TO 0, THEN (PASS)
                        ;   IS SET TO 00 AND 30H, ASCII 0, PLUS (PASS) SHIFTED
                        ;   RIGHT 1 BIT POSITION IS RETURNED IN A-REG.  IF (PASS)
                        ;   IS EQUAL TO 00, JMP UPDAT OCCURS.

EA40 3AE03C     IPASS:  LDA     PASS
EA43 1F                 RAR
EA44 D253EA             JNC     PASS2
EA47 3AE03C             LDA     PASS
EA4A 3D                 DCR     A
EA4B 32E03C             STA     PASS
EA4E 3E01               MVI     A,1
EA50 C364EA             JMP     PASS3
EA53 A7         PASS2:  ANA     A
EA54 CA45E8             JZ      UPDAT
EA57 CD7AE8             CALL    RESTR
```

```
EA5A 3AE03C          LDA     PASS
EA5D 1F              RAR
EA5E F5              PUSH    PSW
EA5F 97              SUB     A
EA60 32E03C          STA     PASS
EA63 F1              POP     PSW
EA64 C630    PASS3:  ADI     30H
EA66 C9              RET

                     ;PHYSICAL SECTOR TABLE.  IS IN ORDER
                     ;    OF LOGICAL SECTOR NUMBER.
                     ;
EA67 01      TBL:    DB      1
EA68 0A              DB      0AH
EA69 13              DB      13H
EA6A 02              DB      2
EA6B 0B              DB      0BH
EA6C 14              DB      14H
EA6D 03              DB      3
EA6E 0C              DB      0CH
EA6F 15              DB      15H
EA70 04              DB      4
EA71 0D              DB      0DH
EA72 16              DB      16H
EA73 05              DB      5
EA74 0E              DB      0EH
EA75 17              DB      17H
EA76 06              DB      6
EA77 0F              DB      0FH
EA78 18              DB      18H
EA79 07              DB      7
EA7A 10              DB      10H
EA7B 19              DB      19H
EA7C 08              DB      8
EA7D 11              DB      11H
EA7E 1A              DB      1AH
EA7F 09              DB      9
EA80 12              DB      12H
EA81 00              NOP
EA82 00              NOP
EA83 00              NOP
                     END
```

APPENDIX B

FDOS-II FOR SBC/8800/ALTAIR/IMSAI/POLY88

OPERATOR'S MANUAL

```
                ;
0000            FROM    EQU     0C000H
                ;*****
                ;
                ;RESIDENT BOOS AND FDOSII  VERSION 1.0
                ;
                ;*****
                ;
                ;
                ;ENTRY ADDRESSES-
                ;       POWER UP = 03E7 HEX
                ;       RE-ENTRY = 03E4 HEX
                ;
                ;
                ;
00C1            DATAO   EQU     0C1H
00C0            DATAI   EQU     0C0H
00C0            CNTRL   EQU     0C0H
0000            CCTRL   EQU     0       ;CONSOLE CONTROL PORT
0001            CDATA   EQU     1       ;CONSOLE DATA PORT
0001            CRRDY   EQU     1       ;CONSOLE DATA READY
0080            CTRDY   EQU     80H     ;CONSOLE XMIT READY
                ;
                ;
0400            SCTCH   EQU     0C400H  ;SCRATCH RAM
0400            VCTRS   EQU     SCTCH   ;I/O VECTORS
0430            BASE    EQU     SCTCH+30H
047F            STACK   EQU     SCTCH+7FH
                ;
                ;

0430            PASS    EQU     BASE
0431            OFILE   EQU     BASE+1
0432            OUNIT   EQU     BASE+2
0433            IUNIT   EQU     BASE+3
0434            ISIZE   EQU     BASE+4
0436            ITRK    EQU     BASE+6
0437            ISCTR   EQU     BASE+7
0438            ICNTR   EQU     BASE+8
0439            OSIZE   EQU     BASE+9
043B            OTRK    EQU     BASE+11
043C            OSCTR   EQU     BASE+12
043D            OCNTR   EQU     BASE+13
042F            TITRK   EQU     BASE-1
043E            TISZE   EQU     BASE+14

0418            ASMB    EQU     VCTRS+24
041B            START   EQU     VCTRS+27
041E            UPDTX   EQU     VCTRS+30
```

```
0000                    ORG     PROM

                        ;
                        ;
                        ;
                        ;ENTRY POINT WHEN R IS TYPED
                        ;LOADS FDOS AND BRANCHES TO FDOS S.A.

