PRODUCT SPECIFICATIONS

Intelligent Diskette Controller

Model 1070

TABLE OF CONTENTS

1.0 GENERAL DESCRIPTION

1.1 SUMMARY OF FEATURES

The PerSci Model 1070 is the first truly intelligent diskette controller. Can you imagine a controller which manipulates diskette files by name and provides the full functional capabilities of an advanced disk operating system, yet which requires no more support software in your microcomputer than does a paper tape reader or magnetic tape cassette drive? The Model 1070 accomplishes all of this on a single $4\frac{1}{2}$" X 7" circuit board through a combination of state-of-the-art LSI and microprocessor technology, advanced firmware techniques, and high-density packaging. The controller supports up to four PerSci Model 70 single diskette drives or up to two PerSci Model 277 dual diskette drives, providing a high-performance mass storage subsystem with an on-line capacity of more than one million bytes.

1.2 HARDWARE

The controller board incorporates a microprocessor and its associated support ICs, a single-chip LSI diskette drive controller, 4K bytes of ROM (optionally EPROM) containing the file management firmware, 1K bytes of RAM used for input/output buffering, an eight-bit parallel microcomputer interface, and an optional RS232 serial asynchronous interface. Required power for the controller (+5 volts and ±12 volts) can be derived from either the microcomputer or diskette drive power supplies.

1.3 FIRMWARE

The controller firmware resides in ROM on the controller board and performs the file management functions normally associated with the most advanced microcomputer disk operating systems. Supported functions include: diskette format initialization with optional sector interleave; maintaining and searching an index of files on each diskette; allocation and deallocation of diskette space; sequential, random, stream, and direct access; blocking and unblocking of both fixed-length and variable-length records; creating, deleting, renaming, and copying of files; error detection and error retry; and even performing diagnostic testing of the diskette drives. These file management functions are specified by means of a high-level controller command language. Minimal support software is needed in the host.

1.4 INTERFACES

Two alternative methods are provided for interfacing with the controller: parallel and serial. The parallel microcomputer interface includes a buffered eight-bit bidirectional data bus with handshake and address selection logic consistent with the interface requirements of most currently-available microprocessors including the 8080, 6800, Z80, etc. The optional RS232 serial asynchronous interface provides sixteen switch-selectable transmission speeds from 50 to 19,200 bits/second, interfacing directly with virtually any standard terminal, modem, or serial input/output port. One of these two alternative interfaces may be used at a time; when using a controller which includes the optional serial interface, the parallel interface is enabled simply by removing the USART chip from its socket.

1.5 DISKETTE FORMAT

The diskette initialization function of the controller creates a diskette format which is IBM 3740 compatible: each diskette contains 77 tracks with 26 sectors per track and 128 data bytes per sector. The first track is reserved by the controller for use as an index of files, while the remaining 76 tracks are available for data storage. Formatted capacity of each diskette is 252,928 bytes, excluding the index track.

1.6 COMPANION DISKETTE DRIVES

The PerSci Model 70 single diskette drive and Model 277 dual diskette drive incorporate many design features previously unique to large disk technology, resulting in unexcelled reliability and performance, small size, and fast access to data. The use of voice coil positioning provides access times which are 5-7 times faster than other available diskette drives with stepping motor positioners. Automatic motor-driven diskette load and unload assures simple and accurate diskette insertion and eliminates the possibility of diskette damage. Power consumption is one fourth of the power required by competitive drives, no cooling fan is required, and operation is virtually noiseless. Compact design permits five single drives or four dual drives to be mounted within the width of a 19" rack. The PerSci Model 1070 intelligent diskette controller is especially designed to take maximum advantage of the high-performance capabilities of these drives.

2.0 HARDWARE SPECIFICATIONS

2.1 PHYSICAL SPECIFICATIONS

The controller consists of a single printed circuit board with dimensions 4.50" X 7.00" which mates with edge connectors along the two 4.50" sides of the board. One edge connector has 72 pins (dual 36) with .100" spacing and carries the parallel interface, RS232 serial interface, and controller power lines. The other edge connector has 50 pins (dual 25) with .100" spacing and provides the interface with the diskette drive(s). The controller board is physically compatible with Vector Electronics plugboards and card cages with 72 pin connectors.

2.2 MICROCOMPUTER INTERFACE SPECIFICATIONS

2.2.1 Mating Connectors

The microcomputer interface is by means of an edge connector with 72 pins (dual 36) with .100" spacing (Amphenol 225-23621-201 or equivalent). In the listing below, all signals are TTL active high except those marked * are TTL active low and those marked ** are RS232 levels.

Pin No.	Signal Name	Pin No.	Signal Name
PARALLEL INTERFACE		RS232 SERIAL INTERFACE	
1 thru 8	Data Bus 0 thru 7	LL	Transmit Data**
E thru T	Addr Bus 4 thru 15	32	Receive Data**
27	External Select	KK	Data Terminal Ready**
18	Read*	31	Data Set Ready**
19	Write*	HH	Request to Send**
A	Status/Data	29	Clear to Send**
CONTROLLER RESET		CONTROLLER POWER	
17	Reset Controller*	RR, 36	Ground
U	Reset Complete	PP, 35	+5 Volts
		34	+12 Volts
		NN	-12 Volts

2.2.2 Signal Definitions

Address Bus 4 through 15

When the controller is jumpered for internal address decode, the presence of a 12-bit address on these lines which corresponds to the jumper-selected controller address causes the controller to be selected.

External Select

When the controller is jumpered for external address decode, a high level on this line causes the controller to be selected.

Read*, Write*

When the controller is selected, a low level on the Read* or Write* line causes the controller to transfer a byte of data to or from the data bus, respectively.

Data Bus 0 through 7

These eight bidirectional data lines are tri-stated (floating) except when the controller is selected and Read* or Write* is active.

Status/Data

A high level on this line causes the controller status port to be selected, and a low level causes the data port to be selected.

Reset Controller*

Grounding this line through a switch contact closure or heavy duty open collector gate causes the controller to reset.

Reset Complete

This line goes low when Reset Controller* is grounded or the controller reset button is depressed, and returns high approximately one second after the reset signal is removed.

Transmit Data**, Receive Data**, Data Terminal Ready**, Data Set Ready**, Request to Send**, Clear to Send**

These lines have their standard RS232 definitions.

2.3 DISKETTE DRIVE INTERFACE SPECIFICATIONS

2.3.1 Mating Connectors

The diskette drive interface is by means of an edge connector with 50 pins (dual 25) with .100" spacing (Scotchflex 3415-0000 or equivalent for ribbon cable, Viking Connector 3VH25/1JN-5 or TI Connector H312125 or equivalent for solder connections). All odd-numbered pins are connected to ground to facilitate the use of twisted-pair cable between the controller and diskette drive(s).

Pin No.	Signal Name
4	Drive Select 2-Right
10	Seek Complete
12	Restore
14	Remote Eject
18	Drive Select 2-Left
20	Index
22	Ready
26	Drive Select 1-Left
28	Drive Select 1-Right
34	Direction
36	Step
38	Write Data
40	Write Gate
42	Track 00
44	Write Protect
48	Separate Data
50	Separate Clock

2.3.2 Signal Definitions

For signal definitions, refer to PerSci Product Specifications, Model 70 or Model 277 Diskette Drive.

2.4 POWER REQUIREMENTS

Power requirements for the Model 1070 controller are:

+5 volts at 1.5 amp max
+12 volts at 150 ma max
-12 volts at 200 ma max

with all voltages regulated within ±5%.

