OUP/M --- A 6502 OPERATING SYSTEM

A Thesis presented to

The Graduate Faculty of

The College of Arts and Sciences of Ohio University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Jian - Xiong Shao

November, 1983

OUP/M -- A 6502 OPERATING SYSTEM

BY

Jian - Xiong Shao

This thesis has been approved

for the Department of Mathematics -- Computer Science Option

and the College of Arts and Sciences by

Hans E. Eldridge

Associate Professor of Computer Science

Wm F. Doyle

Dean, College of Arts and Sciences

Acknowledgement

The author wants to express his deeply appreciate to Dr Klaus E. Eldridge, Chairman of Computer Science Department of Ohio University, from whom most idea and suggestion were got. He also read over all the materials presented in this thesis and corrects all the mistakes.

Thanks also should be given to Mr. Liu Qi-Wen for his writing the CCP part of the OUP/M operating system program which is presented in Appendix B.

Table of contents

List of Tables

List of figures

OUP/M -- A 6502 OPERATING SYSTEM

Chapter 1 Introduction

CP/M is a monitor control program for microcomputer system development which uses flexible disks for back-up storage. It is a standard operating system widely used on 8085-, Z80- and 8080- based machines. An adaptation, CP/M-86, is being used with the newer 16-bit systems based on the 8086 and 8088 microprocessors (for example, it runs on the IBM personal computer), while CP/M-68 is being developed for the 68000 microprocessor.

The CP/M monitor provides rapid access to programs through a comprehensive file management package. The file subsystem supports a named file stucture, allowing dynamic allocation of the file space as well as sequential and random file access. Using this file system, a large number of programs can be easily stored to or loaded from the disk interactively or under program control.

CP/M provides a collection of routines to manage many kinds of I/O devices. These routines can be easily modified to transport the CP/M to any 8080-, Z80- and 8085-based microcomputers.

CP/M also supports a set of useful and powerful utilities such as editor, assembler, debugger and several disk housekeeping programs as

well as various high level language compilers. These facilities make the whole system versatile and give microcomputers capabilities found on minicomputers

Although CP/M has been accepted as a standard operating system for microcomputers and has become very popular, it can not be run on the 6502-based microcomputer system. Further, up to this point there has not been much interest in adapting CP/M for the 6502-based machines. Manufacturers, such as Apple, Commodore, and Radio Shack have developed their own unique operating systems for their 6502-based machines. That is, there has been no concentrated effort to develop a common 6502 based operating system.

The Computer Science Department at Ohio University owns a collection of Ohio Scientific Instruments (OSI) microcomputers which are 6502-based. They are supported by a very primitive operating system which lacks of many of the conveniences found in the CP/M and is also incompatible with the operating systems found on Apple II, Commodore PET's, Commodore 64's and Radio Shack color computers. A need exists for an operating system which could be shared by all 6502 based microcomputers.

The problem addressed in this research project is the development of a portable 6502 based operating system which has all the desirable features of CP/M. That is, CP/M is used as a model in the developing because it has been so widely accepted by the microcomputer users. One

of the appendices of this report is a user's manual for the present version of such a system -- OUP/M, and another contains the complete source listing.

Developing such an operating system is a large undertaking. In order to make it more managable the project has been broken down into several phases. Phase one, the subject of this report, is the development of OUP/M for a specific computer configuration -- OSI. Future phases are to cover the development of assemblers, loaders, editors, etc, -- a host of necessary utilities -- and the addition of a porting system.

In the development of OUP/M, an OSI C-3 microcomputer system was used. This system consists of a 6502 CPU, a dual 8-inch floppy disk drive, a console, a printer, and 56 K bytes of RAM. On this machine all I/O is memory mapped and address space layout has to be taken into consideration.

The general memory layout for the OSI is as follows, with addresses given as hexadecimal values:

$0000 -- $BFFF	48K bytes of static RAM
$C000 -- $CFFF	I/O ports and contollers
$D000 -- $EFFF	8K bytes of static RAM
$F000 -- $FFFF	I/O ports and controllers, disk boot strap and CPU vectors

The printer and console are industry standard serial devices controlled via 6850 ACIAs (asynchronous communication interface adapters). Disk control is through a non-standard interface. A 6820 PIA (peripheral interface adapter) is used to control the head movement and loading/unloading, while a 6850 ACIA is used for the data read/write functions. Exact usage of these controllers will be explained later.

Knowledge of standard CP/M was gained through careful study of existing documentation followed by disassembling the object code into Z-80 nmemonics and then doing an extensive walk-through. This was then followed by extensive experiments to determine disk/track capacity using the OSI disk controller.

After this a design for the file system was worked out and coding and testing began. Mr. Liu Qi-Wen developed the CCP while the author did the work on the BDOS, BIOS and installation. Note CCP, BDOS and BIOS are CP/M stuctures to be explained later.

Most of the development was done between Jan. and Aug. 1982 while Jun. to Aug. 1983 was devoted to the writing of the OUP/M manual.

In the following sections additional details will be presented, including a brief discussion of the architecture for CP/M and the differences between the CP/M and the OUP/M.

At the end of phase one, the 'kernel' for OUP/M has been developed, tested and installed along with a disk utility program -- DSKUTY. This program permits formatting of diskettes and copying of all or parts of a diskette.

CHAPTER 2 Details

2.1 CP/M overview

CP/M is logically divided into three parts:

BIOS Basic Input/Output system.

BDOS Basic Disk Operating system.

CCP Console Command Processor.

The BIOS provides the primitive operations necessary to access the disk drives and to interface standard peripherals (teletype, CRT, reader and punch) and user-defined peripherals, and can be tailored by the user for any particular hardware environment by 'patching'.

The BDOS provides disk management by controlling one or more disk drives containing independent file directories. The BDOS implements disk allocation strategies which provide fully dynamic file construction while minimizing head movement across the disk during access. There are 37 system functions available to the assembly language user. Among them are the following commonly used:

Search look for a particular disk file by name.

Open open a file for further operations.

Close close a file after processing.

Rename change the name of a particular file.

Select select a particular disk drive for further operations.

The CCP provides symbolic interface between the user's console and the remainder of the CP/M system. The CCP reads the console device and processes commands typed in by the user, which include listing the file directory, typing the content of files and controlling the operation of transient programs, such as assembler, editor and debugger. There are 6 built-in commands within the CCP listed below:

ERA erase specified files.

DIR list file names in the directory.

REN rename the specified file.

SAVE save memory contents of a file onto the disk.

USER change the user code.

2.2 Work environment

The work environment, which the OUP/M operating system was written for and was developed on, includes a 6502-based microcomputer, a dual disk drive which can hold two single sided, single density and soft sectored 8-inch floppy disks, a terminal and a printer. The hardware configuration of the microcomputer is as follows:

1. The 6502 CPU : is a central processor unit, which supports a set of 6502 instuctions, an 8-bit data bus and a 16-bit address bus.

2. memory configuration : The memory consists of 56K RAM and 8K I/O interfaces. The assignment of the memory is listed as follows:

$0000 - $00FF (RAM)	:	is page zero.
$0100 - $01FF (RAM)	:	is a hardware stack set aside for the 6502.
$0200 - $BFFF (RAM)	:	is free for programs.
$C000 - $CFFF	:	is reserved for various I/O interfaces.
$D000 - $EFFF (RAM)	:	is free for the programs.
$FE00 - $FFF9 (ROM)	:	is a resident monitor and disk bootstrap.
$FFFA - $FFFF (ROM)	:	are non-maskable interrupt vector, reset vector, and IRQ (or BRK) interrupt vector respectively.

3. I/O interface configuration : Memory locations $C000 through $CFFF are reserved for various I/O interfaces for both the system and the user to use. Only few of them are used now. The rest are reserved for future expansion. Listed below are all installed I/O interfaces along with their usage.

$C000 - $C003 : is a 6820 peripheral interface adapter (PIA)

used as disk controller. (See below for details)

$C010 - $C011 : is a 6850 asynchronous communication interface adapter (ACIA) used as disk I/O port to transfer the data between the disk and memory. It must be initialized(control byte $58) to perform asynchronous transmision as follows:

a. Baud rate is set to clock rate divided by one.

b. Character bit number is set to 8.

c. Parity is set to even.

d. Stop bit number is set to 1.

e. interrupt is disabled.

$CF02 - $CF03 : is an asyncronous communication interface adapter (ACIA) used as printer I/O port to send the data from memory to the printer. It's initialization is the same as disk I/O port mentioned above except that the baud rate is set to clock rate divided by 16.

$FC00 - $FC01 : is an asyncronous communication interface adapter (ACIA) used as console I/O port to transfer the data between the console and memory. It's initialization is the same as the printer I/O port mentioned above.

$FC00 - $FC07 : is a " soft switch" for selecting an alternate processor(6800, Z80)..

4. Disk controller : is a hardware configuration which will control the drive action, such as drive head loading and unloading, drive head stepping forward and backward, data writing or erasing enable and disable, etc. Normally CP/M runs on machines which use a WD-1771 chip as their disk controller. However the OSI machine does not possess this chip. Instead, it uses a 6820 PIA as its disk controller. The port A of it is used as input control (except bit 6 is used as output), where the register addressed by $C000 is the data buffer and the register addressed by $C001 is its corresponding control register. The port B of the PIA is used as output control (including bit 6 of port A), where the register addressed by $C002 is the data buffer and the register addressed by $C003 is its corresponding control register. Listed below is the correspondence between the bits in the data buffers and their various uses.

PORT A

Bit No.	Usage
0	drive A is ready (low active, input)
1	track 0 is concerned (low active, input)
2	fault is found (low active, input)
3	not used.
4	drive B is ready (low active, input)
5	disk is write protected (low active, input)
6	selects drive B (low active, output)
7	index hole is found (low active, input)

Port B

Bit No.	usage
0	enables writing (low active, output)
1	enables erasing (low active, output)
2	indicates drive head stepping direction (0 for stepping toward track No. 76 1 for stepping toward track No. 0)
3	steps the drive head (transition from 1 to 0)
4	resets fault (low active)
5	selects drive A (high active, output)
6	current adjust (output) (0 for track No. 43 - 76 1 for track No. 0 - 42)
7	loads the head (low active, output)

2.3 Memory Layout

Under the OUP/M, memory is divided into several parts as follows: (see Figure 2.2 in Appendix A)

$0000 - $007F : is used by the system, the user should not use it. See Chapter 6 in appendix A for more details.

$0080 - $00FF : is available to the user.

$0100 - $01FF : is reserved for the hardware stack.

$0200 - $AFFF : is the user area where system utility and user programs run. When the user wants to run a program, the system loads it into this area with $0200 as the start of execution address.

$B000 - $BFFF : is disk I/O buffer. Unlike CP/M, we have no physical sector header information on the track. When the system wants to read a record from a certain track, it reads the whole track into this buffer and moves the data between it and the default I/O buffer (or user defined buffer).

$C000 - $CFFF : is peripheral interface area. This area is for peripheral interfaces such as ACIA, PIA and some other interfaces. We currently only use:

$C000 - $C003 -- Disk control ports.

$C010 - $C011 -- Disk I/O port.

$CF02 - $CF03 -- Line printer I/O port.

$D000 - $EFFF : is for operating system. See section 2.2.2 in Appendix A for details.

$FC00 - $FC01 : is a console I/O port.

$F000 - $FFFF : is a resident monitor and disk bootstrap. There are some exceptions mentioned below:

$FFFA - $FFFB Non-remaskable interrupt vector,

$FFFC - $FFFD Reset vector,

$FFFE - $FFFF Interrupt or break vector

are 6502 vectors which point to $01C0, $FFA0, $0130 respectively.

2.4 Disk layout (see figure 2.4 in appendix A)

Track 0 contains cold loader, BIOS and part of BDOS.

Track 1 contains CCP and rest of BDOS.

Track 2 is reserved for further use.

Track 3 contains the directory of each file in the disk.

Track 4 - 76 will store the system utilities and user programs.

There are 7 blocks in each track. Block size in this system is fixed as 512 bytes. From track No. 4 through track No. 76 we number each block from 0 through 510 with each corresponding to a bit in the bit map indicating it's allocation status when this disk is logged in. Each block contains four consecutive sectors. No logical sector number and no physical sector interleaving are needed. See Section 2.3.2 and 3.1 in Appendix A for more details about disk layout.

2.5 OUP/M CCP and system functions.

OUP/M CCP is a console command processor which acts as an interface between the user and the OUP/M operating system. There are six standard commands included in the OUP/M CCP as it's six routines, which form the main part of the CCP itself. There is also a routine which controls the operations of transient programs such as assembler, editor, debugger and ect.

When designing the OUP/M CCP, we fully mimic the CP/M CCP. That is, we included all the features found in the CP/M CCP. See chapter 4 in Appendix A for more details.

There are 35 system functions in the OUP/M available to the assembly programmers. The function names along with their numbers are listed below:

0	System reset	19	Select disk
1	Console input	20	Create file
2	Console output	21	Delete file
3	List output	22	Open file
4	Direct console output	23	Close file
5	Buffer output	24	Read record sequentially
6	Read console buffer	25	Write record sequentially
7	Get I/O byte	26	Read record randomly
8	Set I/O byte	27	Write record randomly

9	Get input console status	28	Search for first
10	Reset disk system	29	Search for next
11	Get log-in vector	30	Rename file
12	Get current disk number	31	Set file attributs
13	Get map address	32	Compute file size
14	Set write potection	33	Set random record
15	Get read only vector	34	Sent I/O buffer to disk
16	Get parammeter table adr.		
17	Set/get usercode		
18	Set DMA address		

One noticable difference in numbering and organization of the system functions between OUP/M and CP/M should be mentioned here. All the functions mentioned above are numbered differently and grouped by peripheral device I/O, disk I/O and other user functions. A translation table betweem OUP/M system number and CP/M system number is shown below:

OUP/M	CP/M	OUP/M	CP/M
0	0	18	26
1	1	19	14
2	2	20	22
3	5	21	19
4	6	22	15
5	9	23	16
6	10	24	20

7	7	25	21
8	8	26	33
9	11	27	34
10	13	28	17
11	24	29	18
12	25	30	23
13	27	31	30
14	28	32	35
15	29	33	36
16	31	34	--
17	32		

2.6 Page zero usage

Page zero plays an important role in 6502 assembly language programming. The system and the user both want to use it when an assembly program is running under the system. In this case, a competition may happen. There are two ways to solve this problem, swopping and sharing.

Swopping allows both the system and the user use the full page zero. The system initially saves the system page zero in a reserved area, freeing page zero for an assembly program. Upon each function call, the system page zero is restored, while the user page zero is saved. After the function call, this process is reversed. Finally, as the program terminates, the system page zero is restored. But this

solution has its disadvantage. It will require more execution time if frequent system calls are involved.

Another way of solving this problem is sharing. This is the method adopted in OUP/M. Page zero locations $00 through $7F are reserved for the system, while locations $80 through $FF are available to the user programs. It is user's responsibility to take care of swopping if the full page zero has to be used, since there is no built-in function for this nor is there any free memory within the OUP/M operating system. The user must allocate one page of memory in the transient area and code a swopping routine within the program which needs all of page zero.