0000 C31500             ;JMP    FDOS

0003 C30004     CI      ;JMP    VCTRS   ;KEYBOARD INPUT VECTOR

0006 C30304     CO      ;JMP    VCTRS+3 ;CONSOLE OUTPUT VECTOR

0009 C30604     RDRIN   ;JMP    VCTRS+6 ;READER INPUT VECTOR

000C C30904     LO      ;JMP    VCTRS+9 ;LIST OUTPUT VECTOR

000F C30C04     PO      ;JMP    VCTRS+12        ;PUNCH OUTPUT VECTOR

0012 C30F04     MNTR    ;JMP    VCTRS+15        ;SYSTEM MONITOR VECTOR

0015 317F04     FDOS    LXI     SP,STACK
0018 CD5EC0             CALL    FDOS1
001B C31504             ;JMP    START

                        ;
001E C35403     RSTV    ;JMP    RESET
0021 C3E601     XUSV    ;JMP    XUS
0024 C3EF01     XXUSV   ;JMP    XXUS
0027 C3F701     SEEKV   ;JMP    SEEK+1
002A C30D02     RFLGV   ;JMP    RFLAG
002D C30502     LOOPV   ;JMP    ;LOOP
0030 C37A01     RSTRV   ;JMP    RESTR

                        ;
                        ;
                        ;
                        ;
                        ;
0033 C30901     RIV     ;JMP    RI
                        ;
0036 C39401     WRTV    ;JMP    WRT
                        ;
                        ;
0039 C33802     PASSV   ;JMP    IPASS   ;ASMB INTERPASS FNC

003C CD9300     ASSEM   CALL    REDX
003F CD7A00             CALL    RESTR
0042 C31904             ;JMP    ASMB
```

```
                        ;
0045 317FC4     UPDAT:  LXI     SP,STACK
0048 CD5EC0             CALL    FDOS1
004B C31EC4             JMP     UPDTX
                        ;
                        ;
004E CD93C0     PROG:   CALL    REDX
0051 C312C0             JMP     MNTR
                        ;
                        ;
0054 3E81       RESET:  MVI     A,81H
0056 CD05C2             CALL    LOOP
0059 3E0D               MVI     A,0DH
005B C305C2             JMP     LOOP
                        ;
005E CD54C0     FDOS1:  CALL    RESET
0061 210000             LXI     H,0     ;SET BIAS=0
0064 E5                 PUSH    H
0065 216900             LXI     H,105
0068 2234C4             SHLD    ISIZE
006B 2136C4             LXI     H,ITRK
006E 3601               MVI     M,1     ;TRACK=1
0070 2C                 INR     L
0071 3600               MVI     M,0     ;SECTOR=0
0073 2C                 INR     L       ;READ BFR EMPTY
0074 3600               MVI     M,0
0076 CD93C0             CALL    REDX
0079 C9                 RET             ;GO TO FDOS
                        ;
                        ;
                        ;
007A 2A3EC4     RESTR:  LHLD    TISZE   ;RESTORE IFILE POINTERS
007D 2234C4             SHLD    ISIZE
0080 3A2FC4             LDA     TITRK
0083 3236C4             STA     ITRK
0086 3A33C4             LDA     IUNIT
0089 0F                 RRC
008A 0F                 RRC
008B 3237C4             STA     ISCTR
008E 97                 SUB     A
008F 3238C4             STA     ICNTR
0092 C9                 RET
                        ;
                        ;
                        ;
                        ;SUBROUTINE TO READ A HEX FILE INTO MEMORY
                        ;STARTS WITH ROUTINE REDO, USES ALL REGISTERS
```