2.5 HARDWARE OPTIONS

2.5.1 RS232 Serial Interface Option

This is a factory-installed option which provides an RS232 serial interface in addition to the standard parallel microcomputer interface. Only one of these interfaces may be used at a time; the parallel interface is enabled simply by removing the USART chip from its socket. The RS232 Serial Interface Option includes an on-board speed selection switch with the following settings:

Switch Setting	Transmission Speed (bps)
0	50
1	75
2	110
3	134.5
4	150
5	300
6	600
7	1,200
8	1,800
9	2,000
A	2,400
B	3,600
C	4,800
D	7,200
E	9,600
F	19,200

3.0 FIRMWARE SPECIFICATIONS

3.1 THEORY OF OPERATION

3.1.1 File Allocation

A diskette volume contains 77 tracks with 26 sectors per track and 128 data bytes per sector. The first track is reserved by the controller for use as an index (i.e., a table of contents) for the volume, while the remaining 76 tracks are available for file storage.

When a new file is created on a diskette volume, it receives an allocation of contiguous sectors. The minimum file allocation is one sector, and the maximum is 1,976 sectors (i.e., 76 tracks of 26 sectors, or 252,928 bytes). The first file created on a newly initialized diskette receives an allocation starting immediately above the index track. Subsequently created files receive an allocation starting immediately above the allocation of the previously created file. The allocation of each file is recorded on the index track.

Whenever a file is deleted, its block of contiguous sectors is deallocated. This leaves a gap in the sequence of allocated sectors on the diskette volume. The controller provides a command ("Gap") to compress the allocations on a volume, eliminating the gaps caused by previous file deletions.

3.1.2 File Access Methods

The controller provides four methods for accessing and updating data stored on diskette.

The stream access method permits an entire file to be read or written as a continuous stream of data bytes (as if the diskette file were a very high speed paper tape). Stream access is the simplest access method to use, requiring only a single controller command to save or load an entire file. It is ideally suited to the storage and retrieval of executable programs or any other use in which paper tape or cassette tape is conventionally used. Stream access is performed using the "Load" and "Save" controller commands.

The punctuated access method treats a file as a sequence of variable-length records separated by punctuation marks (the controller uses the ASCII record separator character "RS" for this). A punctuated file may be positioned at its beginning or end, and variable-length records may be

read or written in sequence, one at a time. Records may span sector boundaries on the diskette but this is made transparent by the controller. Punctuated access is most appropriate for the storage of text files (e.g., source programs, word processing files) or for any application in which sequential access to variable-length records is desirable. Because of its dependency upon a unique punctuation character ("RS") to separate records, punctuated access is not well suited to the storage of arbitrary binary information.

The relative access method treats a file as a byte-addressable memory. A relative file may be positioned at its beginning, end, or to any desired byte position within the file. Any number of bytes may then be read or written. Relative read and write operations may span sector boundaries but this is made transparent by the controller. Relative access is ideal for data base oriented applications in which random access is required. Both punctuated and relative access are performed using the "File", "Position", "Read", and "Write" controller commands.

Finally, the direct access method permits any specified sector of any specified track of a diskette to be read or written directly, bypassing the file management functions of the controller altogether. Direct access is performed using the "Input" and "Output" controller commands.

3.1.3 File References

A file reference identifies a particular file or group of files. File references may be either unique or ambiguous: a unique file reference identifies one file uniquely, while an ambiguous file reference may be satisfied by several different files.

File references consist of four component parts: a name of up to eight characters, a version of up to three characters, a type specified by a single character, and a drive which is a numeric digit between 0 and 3. The version, type, and drive components are optional, and are set off from the name by means of unique leading punctuation characters:

```
NNNNNNNN.VVV:T/D
```

Any component which is missing is assumed to be blank.

The following are examples of valid file references:

MONITOR	MASTER/2	STARTREK.BAS/1
MONITOR.SRC	MASTER:$	STARTREK.XQT
MONITOR.OBJ:A	MASTER.ONE	STARTREK:0/3

The special characters "?" and "*" may be used to make a file reference ambiguous so that it may match a number of different files. The "?" is used as a "wild-card" character which matches any character in the corresponding position in a file reference. Thus the ambiguous file reference:

PER????.BA?

matches all of the following unambiguous file references:

PERFECT.BAL	PERSCI.BAS	PERQ.BAX

The character "*" is used to denote that all character positions to the right are wild-cards. The following examples illustrate the flexibility which this facility provides:

MONITOR.*	=	MONITOR.???:?	matches all files with name "MONITOR"
*.BAS	=	????????.BAS:?	matches all files with version "BAS"
Z*	=	Z???????.???:?	matches all files starting with "Z"
*	=	????????.???:?	matches all files on the diskette

3.2 CONTROLLER COMMANDS

Controller commands consist of a single command letter followed (in most cases) by one or more command parameters. Parameters must not contain embedded spaces, must be set off from one another by spaces, and may optionally be set off from the command letter by spaces.

The various controller commands are summarized in Table 3-1 and described in detail in the following paragraphs.

Command	Command Syntax	Command Functional Description
Allocate	A file sectors	Allocates an empty file "file" of "sectors" sectors.
Copy	C file1 file2 sectors	Copies files matching "file1" to same or different diskette, optionally renaming according to "file2" and reallocating according to "sectors".
Delete	D file	Deletes files matching "file".
Eject	E /drive	Ejects diskette in drive "drive".
File	F unit file	Opens "file" and associates with "unit".
	F unit	Closes the open file associated with "unit".
	F	Closes all open files.
Gap	G /drive	Reallocates diskette in "drive" to eliminate gaps.
Input	I track sector /drive	Reads the specified sector.
Kill	K volume/drive seq	Initializes diskette with interleave "seq".
	K volume/drive	Deletes all files on diskette without initializing.
Load	L file	Reads entire file "file" as a stream.
Mode	M date:options/drive	Sets current date, I/O options, and/or default drive.
Name	N file1 file2	Renames file "file1" in accordance with "file2".
Output	O track sector /drive	Writes the specified sector.
Position	P unit sector byte	Positions the open file associated with "unit".
	P unit	Reports current position of file associated with "unit".
Query	Q file	Reports index information for files matching "file".
Read	R unit bytes	Relative read of file associated with "unit".
	R unit	Punctuated read of file associated with "unit".
Save	S file	Creates new file "file" by writing as a stream.
Test	T option/drive	Executes a diagnostic test on drive "drive".
Write	W unit bytes	Relative write to file associated with "unit".
	W unit	Punctuated write to file associated with "unit".

Table 3-1. Controller Command Summary

3.2.1 Mode Command

The "Mode" command may be used to set the current date, the default diskette drive, and/or various controller I/O options. The current date is entered as a six-character value; the format "YYMMDD" is suggested but not required by the controller. The default diskette drive is entered as the character "/" followed by a digit between 0 and 3; this becomes the drive which is used for all subsequent file references and commands which do not include an explicitly specified drive. The I/O options are entered as the character ":" followed by a digit between 0 and 7 according to the following table:

	Dual Drives	Single Drives
Parallel in/parallel out	0	4
Parallel in/serial out	1	5
Serial in/parallel out	2	6
Serial in/serial out	3	7

Examples:

```
M  770819
M  /1
M  :3
```

3.2.2 Save Command

The "Save" command creates a new file by writing a stream of data onto the diskette. The resulting file receives an allocation of the minimum number of sectors needed to accommodate the length of the stream.

Examples:

```
S  PROGFILE
S  PAYABLES/3
```

3.2.3 Load Command

The "Load" command reads a diskette file in its entirety as a stream.