2.7 Track format and track R/W algorithum

The track format is presented in Chapter 2 of Appendix A. Sector headers and sector gaps have been eliminated. As a consequence of this, we get two benefits: increased track size and simpler, speeder disk I/O.

1. Increasing the track size

The traditional IBM 3740 track format (See Figure 1) is based on track headers and sector headers. A repeating pattern of sector header, gap, sector data record and gap around the disk is needed. The gap after each sector header gives the disk controller time to recover

IBM 3740 TRACK FORMAT

index hole	gap #1 : 32 bytes	sector No.1	.....	sector No.26	gap #4 : 320 bytes

Where sector format is

sector header	gap #2 : 17 bytes	data record area	gap #3 : 33 bytes

a. The sector header is composed of 7 bytes as follows:

Byte 1 contains sector head identification.

Byte 2 contains track No.

Byte 3 contains $00.

Byte 4 contains sector No.

Byte 5 contains $00

Byte 6 -- byte 7 contain CRC bytes

b. Data record area is composed of 131 bytes as follows:

Byte 1 contains the starting flag of the data record area.

Byte 2 -- 129 contain 128 byte data record.

Byte 130 -- 131 contain CRC bytes.

Figure 1

IBM 3740 track format

so that it can read or write the data record. The gap after the data record is required since a write to disk may be slightly longer or shorter than last time, due to slight speed variations in disk rotation, temperature changes or tolerances in disk drives. Consequently some disk space must be used for building up sector headers and sector gaps. Thus a single sided, single density, soft sectored, eight inch floppy disk can usually hold at most 13 pages of data -- 26 sectors of 128 data bytes per track.

The track format in OUP/M eliminates the sector headers and sector gaps, making 14 pages of data available in each track. Thus we have total 77 * 14 * 256 = 275,968 bytes on each disk which is 19K more than under the IBM 3740 format.

2. Simplifing and speeding up the disk I/O operation

The CP/M read/write disk algorithm is based on the IBM 3740 track format. When it tries to read (write) a file, it reads (writes) one sector at a time. In order to speed up these operations, an interleaving read/write algorithm is used by CP/M. A file is organized into logical sector order rather than in physical sectors. The correspondence between the logical sector and physical sector for a particular disk drive is given in the translation vector table located somewhere in the BIOS and addressed by the first word of its disk parameter header (see 2.9.1 for details). The following example shows the correspondence of the logical sectors and the physical ones for a

disk with a sector skew factor of six in CP/M:

In each track, the logical sector is arranged in term of physical sector number as follows:

1, 7 , 13, 19, 25, 5, 11, 17, 23, 3, 9, 15, 21, 2, 8, 14, 20, 26, 6, 12, 18, 24, 4, 10, 16, 22.

The correspondence between physical sector No. (within the track) and its logical sector No. for the whole disk is shown in Table 1.

A file arranged in logical sector order would save some R/W time. The reason is given by the following example. We first assume that a file is arranged in physical sector order and consecutively stored in a track. Between two consecutive sector reading (writing), there needs to be some time to handle the data. Thus the next sector can not be read (written), until the next disk revolution. Consequently, 26 revolutions are needed to read (write) all 26 sectors on a single track.

Next assume that the file is arranged in logical sectors corresponding to interleaved physical sectors with a skew factor of six, the time it takes the disk to rotate over six intervening physical sectors allows the process a chance to do something with the current sector's data. Thus, four or five consecutive logical sectors may be accessed within one revolution of the disk. Nevertheless, it requires six revolutions to read (or write) all logical sectors from one track.

Track No.	Physical sector No.	Logical sector No.
0	1	0
0	7	1
0	13	2
•	•	•
•	•	•
•	•	•
0	22	25
1	1	26
1	7	27
1	13	28
•	•	•
•	•	•
•	•	•
76	22	2001

Table 1
Correspondence between physical sector No. and logical sector No.

The OUP/M disk R/W algorithm is based on the new track format and the following considerations:

1) There is a disk I/O buffer of 3.5 K bytes in memory. When a record needs to be read, the whole track on which the record is stored is read to this buffer. The blocking / deblocking routine in the BIOS will then take care of getting the 128-byte record requested by the BDOS and putting into the location specified by the BDOS. Note in the case of writing, the blocking / deblocking routine will first block the record to be written from the location given by the BDOS into the above buffer and then the modified track in the buffer is written onto the disk.

2) A track indicator which records the number of the current track in the disk I/O buffer is used. When BDOS passes the desired track No. to the BIOS, a comparision is made to see whether the desired track is already in the buffer. If it is in the buffer, no disk I/O is needed and the data will flow back and forth between the disk I/O buffer and the buffer used by the BDOS. Otherwise the disk I/O operation occurs (see below for details) and the data then flow back and forth in the disk I/O buffer. At the end, the indicator is set to the new track No.

3) A flush flag, indicating whether the current track in the disk I/O buffer has been written to, is also used. When a track boundary is crossed and an I/O operation is needed, a check of this flag will yield one of the following situations:

A. If the flag is set, we need to write the current disk I/O buffer back onto the track indicated by the track indicator and then read the desired track into the disk I/O buffer. The

flush flag is then reset to zero.

B. If the flag is not set, we just simply read the desired track into the disk I/O buffer.

Based on the above considerations, the new disk read algorithum is given as follows:

entry parameters:
1)desired track No.
2)desisred relative sector No. within the track.
3)the address of the buffer required by the BDOS.

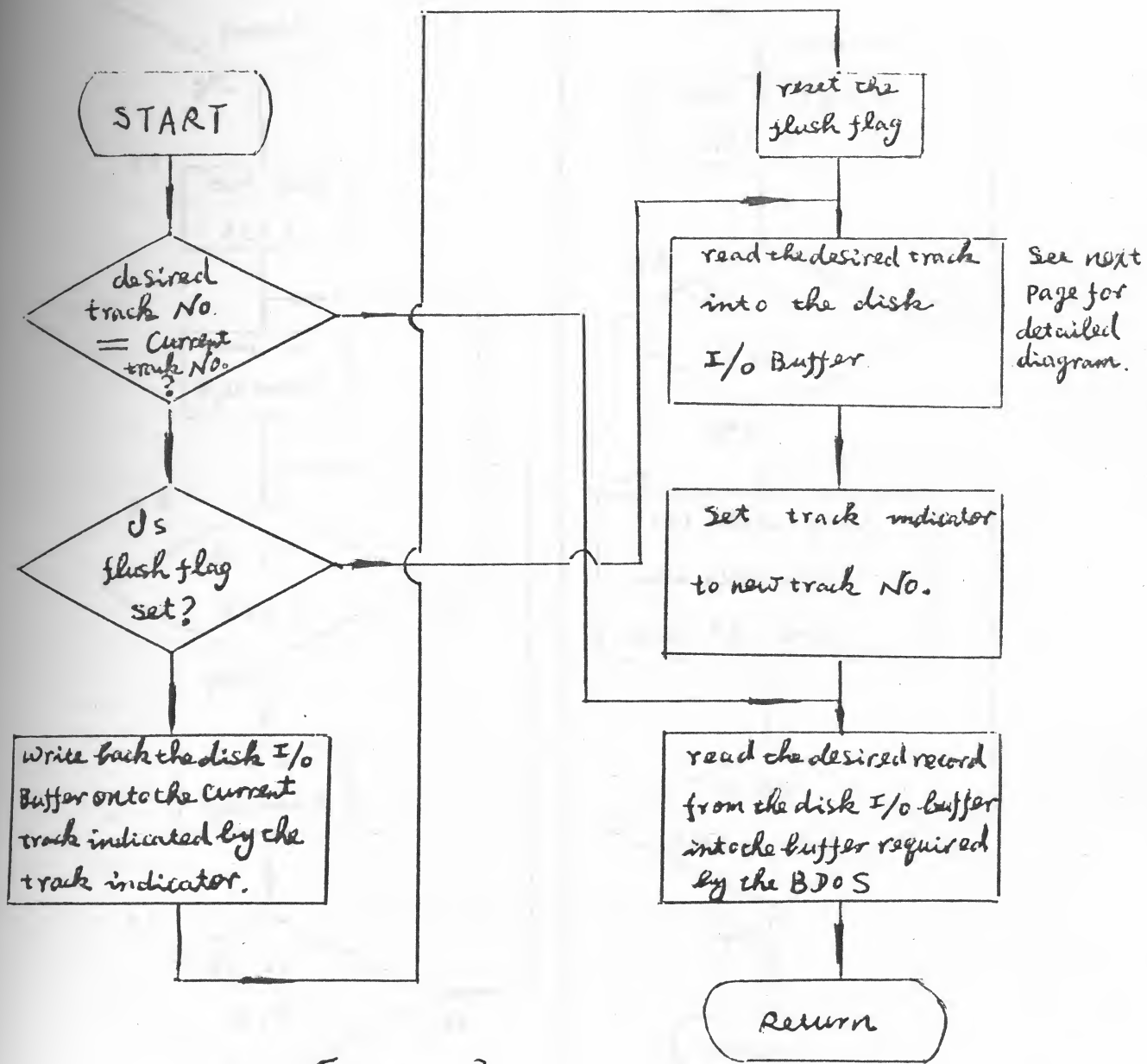


Figure 2

Disk read algorithm in OUP/M

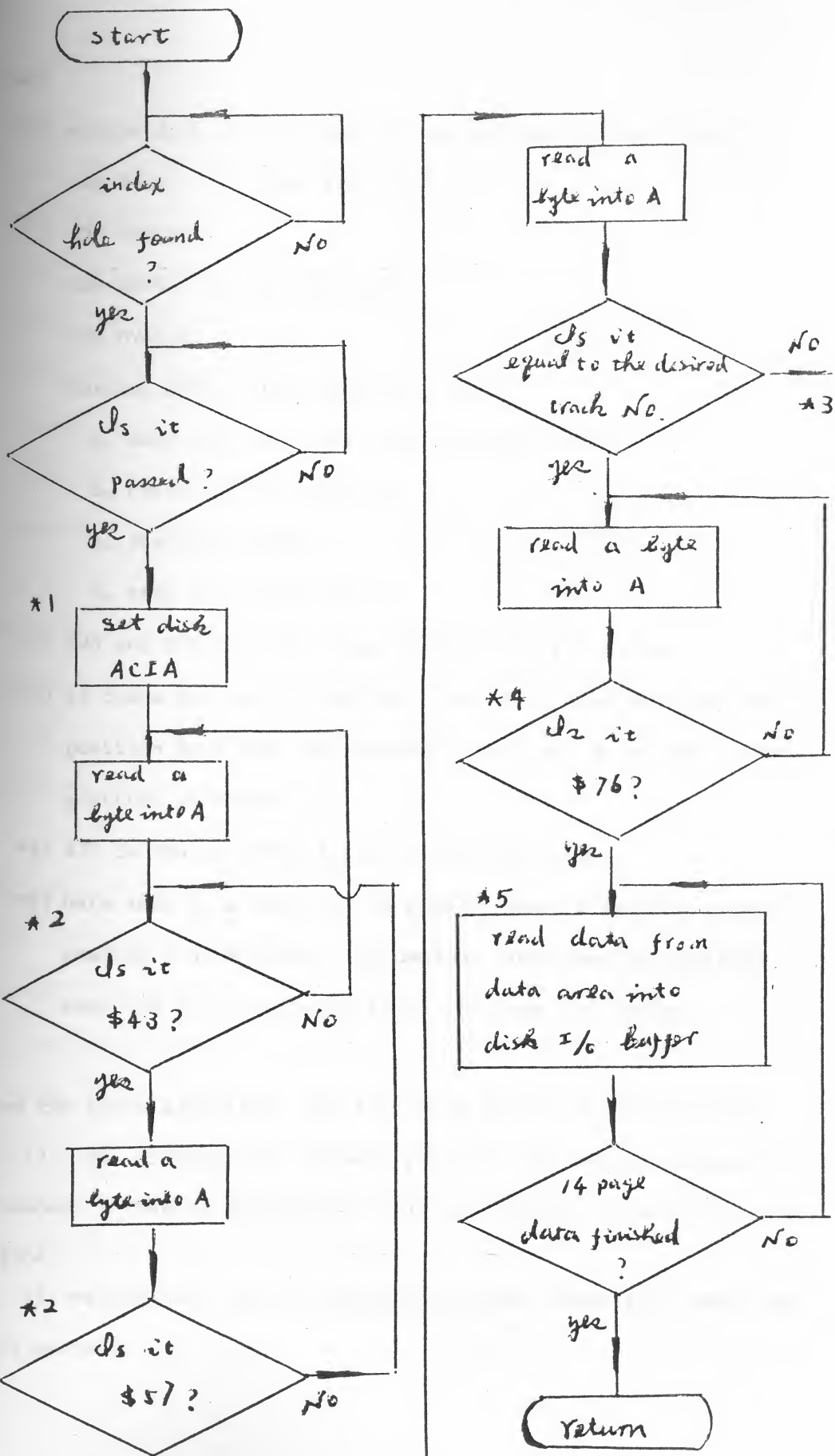

start
index hole found ?
NO
yes
Is it passed ?
NO
yes
*1
set disk ACIA
read a byte into A
*2
Is it $43 ?
NO
yes
read a byte into A
*2
Is it $57 ?
NO
yes
read a byte into A
Is it equal to the desired track No.
NO
*3
yes
read a byte into A
*4
Is it $76 ?
NO
yes
*5
read data from data area into disk I/O buffer
14 page data finished ?
NO
yes
return

Figure 3

Where

*1) setting disk ACIA is done by the following instuctions:

```
LDA #$03     ; reset disk ACIA
STA DVACIA
LDA #$58     ; set disk ACIA
STA DVACIA
```

Putting $58 to disk ACIA will set:

a. baud rate to clock rate divided by one.

b. chracter bit number to 8.

c. parity to even.

d. stop bit number to 1.

*2) $43 and $57 are the track header identification.

*3) If track No. is not matched, then home head to track zero, position head over the desired track and go to the 'start' position of Figure 3.

*4' $76 is the starting flag for the data area.

*5) Data area in a track is 14 pages. When a desired record reading (or writing) is needed, this area is entirely read out to (or written from) the disk I/O buffer.

From the above algorithm, the following points should be clear:

1) No distinction between physical and logical sectors is needed. Thus no conversion table and routine is needed in the BIOS.

2) We do not need to recognize sector headers -- there are no sectors.

3) No delay loop is needed for sector gaps -- there are no gaps.

4) No disk I/O is needed except when the track boundaries are crossed.

5) Only one revolution is needed to read (or write) a whole track. This is faster than what CP/M does, where 6 revolutions are required for reading (writing) an entire track.

6) As a consequence of the above point, there is less headwear, lengthening the life of the read-write head.

Thus we have simplified and increased the speed of the disk I/O operation -- a faster and simpler disk performance is obtained at the cost of using some extra memory, namely the utilization of 3.5 K bytes for a track buffer.