```
                        ;
00A3 E3         REDX    POP     H       ;SWAP BIAS & RETURN
00A4 E3                 XTHL
00A5 E5                 PUSH    H
00A6 E1         REDO    POP     H       ;GET BIAS
00A7 E5                 PUSH    H
00A8 CD00C1             CALL    RIX     ;GET CHAR INTO A
00AB D630               MVI     B,'0'
00AD 90                 SUB     B
00AE C2A100             JNZ     REDO
00B1 57                 MOV     D,A
00B2 CDD700             CALL    BYTE
00B5 CAC800             JZ      RED2
00B8 5F                 MOV     E,A
00B9 CDD700             CALL    BYTE
00BC F5                 PUSH    PSW
00BD CDD700             CALL    BYTE
00C0 C1                 POP     B
00C1 4F                 MOV     C,A
00C2 09                 DAD     B
00C3 CDD700             CALL    BYTE
00C6 CDD700     RED1    CALL    BYTE
00C9 77                 MOV     M,A
00CA 03                 INX     H
00CB 1D                 DCR     E
00CC C2C600             JNZ     RED1
00CF CDD700             CALL    BYTE
00D2 C2D100             JNZ     LFB
00D5 C39600             JMP     REDO
00D8 CDD700     RED2:   CALL    BYTE
00DB 67                 MOV     H,A
00DC CDD700             CALL    BYTE
00DF 6F                 MOV     L,A
00E0 B4                 ORA     H
00E1 CAE500             JZ      REDS
00E4 E9                 PCHL
00E5 E1         REDS    POP     H
00E6 C9                 RET
                        ;
                        ;
                        ;
00E7 CD00C1     BYTE:   CALL    RIX
00EA CDEEC0             CALL    NBL
00ED 07                 RLC
00EE 07                 RLC
00EF 07                 RLC
00F0 07                 RLC
00F1 4F                 MOV     C,A
00F2 CD00C1             CALL    RIX
```

```
00E5 CDEEC0         CALL    NBL
00E8 B1             ORA     C
00E9 4F             MOV     C,A
00EA 82             ADD     D
00EB 57             MOV     D,A
00EC 79             MOV     A,C
00ED C9             RET
                    ;
                    ;SUBROUTINE TO CONVERT TWO HEX CHARACTERS
                    ;   TO ONE BYTE
00EE D630   NBL     SUI     '0'
00F0 D8             RC
00F1 C6E9           ADI     0E9H
00F3 D8             RC
00F4 C60A           ADI     6
00F6 F2FDC0         JP      NIO
00F9 C607           ADI     7
00FB D8             RC
00FC C60A   NIO     ADI     10
00FE B7             ORA     A
00FF C9             RET
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;SUBROUTNE TO READ A BYTE FROM DISK
                    ;PLACES CHAR IN A-REG.  ENTRY AT RI READS 8 BITS,
                    ;ENTRY AT RIX READS 7 BITS
                    ;
                    ;
0100 CD09C1 RIX     CALL    RI
0103 DA12C0         JC      MNTR
0106 E67F           ANI     7FH
0108 C9             RET
                    ;
                    ;
0109 C5     RI:     PUSH    B       ;SAVE REG D-L
010A E5             PUSH    H
010B 2138C4         LXI     H,ICNTR
010E 7E             MOV     A,M
010F A7             ANA     A       ;CNT=0?
0110 C26EC1         JNZ     RI10    ;NO
0113 2E37   RI5:    MVI     L,ISCTR AND 0FFH        ;YES-INCR D.A.
```