Examples:

```
L  PROGFILE
L  PAYABLES/3
```

3.2.4 Name Command

The "Name" command modifies the name, version, and/or type of a file. The wild-card characters "?" and "*" are used to indicate that selected portions of the file reference are to be left unchanged, as illustrated in the examples.

Examples:

```
N  FRED   GEORGE
N  BACKUP.2   *.3
N  X-RATED   R*
```

The first example changes the file called "FRED" to one called "GEORGE". The second changes "BACKUP.2" to "BACKUP.3", while the last changes "X-RATED" to "R-RATED".

3.2.5 Delete Command

The "Delete" command deletes a file or a collection of files from a diskette.

Examples:

```
D  GEORGE
D  *.OBJ/1
D  XZ??
```

The first example deletes a single file, "GEORGE", from the default drive. The second example deletes all files on drive 1 which have type "OBJ". The last example deletes all files with 2 to 4 character names starting with "XZ".

3.2.6 Query Command

The "Query" command lists the following index information for one, some, or all files on a diskette:

- Name, version, and type
- Start of allocation (decimal track and sector)
- Length of allocation (decimal number of sectors)
- Position of the end-of-data mark (decimal sector and byte offset)
- Date of creation
- Date of last update

Examples:

```
Q  GEORGE/2
Q  *.SRC
Q  *
```

3.2.7 Kill Command

The "Kill" command deletes all files on a diskette volume. Optionally, the command also initializes (formats) the entire diskette, erasing all previously recorded information thereon and writing new sector headers on each track. The diskette may be initialized with any one of thirteen optional sector interleave sequences to enhance read/write performance (see also paragraph 3.4.3 below).

Examples:

```
K  SCRATCH/3
K  BACKUP.2  1
K  MASTER  9
```

The first example deletes all files on drive 3, labels the diskette volume "SCRATCH", but does not initialize each track. The second example initializes the diskette with no interleave, while the last example initializes with interleave 9.

3.2.8 Gap Command

The "Gap" command compresses the allocations on a diskette volume to eliminate any gaps in the allocations caused by prior file deletions.

Examples:

```
G  /3
G
```

3.2.9 Allocate Command

The "Allocate" command allocates a new, empty file of specified length (decimal number of sectors).

Example:

```
A  BIG-FILE  1000
```

3.2.10 Copy Command

The "Copy" command copies one or a collection of files from a diskette volume to the same or a different diskette volume. The copied files may have the same or different names as the original files, and may have the same or different allocations.

Examples:

```
C  FRED  GEORGE
C  FRED/0  */1
C  FRED/0  GEORGE/1  100
C  */0  */1
C  XX*/0  ZZ*/1
```

The first example makes a duplicate of the file "FRED" on the same diskette (default drive), calling the duplicate "GEORGE". The second example copies the file "FRED" from drive 0 to drive 1, leaving the name and allocation unchanged. The third example also copies "FRED" from drive 0 to drive 1, but changes the name to "GEORGE" and gives the new file an allocation of 100 sectors (which may be larger or smaller than "FRED" has). The fourth example copies all files from drive 0 to drive 1, preserving all file names and allocations. The last

example copies only files with names starting with "XX" from drive 0 to drive 1, changing the first two characters of each file name from "XX" to "ZZ".

3.2.11 File Command

The "File" command opens and closes diskette files. A file must be open before punctuated or relative access is permitted by the controller. An open file is associated with a logical unit number between 1 and 5; hence, a maximum of five files may be open simultaneously.

Examples:

```
F  2  MASTER/1
F  2
F
```

The first example opens the file "MASTER" on drive 1, and associates it with logical unit 2. The second example closes the open file associated with logical unit 2. The third example closes all open files.

3.2.12 Position Command

The "Position" command permits open files to be positioned at the beginning, the end, or at any specified byte position. The command may also report the current position of an open file.

Examples:

```
P  2  213  88
P  2  213
P  2  0
P  2  9999
P  2
```

The first example positions the open file associated with logical unit 2 to byte 88 in sector 213 of the file. The second example positions the file to byte 0 of sector 213. The third example positions the file to its beginning, and the fourth example positions the file to its end-of-data mark (note that the controller does not permit a file to be positioned beyond its end-of-data). Finally, the last example simply reports the current position of the file.

3.2.13 Read Command

The "Read" command reads an open file by means of either the relative or punctuated access method (i.e., fixed-length or variable-length records).

Examples:

```
R  2  80
R  2
```

The first example reads a fixed-length record of 80 bytes from the current position of the open file associated with logical unit 2. The second example reads a variable-length record delimited by a record separator character ("RS").

3.2.14 Write Command

The "Write" command writes an open file by means of either the relative or punctuated acces method (i.e., fixed-length or variable-length records). If data is written beyond the end-of-data mark of the file, the end-of-data mark is moved accordingly. The controller will not permit data to be written beyond the last sector allocated to the file.

Examples:

```
W  2  80
W  2
```

The first example writes a fixed-length record of 80 bytes to the open file associated with logical unit 2, starting at the current position of the file. The second example writes a variable-length record to the file, followed by a record separator character ("RS").

3.2.15 Input Command

The "Input" command reads a single specified sector of a diskette volume. The sector is specified by decimal track number (00-76), decimal sector number (01-26), and drive numbe

Examples:

```
I  43  10  /1
I  1  1
```

3.2.16 Output Command

The "Output" command writes a single specified sector of a diskette volume.

Examples:

```
O  43  10  /1
O  1   1
```

3.2.17 Eject Command

The "Eject" command causes the diskette to be ejected from the specified drive. Note that this command is effective only if the diskette drive is equipped with the Remote Eject feature.

Examples:

```
E  /2
E
```

3.2.18 Test Command

The "Text" command performs one of several diagnostic tests on the specified drive. The available tests are: V (random seek-verify test), R (random seek-read test), and I (incremental seek-read test).

Examples:

```
T  V/1
T  R/0
T  I
```

3.3 CONTROLLER INTERFACE PROTOCOL

3.3.1 Protocol Definition

The interface protocol between the microcomputer and the controller consists of sequences of ASCII characters and makes use of standard ASCII communications controls. The protocol for the simplest controller commands (Allocate, Eject, File, Kill, Mode, Name, Test) is the following:

Microcomputer transmits:	command-text CR LF EOT
Controller transmits:	ACK EOT

The protocol for controller commands which return informational text (Copy, Delete, Gap, Position, Query) is the following:

Microcomputer transmits:	command-text CR LF EOT
Controller transmits:	informational-text CR LF ACK EOT

The protocol for controller commands which read data from the diskette (Input, Load, Read) is the following:

Microcomputer transmits:	command-text CR LF EOT
Controller transmits:	SOH diskette-data ACK EOT

The protocol for controller commands which write data to the diskette (Output, Save, Write) is the following:

Microcomputer transmits:	command-text CR LF EOT
Controller transmits:	ENQ EOT
Microcomputer transmits:	diskette-data EOT
Controller transmits:	ACK EOT

Finally, the controller may terminate any command at any time with a fatal error diagnostic

message, using the following protocol:

Controller transmits: NAK fatal-error-message CR LF EOT

Note that no ACK will be transmitted by the controller in this case.

3.3.2 Error Diagnostic Messages

The controller issues two classes of error diagnostic messages: fatal and non-fatal. Fatal error diagnostic messages are always preceded by a NAK and followed by an EOT. They indicate the premature and unsuccessful termination of a controller command. The various fatal error diagnostic messages are listed below:

COMMAND ERROR - Indicates an invalid command or command parameter.

DUPLICATE FILE ERROR - Indicates an attempt to create a new file with the same name as an existing file on the same diskette.

NOT FOUND ERROR - Indicates that a file with the specified name could not be found in the index of the specified diskette.