One more thing needs to be mentioned, before we advance to next the topic. From the disk R/W algorithm, we know a modified track in the disk I/O buffer will not be written to its corresponding track until the track boundary is crossed. Thus the latest track will remain in the disk buffer and the modified data could be lost, if the current disk is switched or the current program has completed.

This problem is solved as follows:

1) When the current disk is switched, a special routine is called to write the modified track buffer to the current

track.

2) When the program is over and the control is returned to the CCP, the same routine is called in the CCP to take care of writing the modified buffer to the current track.

Warning: None of these protections will be effective if a disk is physically removed from the drive. In this case, lots of data on the disk may be lost.

2.8 Block size and an even/odd addressing algorithm

The block size or allocation size is 512 bytes in this system. Therefore each file is a multiple of 512 bytes in length and there are 511 blocks available for file storage. Thus it differs from CP/M where block sizes are multiples of 1 K bytes.

The reason for adopting the smaller block size is to let the user create smaller files without wasting much disk space. For example, a file of 9 records will occupy 3 blocks (1.5 K) if the block size is 512 bytes, while it will occupy 2 blocks (2 K) in 1 K block size system. There are 3 * 128 = 384 bytes and 7 * 128 = 896 bytes disk space wasted respectively. Thus the larger block size will waste more disk space than the smaller one and this will become tremendous when many tiny files are involved.

Recall that a CP/M directory entry is a 32-byte record. The first 16

bytes , called the file parameter area, hold information such as the file name, the file type and the extent number. The second 16 bytes, called file location area, are pointers to the file location. Usually they are single-byte pointers and can point to at most 256 blocks. Thus how to point to 511 blocks with such pointers will be a problem when the 512 byte block size is to be used.

To solve the problem mentioned above, we number each pointer from 0 through 15 according to their order in the file location area, and thus implicitly assign an even/odd meaning to them. We also define logical block number and store these logical block numbers, instead of physical block numbers, in the even or odd pointers as needed. The conversion rule between them is as follows:

If the pointer is even, we have

physical block number = (logical number - 1) * 2

otherwise, we have

physical block number = (logical number - 1) * 2 + 1.

Note as a consequence of this rule, even physical block numbers must correspond to even pointers while odd ones must correspond to the odd pointers.

Thus the same logical block number in different even or odd pointers will point to a different physical blocks. In another words, we can use a single-byte pointer in the file location area to point to up to 511 physical blocks.

However, this method has a disadvantage. If there are more files ending with an even block than those ending with an odd block, then more odd blocks will be unfilled. Thus the disk may appear full without actually being full. This is because every file must start at an even block, by the rule that each block No. must match its even or odd pointer.

The following are two alternative ways to solve the above problem:

1) In the file location area, we could use pairs of consecutive bytes as a double-byte pointer, which will then permit us to point to any of the 511 blocks. But the amount of total directory entries will be reduced by a factor of 2, unless a reserved track (say, track No. 2) is used as another directory track.

2) In fact, to point to 511 blocks, we only need 9 bits. The extra bit could be found somewhere in the directory entry. By using the bit 7 of each first 16 bytes in the directory as the bit 8 of each corresponding second 16 bytes, we expand the single-byte pointer from 8 bits to 9 bits, which can point to any of 511 blocks.

In order to use bit 7 of each of the first 16 bytes, the following changes or restriction should be mentioned:

a. We would restrict the user code in the 'dr' field to be less than $80.

b. Bit 7 of 'f1' through 'f8' and 't1' through 't3' is

regarded as the file attribute bit in this system. In order to use this bit, we could change the file attribute bits to bits 0 through 5 of 's1' and 's2' which are not used by the system or the user.

c. The maximum extent No. and record No. within the directory are 112 and 64 respectively. Thus there is no problem to use their bit 7.

2.9 Disk parameter tables

Tables are included in the BIOS which describe the particular characteristic of the disk subsystem used by the CP/M. These tables can be either hand-coded or automatically generated using the DISKDEF macro library included in the CP/M software package. With these tables, CP/M can easily deal with various disk subsystems and therefore has become a widely used standard oprating system for the 8080- and Z80-based microcomputers.

The first version of OUP/M operating system is intended to be specific to the OSI work environment. Thus a much simpler but more specific disk parameter header table is used in OUP/M.

In the following subsections, a comparision is made betwteen the tables in the CP/M and those in OUP/M along with the detailed description of the terms in each table.

2.9.1 Disk parameter header (DPH)

In general, each disk drive has an associated disk parameter header(16 bytes) which contains information about this disk and provides some scratchpad areas for certain BDOS operations.

The format of a disk parameter header in CP/M is shown below:

XLT	SRP1	SRP2	SRP3	DIRBUF	DPB	CSV	ALV
16b	16b	16b	16b	16b	16b	16b	16b

The format of disk parameter header in the OUP/M is shown below:

DIRBUF	CSV	ALV	0000
16b	16b	16b	16b

where each term refers to a 16 bit word. The meaning of each term is:

XLT contains the starting address of the conversion table between logical sector No. and physical sector No., if the value of XLT is non-zero. Otherwise logical and physical sector No. are same and no conversion is needed. Disk systems with the same sector skew factors will share the same conversion table. This word is not used in OUP/M.

SRP1, SRP2 and SRP3 are scratchpad area reserved for special BDOS operations. The initial values are not critical. These are not used in OUP/M.

DIRBUF contains the starting address of a 128-byte directory buffer. Since there is only one directory buffer, every DIRBUF in each DPH has the same value. DIRBUF in OUP/M contains the starting address of a 512-byte directory buffer. Two drives in OUP/M adress the same directory buffer.

DPB contains the starting address of the disk parameter block for this disk. Disks with the same characteristics have the same value in DPB. OUP/M does not use it (see next section for details).

CVS contains the starting address of the checksum area and is unique to each drive. The size of the area is determined by CKS in the DPB (see next section for details). In OUP/M, CVS contains the starting address of the checksum area and is unique to drive 'A' and 'B'. The size of this area has a fixed value, namely 14 bytes (see section 2.2.2 of Appendix A for details).

ALV contains the starting address of the bit map area and is unique to each drive. The size of the area is determined by the maximum number of data blocks for this particular disk and computed as (DSM / 8) + 1 (see next section for details). In OUP/M, ALV contains the starting adress of the bit map area and is unique to drive 'A' and 'B'. The size of this area is fixed and computed as 512 / 8 = 64 bytes.

2.9.2 Disk parameter block table (DPB)

The disk parameter block (DPB) for each disk in the CP/M is rather complex. A particular DPB, which is addressed by one or more DPH's, takes the form mentioned below. However there is no actual DPB table installed in OUP/M. All values are fixed and buried in the algorithm for the specific OSI work environment. This is one of the things which should be modified in the second version of OUP/M.

DPB table in CP/M

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
16b	8b	8b	8b	16b	16b	8b	8b	16b	16b

Where each term is a byte or word value, as shown by the '8b' or '16b' indicator below the field.

SPT is the total number of sectors per track. In OUP/M, SPT is equal to 28 (sectors).

BSH and BLM are the data allocation block shift factors. The value of BSH and BLM determine implicitly the block size BLS (bytes per block). The correspondence between them is shown below:

BLS	BSH	BLM
1K	3	7
2K	4	15
4K	5	31
8K	6	63
16K	7	127

Using BSH and BLM, instead of BLS, the desired logical sector number corresponding to various block size can be easily decided. In OUP/M, we do not use BSH and BLM and the BLS is fixed as 512 (bytes).

EXM is the extent mask. The value of EXM depends upon both BLS and whether the DSM (see DSM term) value is less than or greater than 255 as shown in the following table.

BLS	DSM < 256	BSM >= 256
1K	0	-
2K	1	0
4K	3	1
8K	7	3
16K	15	7

In OUP/M we don't use EXM.

DSM is the maximum number of data blocks supported by this particular drive, measured in BLS units. Note it doesn't include the reserved operating system tracks. In OUP/M, DSM is equal to 511 (blocks) and does not include the operating system tracks and directory track.

DRM is one less than the total number of directory entries, which can take on a 16-bit value. In OUP/M, DRM is equal to 7 * 4 * 4 - 1 = 111, a fixed value.

AL0 and AL1 are determined by DRM. They can be considered together as

a string of 16-bits, namely:

```
-------------------------------------------------
          AL0          I          AL1
-------------------------------------------------
I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
-------------------------------------------------
 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
```

Position 00 corresponds to the high order bit of the byte labelled AL0, and position 15 corresponds to the low order bit of the byte labelled AL1. Each bit position reserves a data block for number of directory entries, thus allowing a total of 16 data blocks to be assigned for directory entries (bits are assigned starting at position 00 and filled to the right until position 15). each directory entry occupies 32 bytes, resulting in the following table.

BLS	DIRECTORY ENTRIES
1K	32 times the bits filled
2K	64 times the bits filled
4K	128 times the bits filled
8K	256 times the bits filled
16 K	512 times the bits filled

Thus, if DRM = 127 (128 directory entries), and BLS = 1024 , then there are 32 entries per block, requiring 4 reserved blocks. In this case, the 4 high order bits of AL0 are set, resulting in the value AL0 = $F0 and AL1 = $00. In OUP/M, we do not use AL0 and AL1, because total number of directory entries is fixed to be 112.

CKS is the size of the checksum area for the drive. If the disk drive media is removable, then CKS = (DRM + 1) / 4. If the media is fixed, then CKS = 0 (no directory record are checked in this case). In

OUP/M, the size of checksum area is equal to 14 bytes.

OFF is number of tracks reserved for system use and skipped at the beginning of the physical disk. In OUP/M, this number is equal to 4 (tracks). Among these, track No.0 and No.1 are reserved for operating system, track No.2 is reserved for future use and track No.3 is used for directory entries.

2.10 Cold loader and system booting

The way we arrange cold loader and BIOS in the same track, track 0, makes the cold loader much simpler than the CP/M does. We don't need to put the disk read routine into the cold loader. When the rest of the operating system needs to be loaded, the cold loader just moves the BIOS to its permanent location and calls 'warm boot'.

Scince the OSI computers have two or more system clock frequencies, OUP/M includes a routine during clod boot to determine which of 1mz, 2mz or 3mz clocks is being used. Knowing the clock frequency is critical in doing disk I/O. The routine looks like:

```
        * = $2200
        DVACIA = $C010
        FRYCST = $E41B
        SFTSWT = $F701
        FRYCCT   .BYTE $31,$31,$31,$31,$31
```

```
                .BYTE $62,$62,$62,$62
                .BYTE $A0,$A0,$A0,$A0
                .BYTE $A0,$A0,$A0,$A0
        LDA #$34                ; select 6502 CPU.
        STA SFTSWT
        LDY #$00
        LDA #$03                ; set disk ACIA.
        STA DVACIA
        LDA #$38
        STA DVACIA
        STA DVACIA + 1          ; partial timing delay
        PHA                     ; depending on the clock
        PLA                     ; frequency requiring from
        PHA
        PLA
        PHA
        PLA
        STX DVACIA + 1
AD$TFY  LDA DVACIA              ; wait until interrupt is
        BMI A$JTFY              ; reported.
        INY
        BNE AD$TFY
A$JTFY  LDA FRYCCT,Y            ; select proper timing
        STA FRYCST              ; constant.
```

2.11 Parameter passing

In 8080-based (or Z80-based) CP/M, there are six registers along with an accumulator, namely B, C, D, E, H, L and A. The system function No. and some required parameters are passed to the BDOS via register C and register pair DE .

In 6502-based OUP/M, there are only two registers X, Y and an accumulator A. Hence the system function No. is passed via register X and the required parameter is passed via register Y and accumulator A.

2.12 Stack usage

Due to the design of the 6502 CPU, There is only one stack, fixed at $0100 through $01FF. Any program must share the stack with the system program when it is running. Thus the user's use of the stack may be limited.

In designning the OUP/M operating system, we seldom use the stack except when 'JSR' and 'RTS' instructions are involved. Thus we leave almost all of the stack for the user.

2.13 Error message handling

There are only three error situations in the CP/M which the BDOS intercepts during file processing, namely 'BAD SECTOR', 'SELECT' and 'READ ONLY'.

The 'BAD SECTOR' message indicates that the disk controller electronics has detected an error condition in reading or writing the disk. The 'SELECT' error occurs when there is an attempt to address a drive beyond the number of installed drives. The 'READ ONLY' error message tells the user that there is an attempt to write to a disk which has been designated as read only.

OUP/M provides a more detailed error message by displaying an error number which is described in the error message table (see chapter 7 of appendix A for details).

For example, instead of displaying 'BAD SECTOR' in the CP/M, the error processing routine in the OUP/M gives the error No. $21 indicating 'index hole found, while reading'. This may happen due to following reasons:

1) Track No. is not matched.

2) Track identification is missed.

3) Starting flag of data area can not be found.

The above information gives the user a more vivid error indication.

2.14 Rearranging the OUP/M in memory

The OUP/M operating system memory map is shown in chapter 2 of appendix A. The system program occupies 6.4 K memory, of which CCP uses

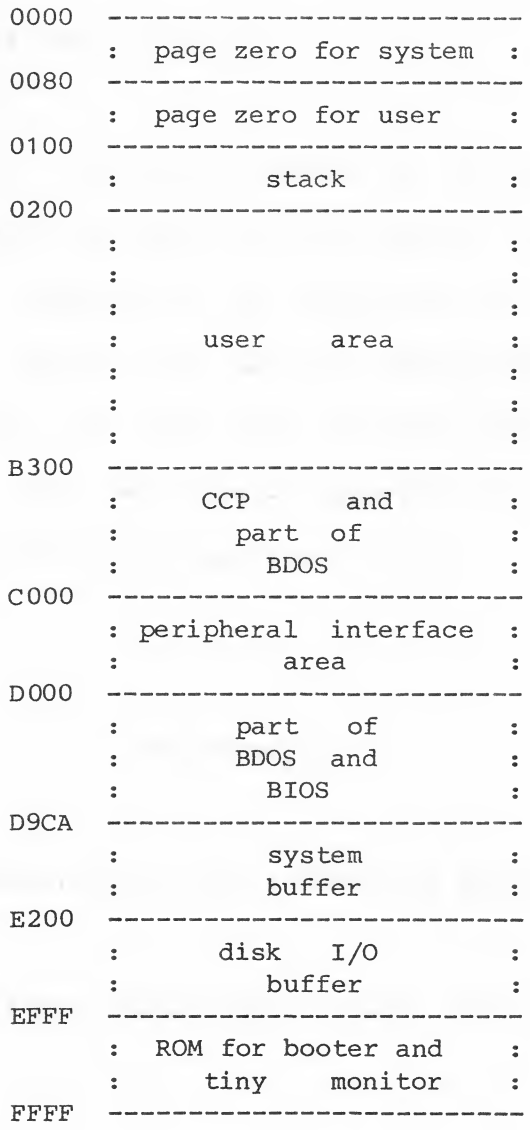


Figure 4
A recommended memory map

2.3K, BDOS uses 2.7 K and BIOS uses 1.4 K. The system buffer occupies 1.6 K memory, of which only 1.35 K are used. The disk I/O buffer occupies 4 K memory, of which only 3.5 K are used. Thus there are approximately 0.75 K memory wasted.