```
0115 CD6CC1          CALL    INCDA
0118 2A7404          LHLD    ISIZE
011B 2B              DCX     H
011C 227404          SHLD    ISIZE
011F 7D              MOV     A,L
0120 A7              ANA     A
0121 C23601          JNZ     RI3
0124 7C              MOV     A,H
0125 A7              ANA     A
0126 C23601          JNZ     RI3
0129 23              INX     H
012A 227404          SHLD    ISIZE
012D 213804          LXI     H,ICNTR
0130 3600            MVI     M,0
0132 37              STC             ;SET EOF
0133 F1      RI2:    POP     H       ;RESTORE D-L
0134 C1              POP     B
0135 C9              RET
             ;
0136 213704  RI3:    LXI     H,ISCTR ;XMIT UNIT/SECTOR
0139 CDE0C1          CALL    XUS
013C CD28C2          CALL    CHK     ;MAKE SURE A DISK
013F 2C              INR     L       ;SET CNTR=128
0140 3680            MVI     M,128
0142 0E05            MVI     C,5     ;SET TRY CNT=5
0144 3E36            MVI     L,ITRK AND 0FFH ;SEEK TRACK
014A CDF6C1          CALL    SEEK
0149 3E02    RI6:    MVI     A,3     ;READ DATA
014B CD05C2          CALL    LOOP
014E DBC0            IN      DATAI   ;DD MARK?
0150 E680            ANI     80H
0152 CA5BC1          JZ      RI4     ;NO
0155 CD03C2          CALL    RFLAG
0158 C31BC1          JMP     RI5
015B DBC0    RI4:    IN      DATAI   ;CRC ERROR?
015D E608            ANI     8H
015F CA6EC1          JZ      RI10    ;NO
0162 CD03C2          CALL    RFLAG
0165 0D              DCR     C       ;DECR CNTR
0166 C249C1          JNZ     RI6
0169 3E01            MVI     A,1
016B C32FC2          JMP     CHK1
             ;
016E 3E40    RI10:   MVI     A,40H   ;READ BYTE INTO A
0170 D3C0            OUT     CNTRL
0172 DBC0            IN      DATAI
0174 4F              MOV     C,A
0175 3E41            MVI     A,41H   ;STROBE BUFFER
0177 CD05C2          CALL    LOOP
017A 2E38            MVI     L,ICNTR AND 0FFH        ;DECR   READ COUNTER
```

```
C17C 35             DCR     M
C17D 79             MOV     A,C
C17E B7             ORA     A
C17F C333C1         JMP     RTZ

                    ;
                    ;
                    ;
                    ;ROUTINE TO INCREMENT DISK ADDRESS
                    ;
C182 34       INCDA:INR     M
C183 7E             MOV     A,M
C184 E61F           ANI     1FH
C186 FE1B           CPI     27
C188 CA8DC1         JZ      INCDB
C18B 2D             DCR     L
C18C C9             RET
C18D 7E       INCDB:MOV     A,M
C18E E6C1           ANI     0C1H
C190 77             MOV     M,A
C191 2D             DCR     L
C192 34             INR     M
C193 C9             RET
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO WRITE A BYTE TO DISK
                    ;EXPECTS CHAR TO BE IN C-REG
                    ;
C194 79       WRT   MOV     A,C
C195 E5             PUSH    H
C196 D3C1           OUT     DATAO   ;OUTPUT WAR
C198 3E31           MVI     A,31H
C19A CD05C2         CALL    LOOP
C19D 213DC4         LXI     H,QCNTR
C1A0 34             INR     M       ;INCREMENT BFR CNT
C1A1 7E             MOV     A,M
C1A2 FE80           CPI     128     ;=128?
C1A4 C2DEC1         JNZ     WRT4    ;NO
C1A7 3A00           MVI     M,0     ;CLEAR COUNT
C1A9 213CC4   WRT1: LXI     H,OSCTR ;XMIT UNIT/SECTOR
C1AC CDF0C1         CALL    XUS
C1AF CD28C2         CALL    CHK     ;MAKE SURE A DISK
C1B2 0E05           MVI     C,5     ;SET TRY CNT=5
C1B4 2D             DCR     L       ;SEEK TRACK
C1B5 CDF6C1         CALL    SEEK
C1B8 3E05     WRT2: MVI     A,5     ;WRITE DATA
C1BA CD05C2         CALL    LOOP
C1BD 3E07           MVI     A,7     ;READ FOR CRC
C1BF CD05C2         CALL    LOOP
C1C2 DBC0           IN      DATAI   ;CRC ERROR?
```