OUT OF SPACE ERROR - Indicates an attempt was made to exceed the capacity of a diskette or of its index track.

READY ERROR - Indicates an attempt was made to access a diskette drive which is not ready.

UNIT ERROR - Indicates an attempt was made to read, write, or position a logical unit number with which no open file is associated.

HARD DISK ERROR - Indicates a seek, read, or write error which could not be successfully resolved in five retries.

Note that each fatal message begins with a unique letter, so that an interfacing program need only to analyze the first character following a NAK to determine the type of fatal error.

Non-fatal error diagnostic messages are issued for soft disk errors. They are not preceded by a NAK, and they contain the following information: (1) type of disk operation (seek, read, or write); (2) error retry number (1 to 5); (3) diskette location at which error occurred (decimal track and sector); and (4) type of error (protect, write fault, verify, CRC, or lost data). During the transmission of diskette data (Load, Save, Read, Write, Input, and Output commands), non-fatal error messages are suppressed.

3.3.3 Parallel Interface Considerations

The parallel interface offers a number of advantages in interfacing the controller to a microcomputer system: (1) its transfer rate is very fast, (2) it provides complete handshaking to coordinate data transfers in both directions, and (3) it provides a means for uniquely distinguishing communications control characters (EOT, ACK, NAK, SOH, ENQ) from data characters. The last two of these functions are accomplished by means of the controller status byte, whose format is:

bit 7 - input byte available, control character
bit 6 - input byte available, data character
bit 1 - output buffer full, data character
bit 0 - output buffer full, control character

When the microcomputer reads the controller data byte, bits 7 and 6 of the status byte are reset and remain so until the controller sends another character to the parallel interface. When the microcomputer writes the controller data or status byte, bit 1 or 0 (respectively) is set and remains so until the controller has received the character from the parallel interface. Since communications control characters cannot be confused with data characters, arbitrary binary data may be read or written freely when using the parallel interface.

3.3.4 RS232 Serial Interface Considerations

The optional RS232 serial interface provides no means either for coordinating data transfers through handshaking or for distinguishing between communications control and data characters. Thus, the user must take care not to transmit data to the controller faster than the controller can write it on diskette (this depends upon the type of operation, type of drive, sector interleave, etc.) Furthermore, the user must ensure that the significant communications control characters (EOT, ACK, NAK, SOH, ENQ) are not embedded in data sent to or from the controller; if arbitrary binary information is to be read or written, the user much provide a suitable escape convention.

3.3.5 Sample Driver Program

In order to provide additional guidance in the interfacing of the controller to a microcomputer system, flowcharts and an assembly listing of a sample driver program are provided at the end of this document. The sample driver program makes use of the parallel interface, and is coded for an 8080-based microcomputer system.

3.4 DISKETTE FORMAT

3.4.1 General Format

The diskette initialization function of the controller (Kill command) creates a diskette format which is IBM 3740 compatible. Each diskette contains 77 tracks with 26 sectors per track and 128 data bytes per sector. Tracks are numbered from 00 to 76 (outer to inner) and sectors are numbered from 01 to 26 on each track. Each sector has a header which defines the track and sector number (soft sectoring). Both the sector header and the data itself are provided with a 16-bit cyclic redundency check (CRC) word.

3.4.2 Index Track Format

Track 00 is reserved by the controller for use as a file index (table of contents) for the diskette. The controller makes use of an index track format which permits up to 100 files on each volume, and which is not IBM 3740 compatible (the IBM 3740 index track format allows only 19 files). Sector 01 of the index track serves as a volume label. Sectors 02 through 26 each contain room for four 32-byte file entries:

bytes 1-8	file name
bytes 9-11	version
byte 12	type
byte 13	(reserved)
bytes 14-15	start of allocation
bytes 16-17	end of allocation
bytes 18-19	end of data (sector)
byte 20	end of data (byte offset)
bytes 21-26	date of creation
bytes 27-32	date of last update

Support of an IBM 3740-compatible index track format is available as a firmware option.

3.4.3 Optional Interleaved Sector Sequences

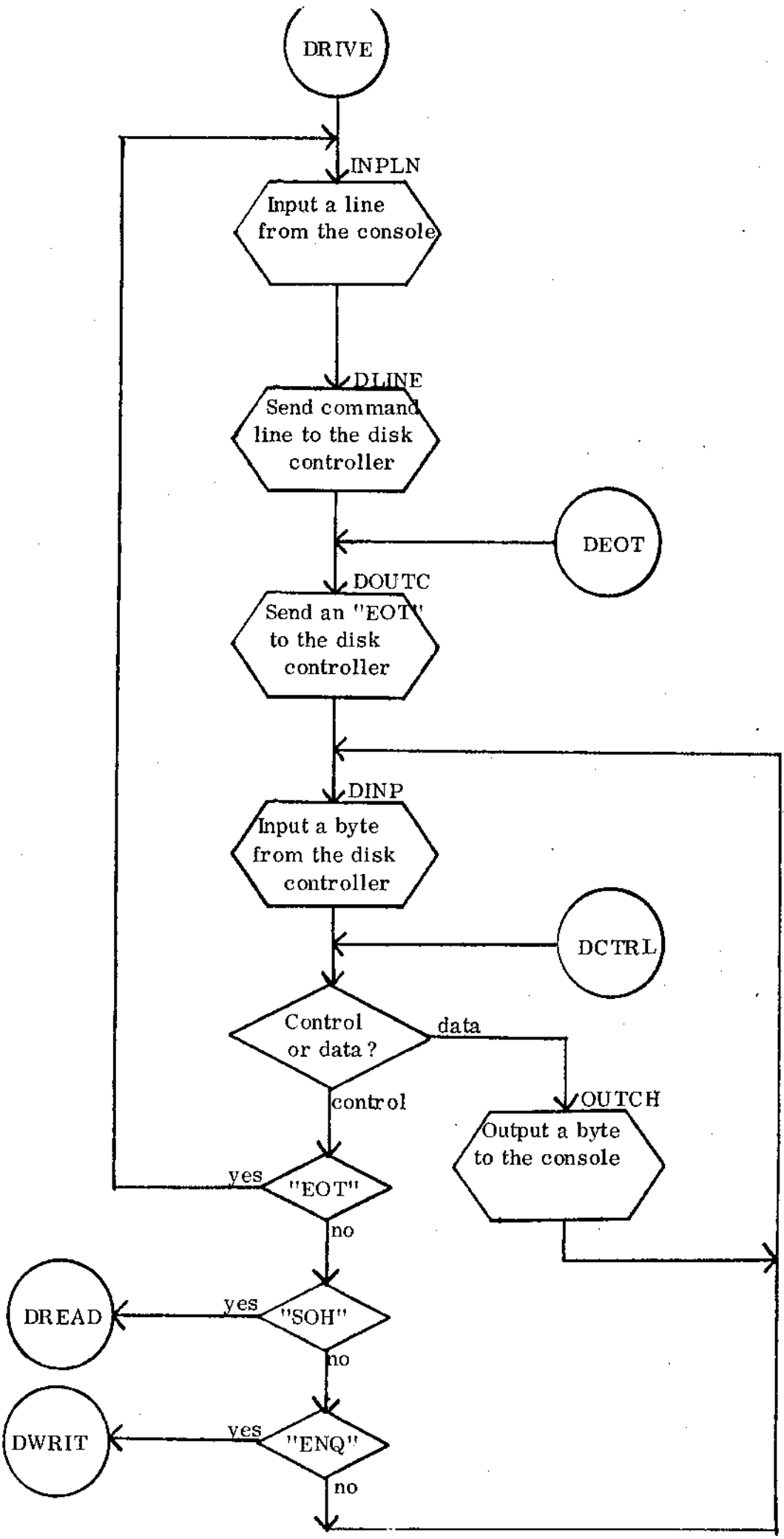
In order to enable the user to optimize diskette subsystem performance in a variety of situations, the diskette initialization function of the controller (Kill command) supports twelve optional interleaved sector sequences in addition to the ordinary non-interleaved sequence. This function is controlled by the value (1-13) of the optional second parameter of the Kill command. The effect of the interleaved sector sequences is to provide additional time to process the data for sector N before sector N+1 is encountered in the course of diskette rotation. Sequence 1 (non-interleaved) provides the shortest time interval between successively numbered sectors, sequence 2 provides the longest time interval, and sequence 3 through sequence 13 provide intermediate intervals. (Sequence 9 is optimal for disk performance when using the parallel interface in most environments.)