The Figure 4 is a recommended layout for the next version of OUP/M. This new layout would not waste 0.75K of memory. Further, the user can get extra memory temporarily by overlaying the CCP area and doing a 'warm boot' just before the program terminates. In the present version, the CCP can also be overlayed but requires delicate programming since the user area is separated from the CCP area by the disk I/O buffer and the general peripheral area.

Bibliography

1. Digital Research, CP/M2 FDOS & UTILITIES MANUAL. NEW YORK, 1979.

2. Klaus E. Eldridge, OULAB3 USER'S GUIDE. OHIO, 1982.

Appendix A : OUP/M Operating System Manual

OUP/M OPERATING SYSTEM

USER'S MANUAL

Table of contents

CHAPTER 1

INTRODUCTION

CP/M is an monitor control program for microcomputer systems. It is a single user, single task operating system. As an early entry into the microcomputer market, CP/M was originally designed to run on the 8080 based computer system which used IBM- compatible flexible disks for data storage. Appropriate modification had to be made to accomodate a particular hardware I/O configuration under the CP/M. A lot of work has been done to modify the BIOS (an I/O dependent portion of the CP/M) and install the modified CP/M to any 8080(or Z80) based computer system.

There was not too much interest in modifying the CP/M to run on the machines which use the 6502 processor as their CPU. As some examples, an APPLE II computer system installs a Z-80 card in its periphral slot in order to run the CP/M system, which makes it not convenient and flexible for use and also costly. The Commodore system does not have the standard CP/M operting system, it has to develop its own DOS system.

OUP/M is a single user, single task operating system developed for OSI computers with a dual 8'' disk drive. To a large extent this system mimics the CP/M operating system. One major difference is that OUP/M is 6502-based whereas CP/M is 8080- or Z80-based.

Every effort has been made to obtain a portable OUP/M system. However, due to limited development resources, this version is pretty much specific to the OSI computer. More work needs to be done to transport OUP/M to such computers as Apple II Plus, Apple IIe, Commodore Pet, Commodore 64, etc.

One needs to be aware of other deviations from the CP/M system besides the CPU difference. The present disk format is specific to the OSI disk and does not follow the IBM 3740 format. Block size for additional file space is handled in increaments of 512 bytes rather than 1024 bytes. Consequently a different algorithm is required for converting logical block to physical block number on the disk. The directory entry itself still remains at 32 bytes per extent.

Disk I/O is handled via a track buffer using 14 pages (14 * 256 bytes) per track. This removes the necessity for interleaving physical records and also simplifies and speeds up disk I/O.

Page zero plays an important role in the 6502 assembly language programming. A compromise is presented for the system and the user to use the page zero if a full page zero is to be used, it is user's resposibility to create a swapper to take care of zero page swaping.

A much simpler cold loader than found in the CP/M is installed on the track No. 0 along with the BIOS to take care of the system booting.

System function No. and some required parameters are passed to the BDOS via register X, register Y and accumulator A in this system, instead of register C and register DE in a 8080 or Z80 system as the CP/M usually does.

One other noticable difference is the numbering and organization of the system functions. All, except two, CP/M functions have been included but are numbered differently and grouped by peripheral device I/O, disk I/O and other user functions.

The six standard CCP commands are included and perform as expected.

Even with these differences, OUP/M has a very close resemblance to the CP/M operating system. Thus those users experienced with CP/M will have minimum difficulties switching systems.

Detailed information for using the OUP/M operating system is presented in the following chapters.

The first two chapters give some basic concepts necessary for understanding this system and reading the remaining chapters. Chapter 2 describes the OUP/M stucture, disk and memory layout from which the user can get the general outline of this system. Chapter 3 contains a complete description of the disk file structure, which plays an important role in the OUP/M system. Some basic concepts concerning the

disk file, such as file directory, bit map, log-in vector, etc, are also discussed in detail.

The next two chapters talk about two important parts of the system: CCP and BDOS. Chapter 4 presents the 6 built-in commands along with various examples to show how to use these commands. Chapter 5 is the central part of this manual as well as of this system. It presents 35 system functions along with the explanation of their usage and the way to call them from assembly programms. A very important concept -- file control block (FCB) , which the user will frequently meet when dealing with disk I/O operation, is carefully explained.

Chapter 6 describes each page zero location used by the system. Those who intend to know more about this system and plan to read the system program will benefit from this information. Chapter 7 lists all the error messages along with their numbers. Chapter 8 presents a way for developing other utilities such as assembler, editor, loader, etc, into this system under the OSI work environment. Chapter 9 briefly describes the disk read/write utility -- the only utility installed so far. With this utility, back-up copies can be made and modification of the system may be easily saved.

CHAPTER 2

OUP/M Structure, Memory and Disk Layout and booting the system

2.1 OUP/M Structure

Like CP/M, OUP/M is logically divided into three parts:

BIOS Basic Input/Output System.

BDOS Basic Disk Operation System.

CCP Console Command Processor.

. BIOS is a collection of I/O subroutines, which provides the primitive operations necessary to access the disk drives and to interface the system with standard peripherals. The standard devices available now are a console and a line printer. It is possible to add another standard devices (such as paper tape reader/punch) and user-defined devices (such as a serial communications device).

. The BDOS basically provides dynamic disk file control and management. There are 35 system functions available to the assembly

user.

. CCP is a built-in command interpreter program. It accepts, recognizes and executes the command typed in by the user through the console.

The basic relationship among these three parts is as follows:

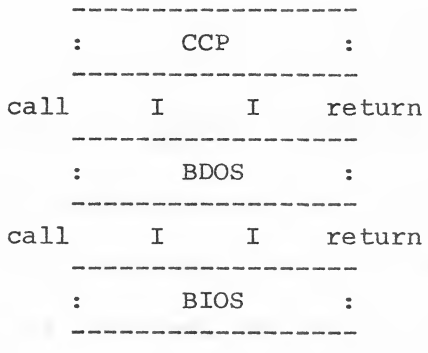


Fig. 2.1 System Structure

2.2 Memory Layout

2.2.1 General Memory Map

Under OUP/M, memory is divided into several parts as follows: (see Figure 2.2)

Page zero: Locations $0000 - $007F are used by the system, the user should not use them.

Locations $0080 - $00FF are available to the user

Page one: Locations $0100 - $01FF are reserved for the hardware stack.

User area : Locations $0200 - $AFFF is the area where system utility and user programs run. When the user wants to run a program, the system loads it into this area with $0200 as the start of execution address.

Disk I/O buffer ($B000 - $BFFF) : Unlike CP/M, we have no physical sector header information on the track. When the system wants to read a record from a certain track, it reads the whole track into this buffer and moves the data between it and the default I/O buffer (or user defined buffer).

Peripheral interface area ($C000 - $CFFF) : This area is for peripheral interfaces such as ACIA, PIA and some other interfaces. We currently only use:

$C000 - $C003 -- Disk control ports.

$C010 - $C011 -- Disk I/O port.

$CF02 - $CF03 -- Line printer I/O port.

OUP/M operating system ($D000 - $EFFF) : See next section for detail.

Console I/O port : $FC00 - $FC01.

Boot ROM and tiny monitor ($F000 - $FFFF) : This area is for the system bootstrap program and tiny monitor. There are some exceptions.

$FFFA - $FFFB Non-remaskable interrupt vector,

$FFFC - $FFFD Reset vector,

$FFFE - $FFFF Interrupt or break vector

are 6502 vectors which point to $01C0, $FFA0,$0130 respectively.

2.2.2 Operating System Memory Usage

When the system is booted, it will occupy the memory from $D000 - $EFFF. Roughly, we can divided it into four areas (see Fig. 2.3).

CCP : $D000 - $D939

BDOS : $D93A - $E419

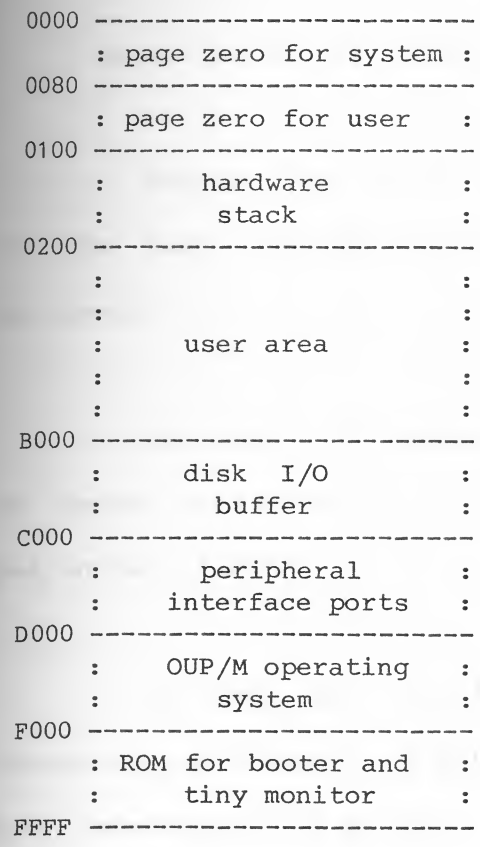


Figure 2.2
memory layout

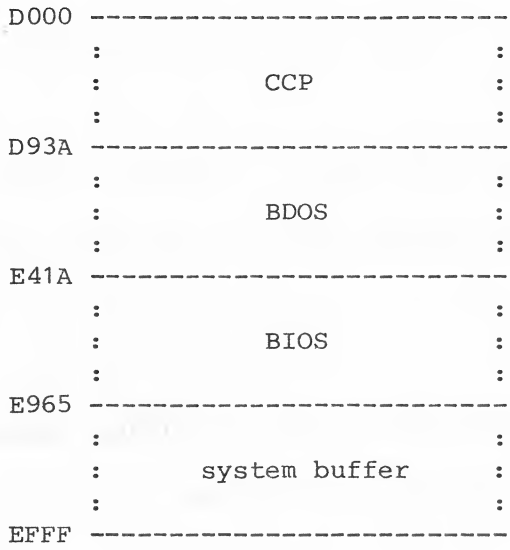


Figure 2.3
Operating system layout

BIOS : $E41A - $E964

System buffers : $E965 - $EFFF.

System buffers are arranged as follows: (see Fig. 2.4)

Default I/O buffer ($E965 - $EB64) : Unlike CP/M ,it is 512 byte long. This permits reading / writing 128 byte records or 512 byte blocks.

Directory I/O buffer ($EB65 - $ED64) : This is for holding the content of directory while doing file operations such as open, read, write and etc.

Bit map area for disk A ($ED65 - $EDA4) : It holds usage information for disk A. A bit value of 0 means corresponding block is empty, otherwise it is in use.

Bit map area for disk B ($EDA5 - $EDE4) : The same as above,but for disk B.

Check sum area for disk A ($EDE5 - $EDF2) : Consecutive byte pairs contain the check sum for one of the seven blocks in the directory, which occupy exactly one track. If the user does not type CTRL- C after inserting another disk, the values in this area will not match the check sum of the corresponding directory blocks and an error

will occur.

Check sum area for disk B ($EDF3 - $EE00) : The same as above, but for disk B.

Default file control block ($EE01 - $EE23) : See 5.1 for detail.

Command input buffer ($EE80 - $EEFF) : This buffer is used by the CCP for responding to user typed-in commands.

FCB copy area ($EE10 - $EF2F) : A copy of the File Control Block when a transient command or a user program is being executed.

2.3 Disk Distribution and Structure

2.3.1 Track assignment (see figure 2.5)

Track 0 contains cold loader, BIOS and part of BDOS.

Track 1 contains CCP and rest of BDOS.

Track 2 is reserved for further use.

Track 3 contains the directory of each file in the disk.

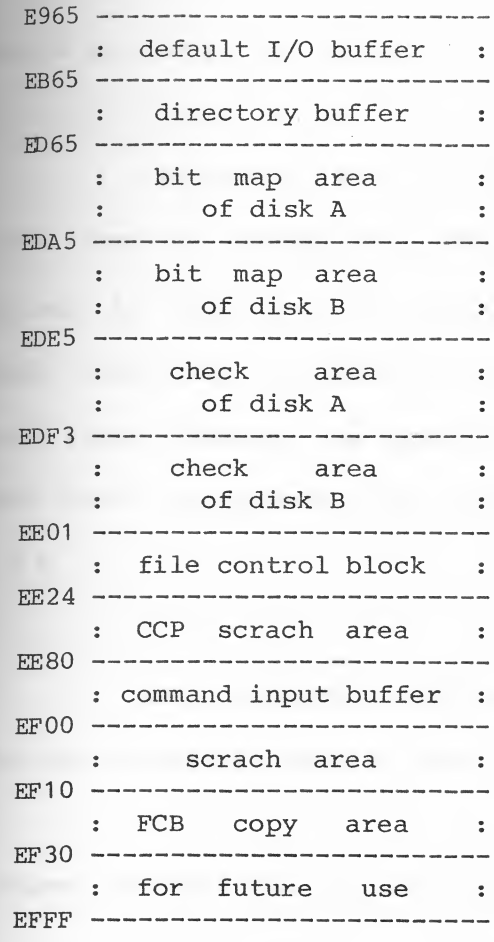


Figure 2.4
system buffer

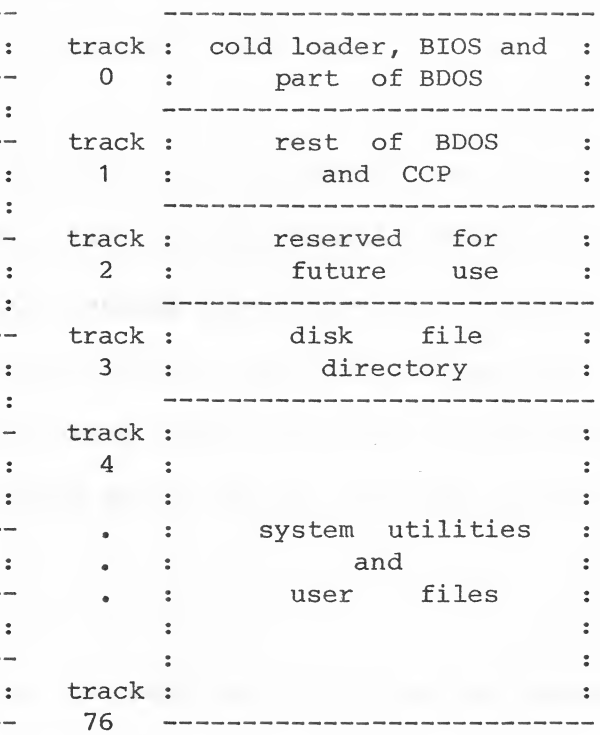


Figure 2.5
disk layout

Track 4 - 76 are storing system utilities and user programs.

2.3.2 Physical disk orgnization

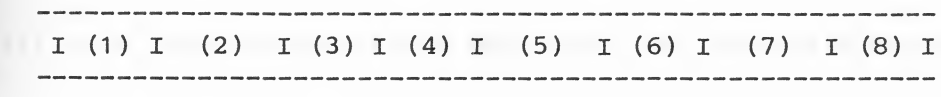
. Each track has 7 blocks. Each block has 4 records. Each record has 128 bytes. Thus each track holds 3584 bytes.