```
C1C4 E606           ANI     06H
C1C6 C4A8C1         JZ      WRT3    ;NO
C1C9 CD03C2         CALL    RFLAG
C1CC 0D             DCR     C       ;DECR TRY CNT
C1CD C2B8C1         JNZ     WRT2    ;TRY AGAIN
C1D0 3E0F           MVI     A,0FH   ;WRITE AS DD
C1D2 CD05C2         CALL    LOOP
C1D5 CD11C2         CALL    WRTN    ;INCREMENT DA & CHK SIZE
C1D8 C3A3C1         JMP     WRT1
C1DB CD11C2   WRT3: CALL    WRTN    ;INCREMENT DA & CHK SIZE
C1DE E1       WRT4: POP     H       ;RESTORE D-L
C1DF C9             RET
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO TRANSMIT UNIT/SECTOR BYTE
                    ;
C1E0 7E       XUS:  MOV     A,M
C1E1 E61F           ANI     1FH     ;EXTRACT LOG SECTOR
C1E3 E5             PUSH    H
C1E4 215EC2         LXI     H,TBL-1 ;GET TABLE PNTR
C1E7 85             ADD     L       ;MAKE SECTOR PNTR
C1E8 6F             MOV     L,A
C1E9 4E             MOV     C,M     ;GET PHYS SECTOR
C1EA E1             POP     H
C1EB 7E             MOV     A,M
C1EC E6C0           ANI     0C0H
C1EE B1             ORA     C       ;MERGE UNIT & PHYS SCTR
C1EF D3C1     XXUS: OUT     DATAO
C1F1 3E21           MVI     A,21H
C1F3 C305C2         JMP     LOOP
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO SEEK TRACK IN A
                    ;
C1F6 7E       SEEK: MOV     A,M
C1F7 D3C1           OUT     DATAO
C1F9 3E11           MVI     A,11H
C1FB CD05C2         CALL    LOOP
C1FE 3E09           MVI     A,09
C200 C305C2         JMP     LOOP
                    ;
                    ;
                    ;
                    ;SUBROUTINE TO RESET FLAG
                    ;
C203 3E0B     RFLAG: MVI    A,0BH
                    ;SUBROUTINE TO ISSUE CMD & LOOP ON BUSY
                    ;
```

```
C207 D3C0     LOOP    OUT     CNTRL
C209 97               SUB     A
C20A D3C0             OUT     CNTRL
C20C DBC0     LOOP1:  IN      DATAI
C20E 1F               RAR
C20F DA0AC2           JC      LOOP1
C212 C9               RET

;
;
;SUBROUTINE TO INCR DISK ADDR & CHK OFILE SIZE
;
C213 2E3C     WRTN    MVI     L,OSCTR AND 0FFH
C215 CD62C1           CALL    INCDA
C218 2A89C4   WRTN2   LHLD    OSIZE
C21B 2B               DCX     H
C21C 2289C4           SHLD    OSIZE
C21F 7D               MOV     A,L
C220 A7               ANA     A
C221 C0               RNZ
C222 7C               MOV     A,H
C223 A7               ANA     A
C224 C0               RNZ
C225 3E02             MVI     A,2
C227 C32FC2           JMP     CHK1

;
;
;SUBROUTINE TO CHECK IF A DISK, ELSE ERR03
;
C22A DBC0     CHK     IN      DATAI
C22C E620             ANI     20H
C22E C8               RZ
C22F 3E03             MVI     A,3
;ROUTINE TO PRINT ERR(E)
C231 F630     CHK1:   ORI     30H     ;CONVERT TO ASCII
C233 4F               MOV     C,A
C234 CD06C0           CALL    CO
C237 C312C0           JMP     MNTR

;
;
;
;INTERPASS FUNCTIONS
;IF BIT 0 OF (PASS) IS EQUAL TO 1, THEN BIT 0 OF
;   (PASS) IS SET TO 0 AND 31H, ASCII 1, IS RETURNED IN
;   A-REG   IF BIT 0 OF (PASS) IS EQUAL TO 0, THEN (PASS)
;   IS SET TO 00 AND 30H, ASCII 0, PLUS (PASS) SHIFTED
;   RIGHT 1 BIT POSITION IS RETURNED IN A-REG.  IF (PASS)
;   IS EQUAL TO 00, JMP UPDAT OCCURS.
```

```
023B 3A3004   IPASS:  LDA     PASS
023E 1F               RAR
023F D24B02           JNC     PASS2
0242 3D               DCR     A
0243 323004           STA     PASS
0246 3E01             MVI     A,1
0248 C35002           JMP     PASS3
024B A7       PASS2:  ANA     A
024C CA4500           JZ      UPDAT
024F CD7400           CALL    RESTR
0252 3A3004           LDA     PASS
0255 1F               RAR
0256 F5               PUSH    PSW
0257 97               SUB     A
0258 323004           STA     PASS
025B F1               POP     PSW
025C C630     PASS3:  ADI     30H
025E C9               RET