Additional information about these optional sector sequences and other diskette formatting considerations may be found in the following IBM document: The IBM Diskette for Standard Data Interchange, GA 21-9182-0, File No. GENL-03/80.

SAMPLE DRIVER PROGRAM

TO INTERFACE WITH

PERSCI MODEL 1070 CONTROLLER


DRIVE
INPLN
Input a line from the console
DLINE
Send command line to the disk controller
DEOT
DOUTC
Send an "EOT" to the disk controller
DINP
Input a byte from the disk controller
DCTRL
Control or data ?
data
control
OUTCH
Output a byte to the console
yes
"EOT"
no
DREAD
yes
"SOH"
no
DWRIT
yes
"ENQ"
no

```
0000                    0010 ****************************************
0000                    0020 * SAMPLE DRIVER PROGRAM TO INTERFACE WITH *
0000                    0030 *  PERSCI MODEL 1070 DISKETTE CONTROLLER  *
0000                    0040 ****************************************
0000                    0050 *
0000                    0060 * THIS PROGRAM OPERATES ON A 8080-BASED MICRO-
0000                    0070 * COMPUTER,  IT ASSUMES THAT THE PERSCI MODEL
0000                    0080 * 1070 DISKETTE CONTROLLER IS INTERFACED VIA
0000                    0090 * ITS PARALLEL PORT IN SUCH A MANNER THAT ITS
0000                    0100 * DATA AND STATUS BYTES APPEAR TO THE 8080 AS
0000                    0110 * MEMORY LOCATIONS C000 AND C001 HEX, RESPEC-
0000                    0120 * TIVELY.  IT ALSO ASSUMES THAT AN ASCII CON-
0000                    0130 * SOLE DEVICE (TELETYPEWRITER OR KEYBOARD/CRT)
0000                    0140 * IS CONNECTED TO THE MICROCOMPUTER.
0000                    0150 *
0000                    0160 *
0000                    0170 * THIS PROGRAM LISTING IS DIVIDED IN TWO SEC-
0000                    0180 * TIONS.  SECTION ONE CONTAINS THOSE ROUTINES
0000                    0190 * WHICH ARE UNIQUE TO THE DISKETTE CONTROLLER
0000                    0200 * INTERFACE.  IT REQUIRES ONLY 166 BYTES OF
0000                    0210 * PROGRAM STORAGE AND 2 BYTES OF RAM.
0000                    0220 *
0000                    0230 * SECTION TWO CONTAINS GENERAL I/O SUBROUTINES
0000                    0240 * WHICH ARE ROUTINELY A PART OF MOST MICRO-
0000                    0250 * COMPUTER OPERATING SYSTEMS OR MONITORS, AND
0000                    0260 * THUS WHICH WILL NOT NEED TO BE DUPLICATED IN
0000                    0270 * MOST INSTALLATIONS.
0000                    0280 *
0000                    0290 *
0000                    0300 * * * * * S E C T I O N   O N E * * * * * *
0000                    0310 *
0000                    0320 *
0000                    0330 * THIS IS THE BASIC DRIVER ROUTINE WHICH SENDS
0000                    0340 * CONSOLE COMMANDS TO THE CONTROLLER, CONTROLLER
0000                    0350 * MESSAGES TO THE CONSOLE, AND CONTROLS THE
0000                    0360 * TRANSMISSION OF FILES AND RECORDS BETWEEN THE
0000                    0370 * CONTROLLER AND MICROCOMPUTER RAM.
0000                    0380 *
0000 31 6C 01           0390 DRIVE  LXI   SP,STACK      INITIALIZE STACK
0003 CD A8 00           0400        CALL  INPLN         INPUT CONSOLE LINE
```