. Beginning from track No.4 through track No.76, we number the block from 0 through 510. Thus track no.4 has blocks numbered from 0 through 6, track no.5 has blocks numbered from 7 through 13, and so on until track no. 76 which has blocks numbered from 504 through 510. We call these numbers the physical block numbers in order to distinguish them from the logical block numbers which will be discussed in section 3.2.2.

. As mentioned before, we have no physical sector headers. The only headers we have on the disk are track headers.

There are two kinds of track formats which we will discuss below.

1. Track no. 2 to no.76 format



where

(1) is a index hole position from which the track R/W starts.

(2) indicates a 1 ms delay is needed.

(3) contains track beginning identification values: $43 and $57.

(4) contains the track number.

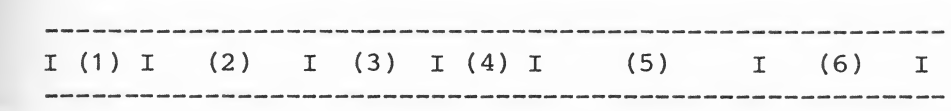
(5) indicates a 1.4 ms delay is needed.

(6) contains the data area starting flag: $76.

(7) represents the data area.

(8) contains the ending flags: $47 and $53.

2. Track zero format



where

(1) is a index hole position from which the track R/W starts.

(2) indicates a 1 ms delay is needed.

(3) contains the loader vector. It is a memory location address to which the data in this track will be loaded. Currently it is set to $2200.

(4) contains the total number of pages for track zero.

(5) represents the data area.

(6) contains the ending flags: $47 and $53.

This information is required for the disk boot ROM.

2.4 System Booting

The disk to be booted must be in the drive A because the existing ROM boostrap is written with this assumption. After power on, pressing ' reset ' and typing ' D ', the system will be initialized as follows.

First step : The permanent bootstrap program will load the program on track 0, which contains cold loader, BIOS and part of BDOS into memory starting from $2200, then give the control to cold loader.

Second step : The cold loader first moves the BIOS and part of the BDOS into their permanent locations. Then it passes the control to the BIOS.

Third step : The BIOS uses the 'warm boot' routine to read the CCP and the rest of BDOS from track no. 1 into memory. Control is then passed to the CCP.

Fourth step : The CCP displays the system version message and gives the prompt 'A>', then waits for a command from the user.

CHAPTER 3

Disk File

3.1 Disk File space

The disk file space is from track No. 4 through track No. 76. Track No. 0 through No. 3 contain the operating system and the directory.

The unit of a file is the block. There are 511 blocks, numbered from 0 through 510 in the disk file space as mentioned before. A file whose size is less than one block (512 bytes) will occupy one block on the disk. A file whose size is between one block (512 bytes) and two blocks (1024 bytes) will occupy two blocks on the disk, and so on.

The block in a file need not be contiguous, this is because of the dynamics of file accessing. Blocks are allocated to a file, de-allocated from a file as needed. The directory keeps track of which blocks are allocated to which files.

The bit map in memory records allocated and free blocks for a particular disk.

Unlike CP/M, in our system, each block contains four consecutive sectors (records). No logical sector number and no physical sector interleaving are needed.

Physical records are not numbered internally. Numbering is through relative location from the beginning of the file in step of 128 bytes. It's up to the user to keep track of logical records which are smaller or larger than one physical record.

3.2 File Directory

3.2.1 Directoy space

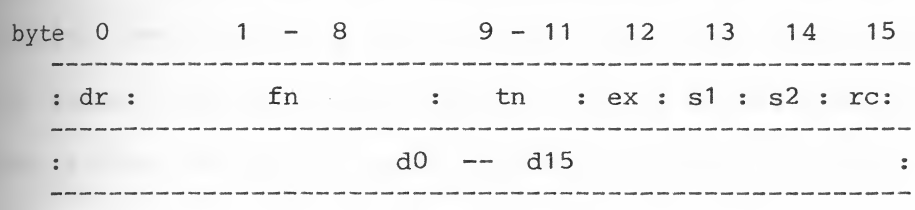
OUP/M uses a fixed length directory of 7 blocks on track No. 3. Each directory block has 16 directory entries of 32 bytes. Thus we can have at most 112 files on each disk.

3.2.2 Directory Entry

The directory entry consists of 32 bytes. We call the first 16 bytes the file parameter area and the last 16 bytes the file location area.

The directory entry format is illustrated as follows:

File Parameter Area



File Location Area

Fig. 3.1

. File parameter area (see Fig. 3.1)

dr (byte 0) : A value E5 represents an empty entry available to the system for reassignment. Otherwise any value represents a user code for an non-empty entry.

fn (byte 1 - 8) : file name. This holds the file name with byte 1 containing first character. If less than 8 characters are needed for the file name, the remaining bytes will be filled with blanks ($20).

tn (byte 9 - 11) : file type. This holds the file type with byte 9 containing the first character. If less than 3 characters are needed, the others will be filled with blanks.

ex (byte 12) : file extent. We call 16 blocks as one

extent. A file having a size less than one extent (16 blocks) occupies one directory entry and it's file extent number is 0. A file having a size between one extent (16 blocks) and two extents (32 blocks) occupies two directory entries with first entry having extent number 0 and second having extent number 1, and so on(see examples in detail). Thus, a file can have at most 32 extents filling the disk.

s1 (byte 13) : is for future use.

s2 (byte 14) : is for future use.

rc (byte 15) : number of 128 byte records within this extent.

. File location area (see Fig. 3.1)

d0 -- d15 (byte 16 -31) : The last 16 bytes in the directory entry form the file location area. Each byte within the area is a single-byte pointer and contains the logical block number with which we can locate the corresponding physical file block on the disk. Using file parameter and location information, we can easily locate and access all the blocks of a file extent.

. Recall that the disk file space is divided into physical blocks. But we still need to use the logic block number, why? What is

the definition of the logic block number? We will answer this questions below:

As mentioned before, we have 511 physical blocks which need 2 bytes to point to (one byte can point to at most 256 blocks). If we use 2 bytes in file location area to point to one file block, the total number of files which can be identified in the disk (even though there are a lot of space in the disk file area) will be reduced by 2. In order to solve this problem, we use the so called 'even/odd address' method in defining a logical block number.

In the FCB file location area, we number each single-byte pointer (from the beginning of this area) as 0,1,...15, which also assigns an even/odd meaning to each pointer.

With this assumption, we can now talk about logical block number. The logical block number can be translated into physical block number by the rule:

If the number is stored in the even byte then

physical bolck number = (logic block number - 1) * 2,

otherwise, we have

physical block number = (logical number - 1) * 2 + 1.

The fact that the same logic block number will represent two different physical block numbers will actually make it possible to use a single-byte pointer in the file location area to point to 511 physical blocks.

After finding physical block number, we get the corresponding track number and the relative block number within this track as follows:

Track number = (physical block number DIV 7) + 4

The relative Block number = physical block number MOD 7

Thus, we find the location on the disk on which the corresponding file block is stored.

We illustrate these concepts in the following two examples:

Example 1. The following is a directory entry for a disk utility program.

File Parameter Area

dr :	fn								:	tn		:	ex	s1	s2	rn
01	44	53	4B	55	54	59	20	20	43	4F	50		00	00	00	1A
01	01	02	02	03	03	04	00	00	00	00	00		00	00	00	00
d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11		d12	d13	d14	d15

File Location Area

File parameter area : ' dr ' being 01 means user code is 01. The next eleven bytes correspond to file name DSKUTY and file type COM. The ' ex ' being 0 means the file extent number is 0. From ' rn ', we know there are 26 records in the file.

File location area : From the facts that byte 0 is an even byte and it's value is 01, we know the corresponding physical block number is (1 - 1) x 2 = 0. Further, we also know the track No. where the corresponding file block is located, is

(0 div 7) + 4 = 4,

and the relative block number within this track is

0 mod 7 = 0.

Using the same method we find that the whole file occupies block No. 0, 1, 2, 3, 4, 5, 6 on track No. 4 -- the entire track.

Example 2. The following is an example for a file requiring two directory entries.

Extent No. 0

File Parameter Area

dr :	fn							: tn		: ex	s1	s2	rn
01	58	59	5A	20	20	20	20	20	46	47	52	00	00	00	40
05	05	06	06	07	07	08	08	09	09	0A	0A	0B	0B	0C	04
d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15

File Location Aarea

Extent No. 1

File Parameter Area

dr :	fn							: tn		: ex	s1	s2	rn
01	58	59	5A	20	20	20	20	20	46	47	52	00	00	00	0F
0D	0C	0E	0D	00	00	00	00	00	00	00	00	00	00	00	00
d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15

File Location Area

This file has the name XYZ.FOR and two extents. In extent No. 0, there are 64 records (16 blocks). In extent No. 1, only 15

records are used. Track No.5, 6, and 7 are involved in storing this file. Track No. 5 and 6 are completely used, while on track No. 7, only the first 6 blocks are used, the remaining block is unused.

3.3 Bit map area

To each disk drive , there corresponds a bit map in memory. 512 bits, or 64 bytes, are used to record the status of the 511 physical blocks on the disk. The first byte corresponds to physical block No.0 to block No.7, the second to block No.8 to block No.15 ..... and so on. If a bit is zero, the corresponding block is empty. Otherwise the space is occupied by a file. By checking each bit in the bit map, the system knows where the empty space is located. This makes it easy for the system to assign empty space to a file and get the space back from an erased file.

There is also a byte in memory called the disk log-in vector. Bits 0 and 1 of this byte correspond to disk 'A' and 'B' respectively. A '1' indicates the correspondng disk is logged in and the bit map has been built for it.

Each time the user cold or warm boots the system, the bit map for disk A and the current disk are built and these disk are logged in. When a disk in a drive is exchanged for another one, the user must type CTRL-C to warm boot the system and log in the new disk. Otherwise an I/O error may be occur.

CHAPTER 4

CONSOLE COMMAND PROCESSOR (CCP)

The user interacts with the system through the console command processor (CCP),which reads the command from the console to the command buffer located at $EE80 - $EFFF, analyses it, and then branches to the corresponding command routine.

Upon cold boot, the CCP takes over the control, displays the version message , logs in disk A, and then prompts the user with 'A>' (indicating the current (default) disk is A) for typing in a command.

The CCP implements the user typed-in commands at two levels: built-in command and transient command. Built-in commands are part of the CCP itself,while transient commands are resident on the disk as utilities. They will be loaded into the user area starting at $0200 and executed when needed.

4.1 Console Command Processor

The CCP reads and interprets the commands as follows:

STEP 0 : First it calls system function No. 10 (reset disk system) to select and log in disk A. (Note: This step is executed only in the case of cold boot or warm boot. Otherwise The CCP begins at step 1.)

STEP 1 : Sets the stack pointer to $01FE,prompts the user to type in a command, gets the command(if any) and puts it into the command buffer located at $EE80-$EEFF.

STEP 2 : Builds the FCB by putting the command name into the file name and file type fields, the disk drive No.(if any) into 'dr' field. It checks the FCB to see if the file name exists:

a. If no file name exists, CCP assumes this is a select disk drive commmand and selects the drive according to the drive code in the 'dr'. It then builds the appropriat bit map, logs in the disk if it was not logged in before and then goes to step 1.

b. Otherwise goes to setp 3.

STEP 3 : Checks to see if the file name in the FCB is a built-in command. There are two tables in the CCP. The first one is a built-in command table located at $D021-$D038 which has the following form:

'DIR ERA TYPESAVEREN USER'.

The second one is a command routine entry table with each entry corresponding to the command located in the first table. The CCP scans the first table for a match with the command in the FCB. If found, it gets the corresponding entry address from the second table and branches

to that routine. After the command is executed, it goes to step 1.

STEP 4 : If a match cann't be found, the CCP assumes this is a transient command. Adding the file type 'COM' to it, the CCP scans the file directory to find a match.

If a match is found, the CCP loads the file into the user area starting at $0200 and gives the control to the program. When this program terminates, control is returned to the CCP at step 1.

However, before loading the trnsient command, the CCP does two things:

a. Copies the FCB to the FCB copy area located at $EF10-$EF2F for back-up .

b. Copies the parameters following the command name (if any) in the command buffer to the default I/O buffer located at $E965-$EB64 with the first byte containing the length of total number of characters in the parameter list.

For example, if the command is 'EDIT TEST.ASM', the length of 8 will be put into the first byte and the ASCII string 'TEST.ASM' will be stored into $E966-$E96D. Note 'EDIT' is normally a transient commnad.

STEP 5 : If a match is still not found, the CCP displays the

message ' NO FILE ' and goes to step 1.

4.2 Line editing and output control

The CCP allowes the user to use the followuing editing function when it is expecting a command from the keyboard:

RUB/DEL remove and echoes the last character.

CTRL-C warm boot when at the beginning of a line.

CTRL-E cause physical end of a line.

CTRL-H backspace one character position.

CTRL-J perform a line feed

CTRL-M perform a carriage return.

CTRL-R retype the current line on the next line.

CTRL-X return to the begnning of the current line.

The output control funtion allowed are CTRL-P and CTRL-S.

CTRL-P is a toggle switch between 'all output to the console' and 'all

output to the console and the printer.

CTRL-S will stop the output temporarily until any key is pressed. It is used to stop the output on the high speed display device so as to view a segement of the output before continuing.

4.3 File reference

A file reference identifies a certain file or a group of files on a particular disk. They can be either 'unambiguous' or 'ambiguous'. An unambiguous file reference indicates a unique file while an ambiguous file refernce can be satisfied by a group of files.

4.3.1 Unambiguous file reference

.An unambiguous file reference identifies a unique file in a specified disk. Its form is as follows:

[X:] FILENAME [.FILETYPE]

where

1. X: is the name of the disk drive on which the file is stored. In this system the legal values are 'A:' and 'B:'.

2. FILE NAME is the primary file name for the file and is composed of up to 8 printable characters except any of the following special characters < > . , ; : = ? *

3. FILE TYPE is the extension file name for the file and is composed of up to 3 printable characters with the exception metioned in file name.

By convention, the file type more specifically describes the kind of data in the file. The system recognizes several default file used for special purposes. For example:

COM	Executable command program
ASM	Assemble source file
HEX	Hex format file
PRN	Print or listing file

4. Lower case characters in the file reference are allowed and will be converted to upper case by the CCP.

5. The terms between the square brackets, such as X: and .FILE TYPE, are optional. If a drive name is omitted, it is assumed that the current disk is referred.

.Examples:

Example 1. A: PROGRAM1.ASM represents a file with the primary name PROGRAM1 and the file type ASM and is stored in the disk A.

Example 2. B: PROGRAM1.ASM represents a file with the same file name as in example 1 but is stored in disk B.

Example 3. PROG.ASM represents a file with the primary file name PROG and the file type ASM and is stored in the current disk which may be either diak A or disk B.