                      ;PHYSICAL SECTOR TABLE.  IS IN ORDER
                      ;   OF LOGICAL SECTOR NUMBER.
                      ;
025F 01       TBL:    DB      1
0260 0A               DB      0AH
0261 13               DB      13H
0262 02               DB      2
0263 0B               DB      0BH
0264 14               DB      14H
0265 03               DB      3
0266 0C               DB      0CH
0267 15               DB      15H
0268 04               DB      4
0269 0D               DB      0DH
026A 16               DB      16H
026B 05               DB      5
026C 0E               DB      0EH
026D 17               DB      17H
026E 06               DB      6
026F 0F               DB      0FH
0270 18               DB      18H
0271 07               DB      7
0272 10               DB      10H
0273 19               DB      19H
0274 08               DB      8
0275 11               DB      11H
0276 1A               DB      1AH
0277 09               DB      9
0278 12               DB      12H
```

```
C279 00             NOP
C27A 00             NOP
C27B 00             NOP
                    ;
                    ;
                    ;
                    ;
                    ;       CONSOLE INPUT ROUTINE
                    ;
C27C DB00     CIX:  IN      CCTRL
C27E E601           ANI     CRRDY
C280 C27CC2         JNZ     CIX
C283 DB01           IN      CDATA
C285 E67F           ANI     7FH
C287 C9             RET
                    ;
                    ;
                    ;
C288 0E0D     CRLF: MVI     C,0DH
C28A CD06C0         CALL    CO
C28D 0E0A           MVI     C,0AH
C28F C306C0         JMP     CO
                    ;
                    ;       CONSOLE OUTPUT ROUTINE
                    ;
C292 DB00     COX:  IN      CCTRL
C294 E680           ANI     CTRDY
C296 C292C2         JNZ     COX
C299 79             MOV     A,C
C29A D301           OUT     CDATA
C29C C9             RET
                    ;
                    ;
C29D 010BC3   INIT: LXI     B,0C30BH
C2A0 2100C4         LXI     H,VCTRS
C2A3 11EAC3         LXI     D,VECTR
C2A6 70       INIT1 MOV     M,B
C2A7 23             INX     H
C2A8 1A             LDAX    D
C2A9 77             MOV     M,A
C2AA 23             INX     H
C2AB 13             INX     D
C2AC 1A             LDAX    D
C2AD 77             MOV     M,A
C2AE 23             INX     H
C2AF 13             INX     D
C2B0 0D             DCR     C
C2B1 C2A6C2         JNZ     INIT1
C2B4 317FC4   MNTRX:LXI     SP,STACK
C2B7 CD88C2         CALL    CRLF
```

```
C2BA 0E3E           MVI     C,3EH
C2BC CD06C0         CALL    CO
C2BF CDD9C2         CALL    CECHO
C2C2 FE54           CPI     'T'
C2C4 CA80C3         JZ      TSTM
C2C7 FE4D           CPI     'M'
C2C9 CA58C3         JZ      MEM
C2CC FE47           CPI     'G'
C2CE CAE1C2         JZ      GO
C2D1 0E3F     LER:  MVI     C,'?'
C2D3 CD06C0         CALL    CO
C2D6 C3B4C2         JMP     MNTRY
                    ;
                    ;
C2D9 CD03C0   CECHO CALL    CI
C2DC 4F             MOV     C,A
C2DD CD06C0         CALL    CO
C2E0 C9             RET
                    ;
                    ;
C2E1 CDE8C2   GO:   CALL    PARAM
C2E4 CD8BC2         CALL    CRLF
C2E7 E9             PCHL
                    ;
                    ;
C2E8 210000   PARAM LXI     H,0
C2EB CDD9C2   PARM1 CALL    CECHO
C2EE FE0D           CPI     0DH
C2F0 C8             RZ
C2F1 FE2C           CPI     ','
C2F3 C8             RZ
C2F4 29             DAD     H
C2F5 29             DAD     H
C2F6 29             DAD     H
C2F7 29             DAD     H
C2F8 DAD1C2         JC      LER
C2FB CDEEC0         CALL    NBL
C2FE DAD1C2         JC      LER
C301 B5             ORA     L
C302 6F             MOV     L,A
C303 C3EBC2         JMP     PARM1
                    ;
                    ;
C306 CDD9C2   BYTEC CALL    CECHO
C309 CDEEC0   BYTC1 CALL    NBL
C30C 07             RLC
C30D 07             RLC
C30E 07             RLC
C30F 07             RLC
C310 F5             PUSH    PSW
```