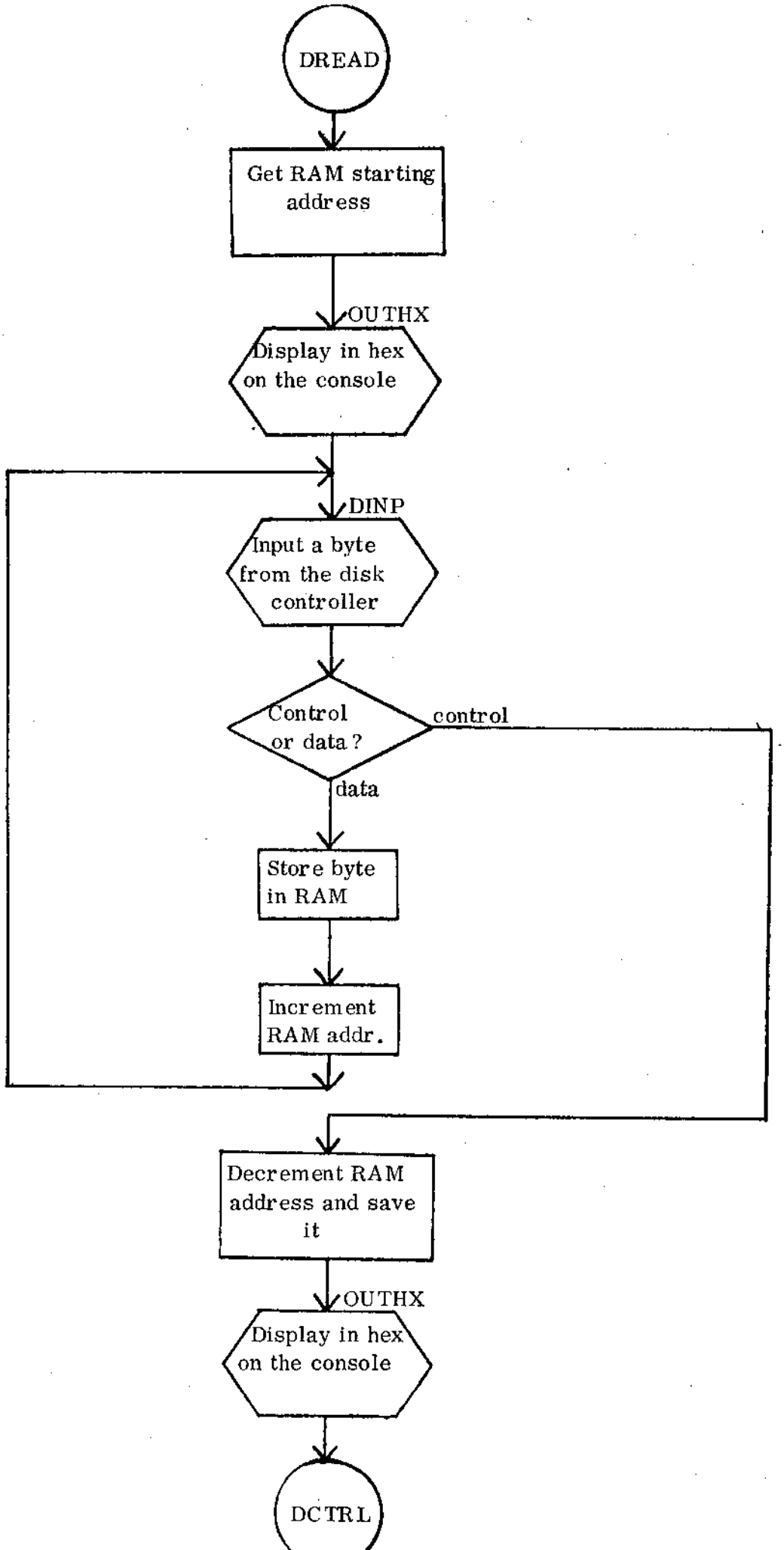
```
0006 CD 6B 00           0410          CALL  BLINE         SEND COMMAND TO DISK
0009 3E 04              0420 DEOT     MVI   A,EOT         SEND "EOT" TO DISK
000B CD 89 00           0430          CALL  BOUTC         AS CONTROL BYTE
000E CD 75 00           0440 DGET     CALL  DINP          INPUT BYTE FROM DISK
0011 DA 1A 00           0450          JC    DCTRL         CONTROL OR DATA BYTE?
0014 CD 06 E0           0460          CALL  OUTCH         DATA, SEND TO CONSOLE
0017 C3 0E 00           0470          JMP   DGET
001A FE 04              0480 DCTRL    CPI   EOT           CONTROL, WHAT KIND?
001C CA 00 00           0490          JZ    DRIVE         "EOT", COMMAND IS DONE
001F FE 01              0500          CPI   SOH
0021 CA 2C 00           0510          JZ    DREAD         "SOH", DO DISK READ
0024 FE 05              0520          CPI   ENQ
0026 CA 49 00           0530          JZ    DWRIT         "ENQ", DO DISK WRITE
0029 C3 0E 00           0540          JMP   DGET          ELSE IGNORE
002C                    0550 *
002C                    0560 *
002C                    0570 * THIS ROUTINE CONTROLS A DISK READ INTO RAM
002C                    0580 *
002C 2A 06 00           0590 DREAD    LHLD  RAM1          GET RAM STARTING ADDR
002F CD FE 00           0600          CALL  OUTHX         DISPLAY ON CONSOLE
0032 CD 75 00           0610 DREAL    CALL  DINP          INPUT BYTE FROM DISK
0035 DA 3B 00           0620          JC    DREAX         CONTROL OR DATA BYTE?
0038 77                 0630          MOV   M,A           DATA, MOVE TO RAM
0039 23                 0640          INX   H             INCREMENT RAM ADDR
003A C3 32 00           0650          JMP   DREAL         NEXT BYTE
003D F5                 0660 DREAX    PUSH  PSW           CONTROL, SAVE BYTE
003E 2B                 0670          DCX   H             DECREMENT RAM ADDR
003F 22 A7 00           0680          SHLD  RAM2          SAVE RAM ENDING ADDR
0042 CD FE 00           0690          CALL  OUTHX         DISPLAY ON CONSOLE
0045 F1                 0700          POP   PSW           GET CONTROL BYTE
0046 C3 1A 00           0710          JMP   DCTRL         GO ANALYZE IT
0049                    0720 *
0049                    0730 *
0049                    0740 * THIS ROUTINE CONTROLS A DISK WRITE FROM RAM
0049                    0750 *
0049 CD 75 00           0760 DWRIT    CALL  DINP          INPUT BYTE FROM DISK
004C D2 49 00           0770          JNC   DWRIT         SHOULD BE AN "EOT"
004F 2A A6 00           0780          LHLD  RAM1          GET RAM STARTING ADDR
0052 CD FE 00           0790          CALL  OUTHX         DISPLAY ON CONSOLE
0055 EB                 0800          XCHG
0056 2A A7 00           0810          LHLD  RAM2          GET RAM ENDING ADDR
0059 CD FE 00           0820          CALL  OUTHX         DISPLAY ON CONSOLE
```