4.3.2 Ambiguous file reference

.An ambiguous file reference is used for directory search and pattern matching. The form of an ambiguous file reference is similar to an unambiguous file reference except the symbol '?' may be interspersed thoughout the primary and extension file names and the symbol '*' may replace the whole primary and/or extension file name. In various commands within the OUP/M, the '?' matches any character of a file name in the '?' position and the '*' is equvalent to '????????', if it occurs in the position of the primary name, or to '???' if it is in the extension name.

.Examples:

The examples which we will discuss are based on the following unambiguous files:

```
PROGRAM1.ASM
PROGRAM2.ASM
PROG    .ASM
PROGRAM1.COM
PROGRAM2.COM
PROG    .COM
```

Ambiguous file reference	Corresponding unambiguous files
PROGRAM?.ASM	PROGRAM1.ASM
	PROGRAM2.ASM
PROG????.ASM	PROG .ASM
	PROGRAM1.ASM
	PROGRAM2.ASM
PROGRAM1.*	PROGRAM1.ASM
	PROGRAM1.COM
.	all files
*.COM	PROG .COM

PROGRAM1.COM

PROGRAM2.COM

4.4 Built-in commands

Built-in commands are part of CCP program itself. There are 6 built-in commands in this system, which will be discussed in detail in the following sections:

ERA erases specified files.

DIR displays the specified file names and their sizes.

REN renames the specified file.

SAVE saves the file into disk under a specified name.

TYPE types an ASCII file at the console.

USER changes the user code.

In the description below, we assume the following abbrevitions:

ufn represents the unambiguous file refereces

afn represents the ambiguous file references

4.4.1 Selecting disk command

The user can select the current (default) disk by typing the disk name followed by a colon (:). The following example will demonstrate the usage of this command and its result. At the beginning, we assume the current disk is 'A'.

```
A > DIR                         ; lists all the files on disk A
PROGRAM1.COM   8
PROGRAM2.ASM   12

A > B:                          ; selects disk B

B > DIR DSKFL1.*                ; list all the files
                                ; with primary name DSKFL1
                                ; in disk B
DSKFL1   ASM   20
DSKFL1   COM   10
DSKFL1   DAT   15

B > A:                          ; switch back to disk A
```

4.4.2 ERAse command

FORMAT: ERA afn

This command will remove all files which match the ambiguous file reference afn from the specified disk. If the file can not be found, an error message ' NO FILE ' will appear. Otherwise files are removed and the space occupied by the files will be returned to the free space list.

The following examples illustrate the use of this command:

Example 1. ERA *.* will erase all files on the current disk. Before erasing, The CCP will prompt the message

ALL (Y/N) ?

If the user types Y, all the files will be removed and the control will be returned to the CCP.

Example 2. ERA PROGRAM?.COM will erase all the files on the current disk which match the ambiguous file reference PROGRAM?.COM.

Example 3. ERA B:PROGRAM1.* will delete all the files on disk B with the primary name PROGRAM1.

4.4.3 DIRectory command

FORMAT: DIR afn or DIR X:

This command will list the names and size of all files which match the

ambiguous file reference afn, if command is ' DIR afn ' or all the files in the specified disk, if the command is ' DIR X: '. If no file can be found, the error message ' NO FILE ' will be displayed at the console.

.Examples

Example 1. DIR A: (or DIR, if A is the current disk) will list all the files at the console in the following format:

```
PROGRAM1  ASM  10   PROGRAM2  ASM  15   PROGRAM3  ASM  20

PROGRAM1  COM   8   PROGRAM2  COM  12   PROG      COM  18
```

where for each file listed, the first column is the primary file name, the second is the extension file name and the third is the file size with the 'page' as its unit.

Example 2. DIR *.* will display the names and size of all files on the current disk.

Example 3. DIR B: *.ASM will display all the files with extension name 'ASM' on the disk B.

4.4.4 RENname command

FORMAT: REN ufn2=unf1

This command will change the file name which satisfies the unambiguous file reference ufn2 to the ufn1. It also allows the user to type a left-arrow instead of the equal sign in the command. An optional drive name is allowed to precede either ufn1 or ufn2 (or both). Thus there are four cases we will discuss below:

Case 1. If both ufn1 and ufn2 are not preceded by the drive name, the current disk is assumed to each file.

For example, REM PROGRAM1.COM=PROGRAM1.ASM will change the file name PROGRAM1.COM in current disk to PROGRAM1.ASM which will reside on the same current disk.

Case 2. If ufn1 is preceded by a drive name, the ufn2 is assumed to exist on the same disk as ufn1.

For example, REN PROG.ASM=B:PROG1.COM will change the file name PROG.ASM on disk B to PROG1.COM and the file will still reside on the same disk with the new file name.

Case 3. If ufn2 is preceded by a drive name, the ufn1 is assumed to reside on the same disk as ufn2.

For example, REM A:PROGRAM1.COM=PROG.COM will change the file name PROGRAM1.COM to PROG.COM and the file will still reside on the same disk

under the new name.

Case 4. If both ufn1 and ufn2 are preceded by the drive names, the same name must be specified. Otherwise The CCP will ignore the drive name of ufn1.

For example, REM A:PROG1.COM=B:PROG1.ASM will change the file PROG1.COM on disk A to PROG1.ASM which will still reside on disk A.

If ufn1 already exists on the specified disk, the error message ' FILE EXISTS ' will appear and no change is made. If ufn2 does not exist on the disk, the message ' NO FILE ' is printed at the console.

4.4.5 SAVE command

FORMAT: SAVE n ufn

where n is a decimal integer for the number pf pages to be saved.

This command will save n pages of data onto the disk from the user area starting with the location $0200 and under the file name ufn. If the disk name is omitted in ufn, the current disk is assumed. Following are examples which illustrate the use of this command:

SAVE 5 PROG.COM saves information from memory location $0200 through $06FF to the current disk under the name PROG.COM.

SAVE 20 B:PROGRAM1.ASM saves imformation from memory location $0200 through $15FF to disk B under the name PROGRAM1.ASM.

4.4.6 TYPE command

FORMAT: TYPE ufn

This command will display an ASCII source file named by ufn at the console. If the disk name is omitted in ufn, the current disk is assumed. Note: only ASCII file should be displayed. Typing the file other than this form will give unpredictable results.

Valid TYPE commands are:

TYPE PROG.ASM will type the ASCII file PROG.ASM in the current disk at the console.

TYPE B:PROGRAM1.ASM will type the ASCII file PROGRAM1.ASM in the disk B at the console.

4.4.7 USER command

FORMAT: USER n

where n is the user code to be changed and could have a decimal integer value in the range 0 to 15.

A user code is used to protect the user's file from unauthorized access.

Upon cold boot, the user code is automatically set to 0. But the user is allowed to change it to his own code by typing this command.

4.5 Transient commands

Transient commands are names of programs residing on the disk in executable form. Normally these fall into two catagories: utilities and user developed programs. They will be loaded into the user area and executed when needed. We have not built up these commands yet except for DSKUTY. Consult chapter 8 to see how to install them into this system when needed.

CHAPTER 5

Basic Disk Operating System (BDOS)

BDOS is a collection of 35 system functions which manage and access disk files dynamically and do all peripheral device I/O by calling BIOS. It gives the assembly programmer flexibility and convenience by allowing a simple call to these functions. In the fllowing sections, we will provide details for them.

5.1 File Control Block (FCB)

Before discussing the 35 system functions, we introduce the FCB (File Control Block) concept which will be frequently met when dealing with disk I/O operation. The FCB and file directory are both required when creating, opening, reading, writing, closing, or deleting files. If a file is created or opened, information from the directiory is placed into the FCB; if a file is closed or deleted, information from the FCB is placed into the directory; if reading or writing occurs, information flows back and forth between the FCB and the directory.

The structure of FCB is almost the same as that of file directory except in 'dr', 'cr' 'r0' and 'r1'. The following is the diagram of the FCB stucture and the listing of all the fields in the FCB and their usage.

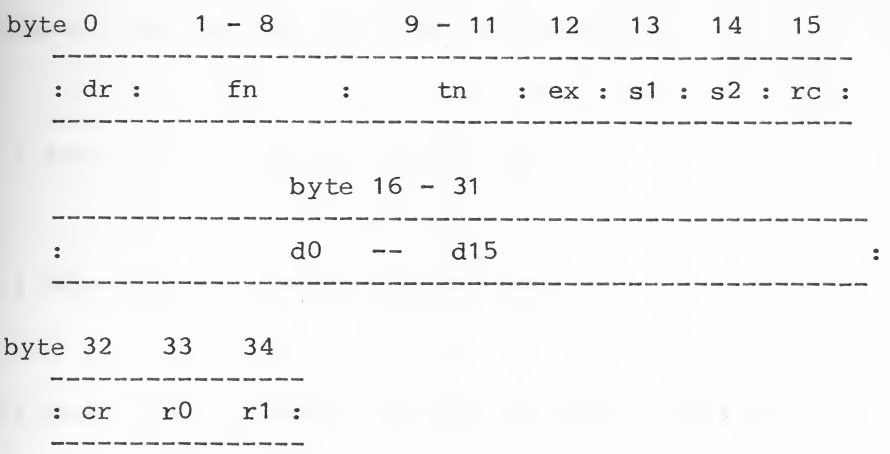


figure 4.1 FCB stucture

where:

1. dr (byte 0) : Disk No. involved

0 -- current drive

1 -- drive A

2 -- drive B

2. fn (byte 1 - 8) : file name. This contains the file name with byte 1 containing the first character. If less than 8 characters are needed, the rest will be filled with blanks ($20).

3. tn (byte 9 -11) : file type. This contains the file type with byte 9 containing the first character. If less than 3 characters are needed, the remaining bytes will be filled with blanks.

4. ex (byte 12) : file extent. This holds the file directory

extent number(see section 3.2.2 for definition).

5. s1 (byte 13) : is for future use.

6. s2 (byte 14) : is for future use.

7. rc (byte 15) : number of records within this extent.

8. d0 - d15 (byte 16 - 33) : file locatin area. (see section 3.2.2 for details).

9. cr (byte 32) : This contains the current record No. available for access in a sequenial file. It must be set to 0 by the user before reading/writing from/to the beginging of a sequential file.

10. r0 and r1 (byte 33 - 34) : contain the random record No. with low byte in r0 and high byte in r1 when the random file access is needed.

5.2 Calling System Functions

In order to call system function, the user needs to do the following things:

1. The system function No. must be placed into register X.

2. Parameters which the function requires must be placed into registers Y and A with high byte in Y and low byte in A.

3. Call location 5 by using 'JSR 5'.

The following is an example of opening a file with it's name and type in the default FCB.

```
FLCNBK = $EE01        ; $EE01 is the start address of default FCB
BDOS   = $0005        ; $0005 is system function call entry
LDX    #$16           ; $16 is the function No. for open file
LDY    #FLCNBK/256    ; high byte of the default FCB to Y
LDA    #FLCNBK        ; low byte  of the default FCB to A
JSR    BDOS           ; call the system
```

There is a branch table containing the start addresses of the 35 system functions in memory from $DBC3 to $DC08 within the BDOS. There is also a system function selecting routine in which each system call is checked, processed and then branched to as follows:

1. Examine the function number to see if it is legal or not, that is, if it is in the range 0 - 34. If it is illegal, an error will be displayed at the console.

2. Multiply the legal function number by 2 to get the offset and add it to the starting address of the branch table mentioned above to get

the address of the corresponding function routine.

3. Branch to the routine by using JMP (adr) , where adr is an address of the location in which the corresponding function address is held.

Thus the BDOS can easily handle each system calls and execute them.

5.3 System functions

. The function names along with their numbers in OUP/M are listed below:

0	System reset	19	Select disk
1	Console input	20	Create file
2	Console output	21	Delete file

3	List output	22	Open file
4	Direct console output	23	Close file
5	Buffer output	24	Read record sequentially
6	Read console buffer	25	Write record sequentially
7	Get I/O byte	26	Read record randomly
8	Set I/O byte	27	Write record randomly
9	Get input console status	28	Search for first
10	Reset disk system	29	Search for next
11	Get log-in vector	30	Rename file
12	Get current disk number	31	Set file attributs
13	Get map address	32	Compute file size
14	Set write potection	33	Set random record
15	Get read only vector	34	Sent I/O buffer to disk

16 Get parammeter table adr.

17 Set/get usercode

18 Set DMA address

. The function names along with their number in CP/M are listed, for those user experienced with CP/M to make the comparision, as follows:

0	System reset	19	Delete file
1	Console input	20	Read sequential
2	Console output	21	Wrtie sequential
3	reader input	22	make file
4	Punch output	23	Rename file
5	List output	24	Return login vector
6	Direct console output	25	Return currene disk
7	Get I/O byte	26	Set DMA address

8	Set I/O byte	27	Get addr (alloc)
9	Print string	28	Write protect disk
10	Read console buffer	29	Get R/O vector
11	Get console status	30	Set file attributes
12	Return version number	31	Get addr(disk parms)
13	Reset disk system	32	Set/get user code
14	Select disk	33	Read random
15	Open file	34	Write random
16	Close file	35	Compute file size
17	Search for first	36	Set raddom record
18	Search for next		

. Details of the OUP/M system functions:

```
*********************************************************
*                                                       *
*  FUNCTION  0 : SYSTEM RESET                           *
*                                                       *
*********************************************************
*                                                       *
*  Entry parameters :                                   *
*      Register  X  :  $00                              *
*                                                       *
*********************************************************
```

This function will reload the CCP, part of the BDOS and return control back to the operating system at the CCP level. The CCP then re-builds the bit map and check sum area for disk A and logs in it by setting the log-in vector. This function has the same effect as typing CTRL-C.

As a benifit to the assembly user,it allows him to use an extra 2K or more memory (from $D000-$D988) in which CCP is located when an assembly program is running. At the end of the program, the user can call this function to restore the system.

```
*********************************************************
*                                                       *
*  FUNCTION  1 :  CONSOLE INPUT                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry parameters  :                                  *
*      Register  X   :  $01                             *
*                                                       *
*  Returned  Value  :                                   *
*      Register  A  :  ASCII char                       *
*                                                       *
*********************************************************
```

This function will read a character from the console to register A. It does not return to the calling program until it receives a character Most control characters will not be echoed except a few such as CTRL-J (LF), CTRL-M (CR), CTRL-H (BK). The TAB character (CTRL-I) is expanded and the next character will be displayed at one of the following TAB positions: 1 ,9 ,17 ,25 ,33 ,41 ,49 ,57 ,65 or 73. CTRL-P will toggle the printer on or off.