```
0311 CDD9C2          CALL    CECHO
0314 CDEEC0          CALL    NBL
0317 C1              POP     B
0318 B0              ORA     B
0319 C9              RET

                     ;
031A F5      BYTE0:  PUSH    PSW
031B CD2AC3          CALL    BYT01
031E 4F              MOV     C,A
031F CD06C0          CALL    CO
0322 F1              POP     PSW
0323 CD2EC3          CALL    BYT02
0326 4F              MOV     C,A
0327 C306C0          JMP     CO

                     ;
032A 0F      BYT01:  RRC
032B 0F              RRC
032C 0F              RRC
032D 0F              RRC
032E E60F    BYT02:  ANI     0FH
0330 FE0A            CPI     0AH
0332 FA37C3          JM      BYT03
0335 C607            ADI     7
0337 C630    BYT03:  ADI     30H
0339 C9              RET

                     ;
033A CD88C2  HLCO:   CALL    CRLF
033D 7C              MOV     A,H
033E CD1AC3          CALL    BYTE0
0341 7D              MOV     A,L
0342 CD1AC3          CALL    BYTE0
0345 C9              RET

                     ;
0346 CD3AC3  DSPYM:  CALL    HLCO
0349 0E3D            MVI     C,'='
034B CD06C0          CALL    CO
034E 7E              MOV     A,M
034F CD1AC3          CALL    BYTE0
0352 0E20            MVI     C,20H
0354 CD06C0          CALL    CO
0357 C9              RET

                     ;
0358 CDE8C2  MEM:    CALL    PARAM
035B CD46C3  MEM1:   CALL    DSPYM
035E CDD9C2          CALL    CECHO
```

```
C361 FE0D           CPI     0DH
C363 CAB4C2         JZ      MNTRX
C366 FE20           CPI     20H
C368 CA6FC3         JZ      MEM9
C36B CD09C3         CALL    BYTC1
C36E 77             MOV     M,A

C36F 23       MEM9: INX     H
C370 C35BC3         JMP     MEM1
                    ;
                    ;
C373 DB00     KBINT: IN     CCTRL
C375 E601           ANI     CRRDY
C377 C0             RNZ
C378 DB01           IN      CDATA
C37A FE03           CPI     3
C37C CA12C0         JZ      MNTR
C37F C9             RET
                    ;
                    ;
0421          HIGH  EQU     SCTCH+21H

C380 CDE8C2   TSTM: CALL    PARAM
C383 E5             PUSH    H
C384 EB             XCHG
C385 CDE8C2         CALL    PARAM
C388 222104         SHLD    HIGH
C38B EB             XCHG
C38C CD88C2         CALL    CRLF
C38F 3600     TSTM2: MVI    M,0
C391 7B             MOV     A,E
C392 BD             CMP     L
C393 C29BC3         JNZ     TSTM3
C396 7A             MOV     A,D
C397 BC             CMP     H
C398 CA9FC3         JZ      TSTM4
C39B 23       TSTM3 INX     H
C39C C38FC3         JMP     TSTM2

C39F 1E01     TSTM4 MVI     E,1
C3A1 E1       TSTM7: POP    H
C3A2 E5             PUSH    H
C3A3 34       TSTM1 INR     M
C3A4 7B             MOV     A,E
C3A5 BE             CMP     M
C3A6 C4C2C3         CNZ     TSTM6
C3A9 3A2104         LDA     HIGH
C3AC BD             CMP     L
C3AD C2BEC3         JNZ     TSTM5
C3B0 3A2204         LDA     HIGH+1
```

```
C3B3 BC             CMP     H
C3B4 C2BEC3         JNZ     TSTM5
C3B7 1C             INR     E
C3B8 CD73C3         CALL    KBINT
C3BB C3A1C3         JMP     TSTM7

C3BE 23       TSTM5:INX     H
C3BF C3A3C3         JMP     TSTM1

C3C2 CD46C3   TSTM6:CALL    DSPYM
C3C5 7B             MOV     A,E
C3C6 CD1AC3         CALL    BYTEO
C3C9 C388C2         JMP     CRLF