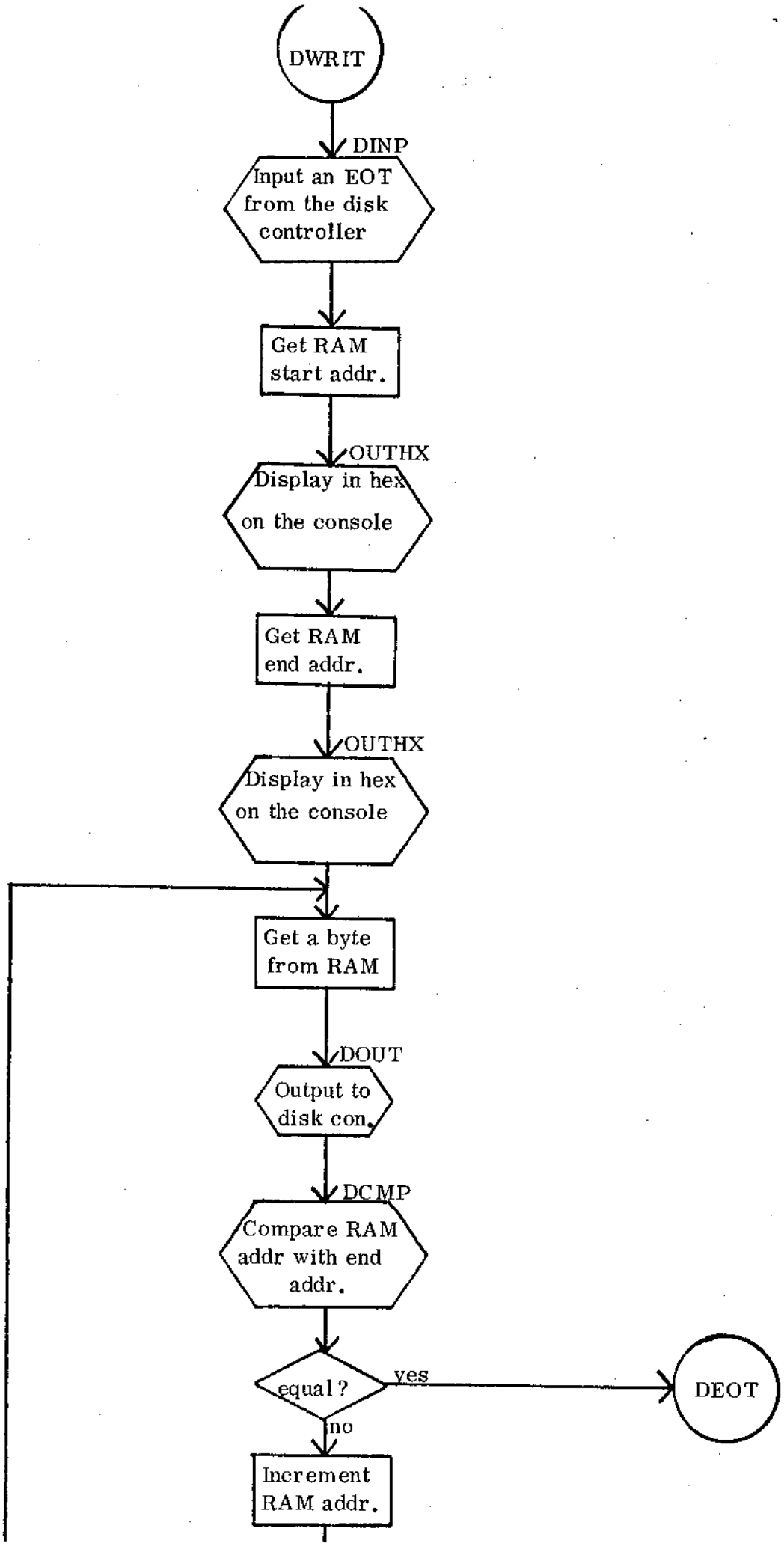
```
005C EB                  0830         XCHG                 START IN HL, END IN DE
005D 7E                  0840 DWRIL   MOV   A,M            GET BYTE FROM RAM
005E CD 82 00            0850         CALL  DOUT           SEND DATA TO DISK
0061 CD 35 01            0860         CALL  DCMP           COMPARE ADDR TO END
0064 D2 09 00            0870         JNC   DEOT           AT END, SEND "EOT"
0067 23                  0880         INX   H              ELSE INCREMENT RAM ADDR
0068 C3 5D 00            0890         JMP   DWRIL          PROCESS NEXT BYTE
006B                     0900 *
006B                     0910 *
006B                     0920 * THIS ROUTINE SENDS A LINE TO THE CONTROLLER
006B                     0930 *
006B CD 1F 01            0940 DLINE   CALL  GETCH          GET CHAR FROM BUFFER
006E D8                  0950         RC    EXHAUSTED,     ALL DONE
006F CD 82 00            0960         CALL  DOUT           SEND CHARACTER TO DISK
0072 C3 6B 00            0970         JMP   DLINE          PROCESS NEXT CHARACTER
0075                     0980 *
0075                     0990 *
0075                     1000 * THIS ROUTINE INPUTS A BYTE FROM THE CONTROLLER
0075                     1010 * AND SETS CARRY=1 IF A CONTROL BYTE
0075                     1020 *
0075 3A 01 C0            1030 DINP    LDA   DSTAT          GET DISK STATUS BYTE
0078 E6 C0               1040         ANI   0C0H           RECEIVE DATA AVAILABLE?
007A CA 75 00            1050         JZ    DINP           NO, WAIT UNTIL IT IS
007D 17                  1060         RAL                  SET CARRY IF CONTROL
007E 3A 00 C0            1070         LDA   DDATA          GET DISK DATA BYTE
0081 C9                  1080         RET                  ALL DONE
0082                     1090 *
0082                     1100 *
0082                     1110 * THIS ROUTINE SENDS A DATA BYTE TO THE CONTROLLER
0082                     1120 *
0082 CD 93 00            1130 DOUT    CALL  DOUTW          WAIT UNTIL READY
0085 32 00 C0            1140         STA   DDATA          WRITE DISK DATA BYTE
0088 C9                  1150         RET                  ALL DONE
0089                     1160 *
0089                     1170 *
0089                     1180 * THIS ROUTINE SENDS A CTRL BYTE TO THE CONTROLLER
0089                     1190 *
0089 CD 93 00            1200 DOUTC   CALL  DOUTW          WAIT UNTIL READY
008C 32 01 C0            1210         STA   DSTAT          WRITE DISK STATUS BYTE
008F 32 00 C0            1220         STA   DDATA          WRITE DISK DATA BYTE
0092 C9                  1230         RET                  ALL DONE
0093                     1240 *
```

```
0093                    1250 *
0093                    1260 * THIS ROUTINE WAITS FOR THE DISK TRANSMIT BUFFER
0093                    1270 * TO BE EMPTY AND READY FOR ANOTHER BYTE.  IT ALSO
0093                    1280 * ARBITRATES IF DISK AND HOST TRY TO TRANSMIT
0093                    1290 * TO ONE ANOTHER AT THE SAME TIME.
0093                    1300 *
0093 F5                 1310 DOUTW  PUSH  PSW           SAVE BYTE TO SEND
0094 3A 01 C0           1320        LDA   DSTAT         GET DISK STATUS BYTE
0097 E6 C0              1330        ANI   0C0H          IS DISK TRANSMITTING?
0099 CA A4 00           1340        JZ    DOUTX         YES, BREAK THE TIE
009C 3A 01 C0           1350        LDA   DSTAT         GET DISK STATUS AGAIN
009F E6 03              1360        ANI   03H           IS TRNSMT BUFFER EMPTY?
00A1 C2 94 00           1370        JNZ   DOUTW+1       NO, WAIT UNTIL IT IS
00A4 F1                 1380 DOUTX  POP   PSW           RESTORE BYTE TO SEND
00A5 C9                 1390        RET                 ALL DONE
00A6                    1400 *
00A6                    1410 *
00A6                    1420 * SYMBOLIC EQUIVALENCES
00A6                    1430 *
00A6                    1440 DDATA  EQU   0C000H        CONTROLLER DATA BYTE
00A6                    1450 DSTAT  EQU   0C001H        CONTROLLER STATUS BYTE
00A6                    1460 EOT    EQU   04H           ASCII "EOT"
00A6                    1470 SOH    EQU   01H           ASCII "SOH"
00A6                    1480 ENQ    EQU   05H           ASCII "ENQ"
00A6                    1490 *
00A6                    1500 *
00A6                    1510 * RAM WORKING STORAGE
00A6 00                 1520 RAM1   DB    0             RAM BUFFER START ADDR
00A7 00                 1530 RAM2   DB    0             RAM BUFFER END ADDR
00A8                    1540 *
00A8                    1550 *
```

```
00A8                    0010 ******SECTION  TWO******
00A8                    0020 *
00A8                    0030 *
00A8                    0040 * THIS ROUTINE INPUTS A LINE FROM THE CONSOLE
00A8                    0050 * INTO A RAM BUFFER, AND PROCESSES BACKSPACE
00A8                    0060 * AND LINE DELETE FUNCTIONS.
00A8                    0070 *
00A8 CD F3 00           0080 INPLN  CALL  CRLF          CR/LF TO CONSOLE
00AB 3E 3E              0090        MVI   A,'>'         GET COMMAND PROMPT
00AD CD 06 E0           0100        CALL  OUTCH         SEND TO CONSOLE
00B0 21 3B 01           0110        LXI   H,IBUFF       GET BUFFER ADDRESS
00B3 22 5B 01           0120        SHLD  IBUFF         INITIALIZE POINTER
00B6 0E 00              0130        MVI   C,0           INITIALIZE COUNT
00B8 CD 03 E0           0140 INPLI  CALL  INPCH         GET CHAR FROM CONSOLE
00BB FE 20              0150        CPI   ' '           TEST IF CONTROL CHAR
00BD DA D0 00           0160        JC    INPLC         YES, GO PROCESS
00C0 77                 0170        MOV   M,A           NO, PUT IN BUFFER
00C1 3E 20              0180        MVI   A,32          GET BUFFER SIZE
00C3 B9                 0190        CMP   C             TEST IF FULL
00C4 CA B8 00           0200        JZ    INPLI         YES, LOOP
00C7 7E                 0210        MOV   A,M           RECALL CHARACTER
00C8 23                 0220        INX   H             INCR POINTER
00C9 0C                 0230        INR   C             AND INCR COUNT
00CA CD 06 E0           0240 INPLE  CALL  OUTCH         ECHO CHARACTER
00CD C3 B8 00           0250        JMP   INPLI         GET NEXT CHAR
00D0 FE 08              0260 INPLC  CPI   08H           TEST IF BACKSPACE
00D2 CA E4 00           0270        JZ    INPLB         YES, KILL CHAR
00D5 FE 1B              0280        CPI   1BH           TEST IF ESCAPE
00D7 CA EE 00           0290        JZ    INPLK         YES, KILL LINE
00DA FE 0D              0300        CPI   0DH           TEST IF RETURN
00DC C2 B8 00           0310        JNZ   INPLI         NO, IGNORE CHAR
00DF 79                 0320        MOV   A,C           GET COUNT
00E0 32 5D 01           0330        STA   IBUFC         SAVE IT
00E3 C9                 0340        RET                 DONE
00E4 2B                 0350 INPLB  DCX   H             DECREMENT POINTER
00E5 0D                 0360        DCR   C             DECREMENT COUNT
00E6 F2 CA 00           0370        JP    INPLE         IF NOT NEG, GO ECHO
00E9 23                 0380        INX   H             IF NEG, UNDO DECR
00EA 0C                 0390        INR   C
00EB C3 B8 00           0400        JMP   INPLI         GET NEXT CHAR
00EE AF                 0410 INPLK  XRA   A             KILL BY SETTING
```

```
00EF 32 5D 01           0420        STA   IBUFC         COUNT TO ZERO
00F2 C9                 0430        RET                 DONE
00F3                    0440 *
00F3                    0450 *
00F3                    0460 * THIS ROUTINE SENDS A CR/LF TO CONSOLE.
00F3                    0470 *
00F3 3E 0D              0480 CRLF   MVI   A,0DH         GET A CR
00F5 CD 06 E0           0490        CALL  OUTCH         DISPLAY IT
00F8 3E 0A              0500        MVI   A,0AH         GET A LF
00FA CD 06 E0           0510        CALL  OUTCH         DISPLAY IT
00FD C9                 0520        RET   DONE
00FE                    0530 *
00FE                    0540 *
00FE                    0550 * THIS ROUTINE OUTPUTS THE CONTENTS OF REG H&L
00FE                    0560 * AS A FOUR-DIGIT HEXADECIMAL NUMBER
00FE                    0570 *
00FE 3E 20              0580 OUTHX  MVI   A,' '         GET A SPACE
0100 CD 06 E0           0590        CALL  OUTCH         SEND TO CONSOLE
0103 7C                 0600        MOV   A,H           GET TOP HALF OF WORD
0104 CD 08 01           0610        CALL  OUTH1         DISPLAY IN HEX
0107 7D                 0620        MOV   A,L           SAME WITH BOTTOM HALF
0108 F5                 0630 OUTH1  PUSH  PSW           SAVE LOW-ORDER DIG
0109 1F                 0640        RAR                 GET HIGH-ORDER DIG
010A 1F                 0650        RAR
010B 1F                 0660        RAR
010C 1F                 0670        RAR
010D CD 11 01           0680        CALL  OUTH          DISPLAY HEX DIGIT
0110 F1                 0690        POP   PSW           GET OTHER DIGIT
0111 E6 0F              0700 OUTH   ANI   0FH           EXTRACT DIGIT
0113 C6 30              0710        ADI   '0'           ADD ASCII ZONE BITS
0115 FE 3A              0720        CPI   '9'+1         TEST IF A-F
0117 DA 06 E0           0730        JC    OUTCH         NO; OUTPUT IT
011A C6 07              0740        ADI   'A'-'9'-1     YES; ADD BIAS FOR A-F
011C C3 06 E0           0750        JMP   OUTCH         OUTPUT IT
011F                    0760 *
011F                    0770 *
011F                    0780 * THIS ROUTINE OBTAINS A CHARACTER FROM THE RAM
011F                    0790 * BUFFER, AND SETS CARRY=1 IF EXHAUSTED.
011F                    0800 *
011F E5                 0810 GETCH  PUSH  H             SAVE REGS
0120 2A 5B 01           0820        LHLD  IBUFP         GET POINTER
0123 3A 5D 01           0830        LDA   IBUFC         GET COUNT
```