All non-control characters are echoed.

```
*******************************************************
*                                                     *
*  FUNCTION  2 :  CONSOLE OUTPUT                      *
*                                                     *
*******************************************************
*                                                     *
*  Entry Parameters  :                                *
*     Register  X   : $02                             *
*     Register  A   : ASCII character                 *
*                                                     *
*******************************************************
```

This function will send the content of register A as an ASCII character to the console . It will wait until the console is ready. If the printer is on, the output will be sent to both console and printer. CTRL-P toggles the printer on or off. The TAB character is expanded as mentioned in function No. 1.

```
*******************************************************
*                                                     *
*  FUNCTION  3 :  LIST OUTPUT                         *
*                                                     *
*******************************************************
*                                                     *
*  Entry Parameters  :                                *
*      Register  X   : $03                            *
*      Register  A   : ASCII char.                    *
*                                                     *
*******************************************************
```

This function will sent the content of register A as an ASCII character only to the logical listing device, usually this is the printer.

```
*******************************************************
*                                                     *
*  FUNCTION  4 :  DIRECT CONSOLE I/O                  *
*                                                     *
*******************************************************
*                                                     *
*  Entry Parameters  :                                *
*      Register  X   : $04                            *
*      Register  A   : $FF(input) or Char(output)     *
*                                                     *
*  Return value  :                                    *
*      register A  : Char or Status                   *
*                                                     *
*******************************************************
```

This function will read a character from console to register A, if previous content of register A is $FF. Otherwise, it will sent the content of register A as an ASCII character to the console. In both case, control characters are treated as any other character. This is an unadorned console I/O function, unlike function No.1, 2, and 3.

```
*********************************************************
*                                                       *
*  FUNCTION  5 :  BUFFER OUTPUT                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameter  :                                   *
*      Register  X  : $05                               *
*      Register  Y,A: Buffer address                    *
*                                                       *
*********************************************************
```

This function will sent the character stored in an output buffer,whose starting address is indicated by the register Y and A, with Y holding high byte and A lower byte, until a '$' sign is encountered. TAB characters will be expanded. This function responds to start/stop scroll control (CTRL-S/CTRL-Q). CTRL-S halts output until CTRL-Q or any key is depressed. CTRL- P toggles the printer on or off, whether it is entered from the keyboard or is imbedded in the output buffer.

```
*********************************************************
*                                                       *
*  FUNCTION  6 :  READ CONSOLE BUFFER                   *
*                                                       *
*********************************************************
*                                                       *
*  Entry parameters  :                                  *
*      Registre  X   : $06                              *
*      Register  Y,A : Buffer address                   *
*                                                       *
*  Returned value  :                                    *
*      Console input in the buffer                      *
*                                                       *
*********************************************************
```

This function will read anf edit a line from the console and put into a buffer addressed by registers Y and A. This function terminates either when a 'CR' is entered or the buffer is full.

The input buffer has the following form:

```
+0 +1 +2 +3 +4 +5 +6 +7 +8 ..... +n
---------------------------------------
mx nc c1 c2 c3 c4 c5 c6 c7 ..... cn-1
---------------------------------------
```

where 'mx' is the maximum number of characters the buffer can hold. This value must be specified before the function is called. 'nc' is the number of the characters which the buffer contains after returning from this function. The remaining bytes (c1 - cn-1) hold the input characters.

This function allows the following control characters for line editing:

RUB/DEL removes and echoes the last character .

CTRL-C warm boot if itis the first characetr entered, otherwise CTRL-C remains in the buffer.

CTRL-E causes physical end of a line.

CTRL-H backspaces one character position.

CTRL-J line feed

CTRL-M carriage return

CTRL-R retype the current line on the next line.

CTRL-X return to the beginning of the current line.

```
*********************************************************
*                                                       *
*  FUNCTION  7  :  GET I/O BYTE                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $07                              *
*                                                       *
*  Returned value  :                                    *
*      Register A  : I/O byte value                     *
*                                                       *
*********************************************************
```

This function will return the I/O byte in register A. For the definition of I/O byte ,see the function No.8.

```
*********************************************************
*                                                       *
*  FUNCTION  8  :  SET I/O BYTE                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $08                              *
*      Register  A   : I/O byte value                   *
*                                                       *
*********************************************************
```

This function allows user program to redirect the I/O by setting the I/O byte.

The I/O byte is located at $0003 and it's format is as follows:

BIT	7,6	5,4	3,2	1,0
DEVICE	LIST	PUNCH	READER	CONSOLE

where

1. For the logical console (bit 1,0)

BIT PATTERN	ACTUAL DEVICE
00	console
01	serial communication device
10	console
11	user defined device

2. For the logical listing device (bit 7,6)

BIT PATTERN	ACTUAL DEVICE
00	console
01	serial communication device
10	printer

3. for logic reader (bit 3,2) and punch (bit 5,4)

This system does not make any provisions for the logical reader and punch. It is available for future expansion of the system.

```
*********************************************************
*                                                       *
*  FUNCTION  9  :  GET INPUT CONSOLE STATUS             *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $09                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : Console status                     *
*                                                       *
*********************************************************
```

This function will test whether a character is ready for input. Upon return, if the value in register A is $FF, the console is ready for input. If is $00, it is not. This is equivlent to testing if a key has been pressed.

```
*********************************************************
*                                                       *
*  FUNCTION  10  :  RESET DISK SYSTEM                   *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0A                              *
*                                                       *
*********************************************************
```

This function allowes the user to reset the disk system under program control. It places all disks into the read/write state, logs in disk A and sets the DMA to the default I/O buffer.

It may be used to change the disk medium without rebooting the system.

```
*********************************************************
*                                                       *
*  FUNCTION  11  :  GET LOG-IN VECTOR                   *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0B                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : Log-in Vector                      *
*                                                       *
*********************************************************
```

This function will return the log-in vector located at $DBC2 in register A. Bit 0 corresponds to drive A and bit 1 to drive B. A '0' bit means the drive is not on-line, while a '1' bit means the drive is actively on-line and the bit map for the disk in the drive has been

built.

```
*********************************************************
*                                                       *
*  FUNCTION  12  :  GET CURRENT DISK NUMBER             *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0C                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : current drive number               *
*                                                       *
*********************************************************
```

This function will return the current selected drive number in register A. '0' indicates drive A, while '1' indicates drive B.

```
*********************************************************
*                                                       *
*  FUNCTION  13  :  GET MAP ADDRESS FOR CURRENT DISK    *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0D                              *
*                                                       *
*  Returned Value  :                                    *
*      Register Y,A: map address of current disk        *
*                                                       *
*********************************************************
```

A bit map area is maintained in memory for each on-line disk. Various system utilites use the information provided by this area to determine the available storage on a disk. The base address of the bit map in the current active disk is returned in registers Y and A, with Y

holding the high byte and A the low byte.

```
*********************************************************
*                                                       *
*  FUNCTION  14   :  SET WRITE PROTECTION FOR           *
*                    CURRENT DISK DRIVE                 *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0E                              *
*                                                       *
*********************************************************
```

There is a read - only vector located at $DBC1. Bit 0 corresponds to disk A, while bit 1 to disk B. This function sets a ' 1 ' into the corresponding bit for current active disk. Thus the current disk can be temporarily write protected.

```
*********************************************************
*                                                       *
*  FUNCTION  15  :  GET READ ONLY VECTOR                *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $0F                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : read only vector                   *
*                                                       *
*********************************************************
```

This function returns the read only vector (see function 14) in register A.

```
*********************************************************
*                                                       *
*  FUNCTION  16  :  GET PARAMETER TABLE ADDRESS OF      *
*                   CURRENT DISK                        *
*                                                       *
*********************************************************
*                                                       *
*  Entry parameters  :                                  *
*      Register  X   : $10                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  Parameter table address           *
*                                                       *
*********************************************************
```

This function will return the BIOS resident disk parameter block address in register A. This address can be used for either of two purposes. First, the disk parameter values can be extracted for display and computational Thus the c or user can dynamically change the values as required.

```
*********************************************************
*                                                       *
*  FUNCTION  17  :  SET/GET USERCODE                    *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $11                              *
*      Register  A   : $FF (get)  or user code (set)    *
*                                                       *
*  Returned value  :                                    *
*      Register A  :  User code (get)                   *
*                                                       *
*********************************************************
```

This function will get the user code from location $DBC0, if register A is $FF. Otherwise it will set the user code equal to the value in register A.

```
*********************************************************
*                                                       *
*  FUNCTION  18  :  SET DMA ADDRESS                     *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $12                              *
*      Register  Y,A : DMA address                      *
*                                                       *
*********************************************************
```

DMA is a logical name of the I/O buffer which holds the data to be sent to or obtained coming from the disk through the disk I/O buffer. Upon cold boot, warm boot or disk system reset, the address of DMA is set to default I/O buffer located at $E965 to EB64. This function ,however,can change it to wherever the data record resides , or is going to reside.

In this version, the length of the DMA is 128 bytes which is the length of a standard record. Provisions have been made for a DMA length of 512 bytes in case block I/O functions are added.

```
*********************************************************
*                                                       *
*  FUNCTION  19  :  SELECT DISK                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $13                              *
*      Register  A   : Disk No. desired                 *
*                                                       *
*********************************************************
```

This function will select the disk drive named in Register A as the default (current) disk for subsequent file operation (with 0 for drive A and 1 for drive B). It will also build up a corresponding map for this drive and log it in.

Once the default disk is selected, a '0' for the drive code (dr) in the FCB will automatically reference to this disk in subsequent disk operation. The drive code of 1 or 2 will refer directly to drive A or B.

Changing disk in this drive without typing CTRL-C while it is on-line will automatically change it to read/only status and an error will occure.

```
*******************************************************
*                                                     *
*  FUNCTION  20  :  CREATE FILE                       *
*                                                     *
*******************************************************
*                                                     *
*  Entry Parameters  :                                *
*     Register   X   : $14                            *
*     Register   Y,A : FCB address                    *
*                                                     *
*  Returned value  :                                  *
*     Register  A  : $FF,If directoy overflow         *
*                    $00 -- $6F, if success           *
*                                                     *
*******************************************************
```

This function will create directory enrty for the named file in the FCB in the disk directoy area. Before calling it ,the user must fill the file name ,type and the desired disk drive No. in the FCB.

The user must assure that no duplicat file name exists in the derictory . Otherwise an error, 'BDOS EROR No. $05', will appear.

Upon return, the value $FF in register A indicates no more space in directory area, while a directory code, from $00 to $6F, in register A indicates a successful operation.

This function will also open the file for subsequent I/O. Thus an open operation after this is redudant.

The following is an example that creates a file named XYZ.COM in disk drive A:

```
BDOS=$0005
*=$EE01
FLCNBK  .BYTE  $01,'XYZ     COM'
;fill the file name,type and drive No. into
; default FCB located in $EE01
*=$0200          ;program starting at $0200
LDX  #$14        ;$14 is the function No. for make file
LDY  #FLCNBK/256 ; high FCB addr to Y
LDA  #FLCNBK     ; low FCB addr to A
JSR  BDOS        ; call the open function
```

```
*********************************************************
*                                                       *
*  FUNCTION  21  :  DELETE FILE                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $15                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : $FF,if file not found              *
*                    directory code,if success          *
*                                                       *
*********************************************************
```

This function deletes all file entries which match the file name in the FCB addressed by registers Y and A. The file name and type may be ambiguous references(i.e. a '?' can be in these position),but the drive code cann't be ambiguous.

Upon return, register A contains $FF,if the file cann't be found. Otherwise a value equal to directory code, from $00 to $6F, is in register A.

```
*********************************************************
*                                                       *
*  FUNCTION  22  :  OPEN FILE                           *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $16                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*      Register A  : $FF,if file not found              *
*                    directory code,if success          *
*                                                       *
*********************************************************
```

This function will open an existing file, whose file name and type is given in the FCB addressed by registers Y and A, to allow a subsequent file read or write operation. Note, an existing file must not be accessed until a successful open operation is completed.

When this function is called, the BDOS scans the the referenced disk directory for a match in position 'dr' through 's2' in the FCB, where an '?' mark matches any character in any of these position. Normally no question mark is included in 'dr' and fields 'ex', 's1' and s2 should be set to 0 by the user before calling this function.

If a directory entry is matched, the relevant directory information, such as 'rc' and 'd0' through 'dn', are copied from the matched entry into the the corresponding positions in the FCB, allowing the subsequent read or write operation to retrieve and get sufficient file location infomation, and thus work properly.

Upon return, register A will contain $FF,if the file is not found. Otherwise register A will contain a directory code, from $00 to $6F, indicating a successful operation.

Note that the current record No. must be zeroed by the user program after calling this function, if a subsequent sequential read or write operation from the first record is needed.

The following is an example of opening a file named TEST.ASM in drive

B for a subsequent read. The FCB, instead of the default, resides in location starting at $2000.

```
BDOS=$0005
*=$2000
FLCNBK  .BYTE $02,'TEST    ASM',$00,$00,$00
FLCNKK * = * +20  ;FCB is located at $2000-$2023
*=$0200           ;program starts at $0200
LDX  #$16         ;$16 is the function No. for open
LDY  #FLCNBK/256  ; high FCB addr to Y
LDA  #FLCNBK      ; low byte addr to A
JSR  BDOS         ; call open function
LDA  #$00         ;set 'cr' to 0
STA  FLCNBK+32
```

```
*********************************************************
*                                                       *
*  FUNCTION  23  :  CLOSE FILE                          *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $17                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned value  :                                    *
*      Register A  : $FF file not found                 *
*                    Directory code if success          *
*                                                       *
*********************************************************
```

This function will store the modified FCB addressed by registers Y

and A to the referenced disk directory. A file doesn't need to be closed if only read operation took place. But if there is some write operation, it is necessary to close this file in order to get a corrected directory entry for it in the disk. Otherwise the data just written may be lost.

Upon return, a value $FF in register A means the file was not found. Otherwise a directory code, from $00 through $6F, will be returned in register A indicating a successful operation.

```
*********************************************************
*                                                       *
*  FUNCTION  24  :  READ RECORD SEQUENTIALLY            *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters                                     *
*      Register  X  : $18                               *
*      Register  Y,A: FCB address                       *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $00 success                       *
*                     $01 end of file                   *
*                     $02 record not in disk            *
*                                                       *
*********************************************************
```

This function will read the next 128 byte record from the file , whose FCB has been actived by an open operation and place it into memory at the current DMA address. The record is read from the position 'cr' of the current extent, and after reading, the 'cr' field is incremented to the next record. If the 'cr' exceeds 64, the next extent is automatically opened and the 'cr' is set to 0 for the

subsequent read operation.

Upon return, a value $00 in register A indicates a successful read. A value of $01 means end of the file. A value of $02 means the block, which should contain the record, does not exist possibly because the file was written via the random functions and this record was never created.