              ;
              ;
              ;
C3CC C303C0   RDIX: JMP     CI
              ;
C3CF C306C0   LOX:  JMP     CO
              ;
C3D2 C306C0   POX:  JMP     CO
              ,
C3E4                ORG     PROM+3E4H
              ,
              ;     I/O VECTOR TABLE
              ;
              ;***** MONITOR RE-ENTRY ADDRESS *********
C3E4 C3B4C2         JMP MNTRX
              ,
              ;
              ,****** MONITOR STARTING ADDRESS *********
C3E7 C39DC2         JMP     INIT    ;*************

              VECTR:DW      CIX     ;CONSOLE IN VECTOR
C3EC 92C2           DW      COX     ;CONSOLE OUT VECTOR
C3EE CCC3           DW      RDIX    ;PAPER TAPE READER VECTOR
C3F0 CFC3           DW      LOX     ;LINE PRINTER VECTOR
C3F2 D2C3           DW      POX     ;PUNCH VECTOR
C3F4 B4C2           DW      MNTRX   ;MONITOR VECTOR
C3F6 09C1           DW      RI      ;DISK READ VECTOR
C3F8 94C1           DW      WRT     ;DISK WRITE VECTOR
C3FA 4000           DW      40H     ;ASSEM/EDIT VECTOR;
C3FC 4000           DW      40H     ;EXECUTIVE VECTOR
C3FE 4300           DW      43H     ,UPDAT VECTOR
              ,
                    END
```

```
ASMB  C418    ASSEM C03C    BASE  C430    BYTC1 C309
BYTE  C0D7    BYTEC C306    BYTE0 C31A    BYT01 C32A
BYT02 C32E    BYT03 C337    CCTRL 0000    CDATA 0001
CECH0 C2D9    CHK   C226    CHK1  C22F    CI    C003
CIX   C27C    CNTRL 00C0    CO    C006    COX   C292
CRLF  C288    CRRDY 0001    CTRDY 0080    DATAI 00C0
DATA0 00C1    DSPYM C346    FDOS  C015    FDOS1 C05E
GO    C2E1    HIGH  C421    HLCO  C33A    ICNTR C438
INCDA C182    INCDB C18D    INIT  C29D    INIT1 C2A6
IPASS C238    ISCTR C437    ISIZE C434    ITRK  C436
IUNIT C433    KBINT C373    LER   C2D1    LO    C00C
LOGF  C205    LOGF1 C20A    LOOPV C02D    LOX   C3CF
MEM   C358    MEM1  C35B    MEM9  C36F    MNTR  C012
MNTRX C2B4    NBL   C0EE    NIO   C0FC    OCNTR C43D
OFILE C431    OSCTR C43C    OSIZE C439    OTRK  C43B
OUNIT C432    PARAM C2E8    PARM1 C2EB    PASS  C430
PASS2 C24B    PASS3 C25C    PASSV C039    PO    C00F
POX   C3D2    PROG  C04E    PROM  C000    RDIX  C3CC
RDRIN C009    RED0  C096    RED1  C0B6    RED2  C0C8
RED3  C0D5    REDX  C093    RESET C054    RESTR C07A
RFLAG C203    RFLGV C02A    RI    C109    RI10  C16E
RI2   C133    RI3   C136    RI4   C15B    RI5   C113
RI6   C149    RIV   C033    RIX   C100    RSTRV C030
RSTV  C01E    SCTCH C400    SEEK  C1F6    SEEKV C027
STACK C47F    START C41B    TBL   C25F    TISZE C43E
TITRK C42F    TSTM  C380    TSTM1 C3A3    TSTM2 C38F
TSTM3 C39B    TSTM4 C39F    TSTM5 C3BE    TSTM6 C3C2
TSTM7 C3A1    UPDAT C045    UPDTX C41E    VCTRS C400
VECTR C3EA    WRT   C194    WRT1  C1A9    WRT2  C1B8
WRT3  C1DB    WRT4  C1DE    WRTN  C211    WRTN2 C216
WRTV  C036    XUS   C1E0    XUSV  C021    XXUS  C1EF
XXUSV C024
```