DREAD
Get RAM starting address
OUTHX
Display in hex on the console
DINP
Input a byte from the disk controller
Control or data ?
control
data
Store byte in RAM
Increment RAM addr.
Decrement RAM address and save it
OUTHX
Display in hex on the console
DCTRL


DWRIT
DINP
Input an EOT from the disk controller
Get RAM start addr.
OUTHX
Display in hex on the console
Get RAM end addr.
OUTHX
Display in hex on the console
Get a byte from RAM
DOUT
Output to disk con.
DCMP
Compare RAM addr with end addr.
equal?
yes
DEOT
no
Increment RAM addr.

```
0126 D6 01              0840          SUI   1             DECREMENT WITH CARRY
0128 DA 33 01           0850          JC    GETCX         NO MORE CHARACTERS
012B 32 5D 01           0860          STA   IBUFC         REPLACE COUNT
012E 7E                 0870          MOV   A,M           GET CHARACTER
012F 23                 0880          INX   H             INCR POINTER
0130 22 5B 01           0890          SHLD  IBUFP         REPLACE POINTER
0133 E1                 0900 GETCX    POP   H             RESTORE REGS
0134 C9                 0910          RET                 DONE (CARRY IF NO CHAR)
0135                    0920 *
0135                    0930 *
0135                    0940 * THIS ROUTINE COMPARES DE WITH HL.
0135                    0950 *
0135 7C                 0960 DCMP     MOV   A,H           GET MOST SIGNIF
0136 BA                 0970          CMP   D             COMPARE MOST SIGNIF
0137 C0                 0980          RNZ                 NONZERO, DONE
0138 7D                 0990          MOV   A,L           GET LEAST SIGNIF
0139 BB                 1000          CMP   E             COMPARE LEAST SIGNIF
013A C9                 1010          RET                 DONE
013B                    1020 *
013B                    1030 *
013B                    1040 * THESE ROUTINES PERFORM INPUT AND OUTPUT FROM
013B                    1050 * AND TO THE CONSOLE, PASSING ONE CHARACTER IN
013B                    1060 * THE A-REGISTER.  THEY MUST BE CODED TO WORK
013B                    1070 * WITH THE PARTICULAR CONSOLE I/O INTERFACE AR-
013B                    1080 * RANGEMENT OF EACH MICROCOMPUTER.
013B                    1090 *
013B                    1100 INPCH    EQU   0E003H        CONSOLE INPUT ROUTINE
013B                    1110 OUTCH    EQU   0E006H        CONSOLE OUTPUT ROUTINE
013B                    1120 *
013B                    1130 *
013B                    1140 * RAM WORKING STORAGE
013B                    1150 *
013B                    1160 IBUFF    DS    32            INPUT TEXT BUFFER
015B                    1170 IBUFP    DS    2             INPUT POINTER
015D                    1180 IBUFC    DS    1             INPUT COUNTER
015E                    1190          DS    16            STACK AREA
016E                    1200 STACK    EQU   $             TOP OF STACK
016E                    1210 *
016E                    1220 *
```

SYMBOL TABLE

CRLF	00F3	DCMP	0135	DCTRL	001A	DDATA	C000	DEOT	0009	DGET	000E
DINP	0075	DLINE	006B	DOUT	0082	DOUTC	0089	DOUTW	0093	DOUTX	00A4
DREAD	002C	DREAL	0032	DREAX	003D	DRIVE	0000	DSTAT	C001	DWRIL	005D
DWRIT	0049	ENQ	0005	EOT	0004	GETCH	011F	GETCX	0133	IBUFC	015D
IBUFF	013B	IBUFP	015B	INPCH	E003	INPLB	00E4	INPLC	00D0	INPLE	00CA
INPLI	00B9	INPLK	00EE	INPLN	00A8	OUTCH	E006	OUTH	0111	OUTH1	0108
OUTHX	00FE	RAM1	00A6	RAM2	00A7	SOH	0001	STACK	016E		

PERSCI, INC.

12210 Nebraska Avenue
West Los Angeles
California 90025

June 1977