The following is an example which will read a 10-record file named TEST.ASM from the current disk into the memory starting at location $4000.

```
    BDOS = $0005
    COUNT = $80       ;record counter
    DMAADS = $81      ;DMA address location
    BUFFER = $4000    ;start of the user buffer
    * = $0200
    FLCNBK  .BYTE  $00,'TEST    ASM',$00,$00,$00
    FLCNKK  * = * + 20
  RDFLBF = *
    LDX  #$16         ;open the file whose FCB is
    LDY  #FLCBNK/256  ;addressed by Y and A
    LDA  #FLCBNK
    JSR  BDOS
    LDA  #$00
    STA  FLCBNK+32    ;set 'cr' to 0
```

```
LDA  #$0A          ;set count to 10 to indicate
STA  COUNT         ;we will read 10 records
LDA  #BUFFER       ;initialize DMA
STA  DMAADS
LDA  #BUFFER/256
STA  DMAADS+1
R$FLBF = *
LDX  #$12          ;$12 is No. for set DMA
LDA  DMAADS
LDY  DMAADS + 1
JSR  BDOS
LDX  #$18          ;read current record to memory
LDY  #FLCNBK/256 ;addressed by DMAADS
LDA  #FLCNBK
JSR  BDOS
CLC                ;increment DMA by 128 bytes
LDA  #$80
ADC  DMAADS
BCC  * + 4
INC  DMAADS + 1
DEC  COUNT       ;finished?
BNE  R$FLBF      ;if not continue
RTS
```

```
*********************************************************
*                                                       *
*  FUNCTION  25  :  WRITE RECORD SEQUENTIALLY           *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $19                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $00  success                      *
*                     $01  directory overflow           *
*                     $02  disk overflow                *
*                                                       *
*********************************************************
```

This function will write the next 128 byte record from the memory at the current DMA address to the file whose FCB has been actived by an open or create file operation. The record is written in 'cr' position of the current extent. After writing ,the 'cr' field is incremented by one. If the 'cr' exceeds 64, the next extent is automatically opened and the 'cr' field is set to 0 for the subsequent write operation.

Upon return, a value $00 in register A means a successful writing. Otherwise A being $01 indicates file directory overflow and $02 indicates disk overflow.

```
*******************************************************
*                                                     *
*  FUNCTION  26  :  READ RECORD RANDOMLY              *
*                                                     *
*******************************************************
*                                                     *
*  Entry Parameters :                                 *
*      Register  X  : $1A                             *
*      Register  Y,A: FCB address                     *
*                                                     *
*  Returned Value  :                                  *
*      Register A  :  $00 success                     *
*                     $01 read unwritten record       *
*                     $03 cann't close current extent *
*                     $04 read unwritten extent       *
*                                                     *
*******************************************************
```

This function is similar to the sequential file read function except the record being read is determined by the 'r0' and 'r1' fields, instead of 'cr' field, in the FCB actived by an open file function. The 'r0' and 'r1' byte pair is treated as a value with r0 holding least significant and r1 most significant part of the record No. to be read.

In order to use this function for file access, the base extent (extent 0) must first be opened, the desired record No. must be put into the 'r0' and 'r1' properly, and then this function may be called.

During this operation, the ' x' and 'cr' are set according to the random record No. desired. Contrary to the sequential operation, upon return, the 'cr' is not advanced and still points to the record just read. Thus a subsequent sequential file read or write operation will start from the current randomly accessed position, causing re-reading or re-writing of the last record. In order to avoid the duplication,

the user should advance the 'cr' by one, if the next sequential read or write operation is needed.

Upon return, the value $00 in register A indicates a successful operation. Otherwise an unsuccessful operation has occured with the error code in the register A listed below:

$01 reading unwritten record

$03 cann't close the current extent

$04 reading an unwritten extent

```
*********************************************************
*                                                       *
*  FUNCTION  27  :  WRITE RECORD RANDOMLY               *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $1B                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $00 success                       *
*                     $02 map overflow                  *
*                     $03 cann't close current extent   *
*                     $05 directory overflow            *
*                                                       *
*********************************************************
```

This function is similar to the read random function except that data is written to the disk from the current DMA address. The same remarks as in the ' read record randomly' function apply, Care should be taken when a subsequent read or write operation is going to be performed.

Upon return, the value $00 in register A indicates a successful operation. Otherwise an unsuccessful operation has occured with error code in register A listed below:

$02 disk overflow

$03 cann't close current extent

$05 directory overflow.

```
*********************************************************
*                                                       *
*  FUNCTION  28  :  SEARCH FOR FIRST                    *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*     Register   X   : $1C                              *
*     Register   Y,A : FCB address                      *
*                                                       *
*  Returned value  :                                    *
*      Register A  :  $FF file not found                *
*                     0, 1, 2 and 3 if success          *
*                                                       *
*********************************************************
```

This function scans the directory of the referenced disk for a match with the file given by the FCB addressed by the register Y and A. Upon return,the value $FF in the register A indicates that the desired file is not found. Otherwise 0,1,2,3 is returned indicating a successful operation. If the file is found, it's directory entry along with that of 3 other entries in the same directory record are copied into the DMA. The relative starting position is determined by (A) * 32 from the beginning address of the DMA, from which one can extract the desired directory information.

An ASCII question mark ($3F) in any position from 'f1' to 'ex' matches the corresponding field of any directory entry on the referenced disk drive.

This function specifically saves the FCB address for use by the function 'search for next' (function No. 29).

```
*********************************************************
*                                                       *
*  FUNCTION  29  :  SEARCH FOR NEXT                     *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $1D                              *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $FF File not found                *
*                     0, 1, 2 and 3 if success          *
*                                                       *
*********************************************************
```

This function is similar to 'search for first' except the directory scan continues from the the last matched entry (not include the present one) and it uses the FCB address saved by the previous ' search for first '(see funtion No.28).

Similar to 'search for first', the value $FF is returned when no further directory entry is found.

```
*********************************************************
*                                                       *
*  FUNCTION  30  :  RENAME FILE                         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters :                                   *
*      Register  X  : $1E                               *
*      Register  Y,A: FCB address                       *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $FF  File not found               *
*                     $80  duplicate file name exist    *
*                     $00 - $6F  success                *
*                                                       *
*********************************************************
```

THis function uses the FCB addessed by the register Y and A to change all occurrence of the file named in the first 16 bytes to the file named in the second 16 bytes. The drive code at position 0 corresponds to the old file,while drive code for the new file at position 16 must be set to 0.

Upon return, register A is set to the directory code, form $00 - $6F, to indicate a successful operation. Otherwise the value $FF in A indicates that the old file could not be found and the value $80 in A indicates the new file name is already in the directory and the renaming was not done.

```
*********************************************************
*                                                       *
*  FUNCTION  31  :  SET FILE ATTRIBUTES                 *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters :                                   *
*      Register  X  : $1F                               *
*      Register  Y,A: FCB address                       *
*                                                       *
*  Returned Value  :                                    *
*      Register A  :  $FF  file not found               *
*                     $00 - $6F if success              *
*                                                       *
*********************************************************
```

This function allows the user program to set or reset some file attributes. In this version, only bit 7 of 't1' is used. This bit is the R/W attribute with '1' meaning read only and '0' R/W. Bit 7 of other bytes from 'f1' to 'f8' and 't2' to 't3' may be used for attributes in future system modifications or by the user.

The file name in the FCB must be unambiguous and the bit 7 of 't1' must set to the desired attribute.

Upon return, the value $FF in register A indicates the file could not be found. Otherwise a directory code, from $00 - $6F, indicates a successful operation.

The following is an example which sets the file TEST1.ASM in disk A to read/only.

```
BDOS=$0005

*=$0200

FLCNBK  .BYTE $01,'TEST1   ASM',$00,$00,$00

FLCNKK  * = * + 20

LDA #$80          ;set bit 7 of 't1' to 1

ORA FLCNBK+9

STA FLCNBK+9

LDX #$1F          ;$1F is the function No. for

                  ;setting attributs

LDY #FLCNBK/256 ;high FCB address to Y

LDA #FLCNBK       ;low FCB address to A

JSR BDOS          ;call set attributes function
```

```
*********************************************************
*                                                       *
*  FUNCTION  32  :  COMPUTE FILE SIZE                   *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $20                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*                    'r0','r1' contain file size        *
*                                                       *
*********************************************************
```

With an unambiguous file name in the FCB, this function will scan the referenced disk directory to find all the directory entries for this file and compute the total number of records belonging to it. Upon

return, the random record bytes r0 and r1 contain the 'virtural' file size with it's high byte in r1 and lower byte in r0.

Data can be appended to the end of an existing file by simply calling this fuction to set the random record position to the end of the file, then performing a sequence of random write operations starting at the record position computed by this function.

The virtural size of a file is equal to the physical size when the file is created by sequential writing operation. If the file is written in random mode and holes exists somewhere, the actual size may be smaller than the size indicated.

```
*********************************************************
*                                                       *
*  FUNCTION  33  :  SET RANDOM RECORD                   *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $21                              *
*      Register  Y,A : FCB address                      *
*                                                       *
*  Returned Value  :                                    *
*                    random record No. with r0          *
*                    holding low byte and r1            *
*                    high byte                          *
*                                                       *
*********************************************************
```

This function will automatically produce the random record position with r0 containing it's low byte and r1 containing high byte, for a file which has been read or written sequentially up to a particular point. The function can be used in two ways.

First, it is ofen needed to initially read and scan a sequential file to extract the position of various 'key' fields. As each key is encountered ,this function is called to caculate the random record position for the data corresponding to this key and the resulting record position is placed into a table with the key for later retrieval. After finishing scanning and tabularizing , one can move instantly to a particular key record by performing a random read using the corresponding random record No. which was saved in the table.

A second use is to switch from a sequential file operation to a random one. This function can set the record No. after the last sequential read or write operation, thus the subsequent random operation can continue from the selected position in the file.

```
*********************************************************
*                                                       *
*  FUNCTION  34  : SEND I/O BUFFER BACK TO DISK         *
*                                                       *
*********************************************************
*                                                       *
*  Entry Parameters  :                                  *
*      Register  X   : $22                              *
*                                                       *
*********************************************************
```

This function will send the disk I/O buffer (one track long) back to the current disk track indicated in $0044.

This function is only used by the system.

CHAPTER 6

PAGE ZERO USAGE

Page zero has a special meaning for the 6502 microprocessor. It plays an important role in the 6502 assembly language programming. In this system, CCP, BDOS and BIOS use 128 bytes, from $00 - $7F, for their special purpose. Thus there are only another 128 bytes, from $80 - $FF, left for the user. In the following sections, we will discuss each page zero location as it is used by the system.

6.1 Page zero usage for BDOS.

1. START ($00 - $02) : is a warm boot vector.
2. IOBYTE ($03) : is an I/O device vector.
3. BDSCLV ($05 - $07) : is a BDOS calling vector.
4. CREG ($08) : Working storage for BDOS.
5. DEREG ($09 - $0A) : The same as above.
6. HLREG ($0B - $0C) : The same as above.
7. HLREG1 ($0D - $0E) : The same as above.
8. FLAG ($0F) :

   Bit 3 '1' for building up checksum area.
   '0' for checking.
   Bit 4 '1' for R/W sector.
   '0' for R/W block.
   Bit 5 '1' for adding map bit.

'0' for deleting map bit.

Bit 6 '1' for writing

'0' for reading.

Bit 7 '1' for finishing DIR scaning.

'0' for not yet.

9. DIRBFA ($12 - $13) : contains the address of directory buffer for current active disk.

10. MAPBFA ($14 - $15) : contains the address of bit map for current active disk.

11. CHKBFA ($16 - $17) : contains the address of check sum area for current active disk.

12. RTNFLG ($18) : is a flag to indicate whether BDOS should change the disk back to the current disk when exiting.

13. RTNFLG + 1 ($19) : Usually is a flag to indicate if the current (BDOS) function processing is successful, or contains the low byte of the address when a return address is needed.

14. RTNFLG + 2 ($1A) : Contains the high byte of the address when a return address is needed.

15. DRERNB ($1C) : is used as directory entry No. counter during the directory scan. When the scan is completed, i.e. a match has been found, the value of the counter is used to caculate the offset of the matched entry from the base address of the directory buffer.

16. DREROF ($1D) : contains the offset mentioned above.

17. TEMP ($1F) : is a temporary storage.

18. BLKNUB ($21) : contains the relative block No. within the track.

19. TRKNUB ($22) : contains the desired track No.

20. PRVDSK ($24) : saves the current disk No. while doing a file operation on the other disk.

21. CRTDRN ($25) : saves the 'dr' of the active FCB. when the BDOS completes file operation, the value is returned to the FCB.

22. CMERCT ($27) : contains the number of bytes in the FCB to be matched with the directory entry.

23. DERDSK ($29) : contains the desired disk number (0: disk A, 1: disk B).

24. RTNFG1 ($2A) : is an auxiliary return flag used in computing values for RTNFLG + 1.

25. FLAG2 ($2C) :

Bit 7 '1' for sequential R/W.
'0' for random R/W.

BIT 6 '1' for read.
'0' for write.

Bit 5 '1' for R/W sector.
'0' for R/W block.

Bit 4 '1' for make new file.
'0' for make new extent.

BIT 3 '1' for no empty block left.

'0' for empty block found.

26. CRRCNB ($2D) : contains a copy of 'cr' in the FCB.

27. TLRCNB ($2E) : contains a copy of 'rc' in the FCB.

28. EXNTNB ($2F) : contains a copy of 'ex' in the FCB.

29. SECNUB ($30) : contains sector No. within a track.

30. BKRBLK ($31 - $32) : contains the block No. while BDOS is performing a backward scan of the bit map for an empty block.

31. ADVBLK ($33 - $34) : contains the block No. while BDOS is performing a forward scan of the bit map for an empty block.

32. BKOSFG ($35) : is a flag to indicate whether the current file location byte in the active FCB is odd or even.

33. BLKOFS ($36) : is an offset of the current location byte from the beginning of the active FCB.

34. IOBFFG ($39) : is a flag to indicate whether the I/O buffer has been written into.

35. CRTDKS ($3A) : contains the current disk No.

36. SCNPTS ($3B) : is a pointer to the starting column of the current input line used in console

I/O editing.

37. CHRNUB ($3E) : is a character No. counter for the console input buffer.

6.2 Page zero usage for BIOS

39. BLKNB1 ($42) : contains the relative block No. within a track.

40. SECNB1 ($43) : contains the relative sector No. within a block.

41. TKNDER ($44) : contains the current desired track No.

42. TRKDRC ($45) : contains the desired track No.

43. DMAREG ($46 - $47) : contains the starting address of the DMA.

44. TRKCRN ($48) : contains the current head position.

45. TKTMAX ($49) : contains the max. number of times for attempting to match the track number in case of error.

46. ERORFG ($4A) : is a error flag.

47. IOBUFA ($4C - $4D) : contains the I/O buffer address.

48. PAGNUB ($4E) : is a page No. counter used in disk I/O.

49. ERTMAX ($4F) : contains the max. number of times for attempting read/write disk in case of error.

50. CBADRV ($55 - $56) : contains current FCB address for 'search next' function.

6.3 Page zero usage for CCP

51. RWQ ($60 - $61) : is a working storage for CCP.
52. RNW ($62 - $63) : The same as above.
53. RWK ($6B - $6C) : The same as above.
54. RBC ($67 - $68) : The same as above.
55. RHL ($69 - $6A) : The same as above.
56. RDE ($6D - $6E) : The same as above.

Page zero locations $6F through $7F are reserved for future expansions of the system.

6.4 Page zero usage for the user

Memory locations from $80 throgh $FF are available to the user. If all of page zero is needed by the user, it is the user's responsibility to save page zero in another area before calling the system and restore it back when exiting.

CHAPTER 7

ERROR MESSAGES

There are three types of errors which may occor while a program is running or a command is being executed: BIOS errors, BDOS errors and CCP errors. When errors of the first two types are recognized by the system, an error message number is displayed at the console (see below). CCP errors are spelled out at the console.

BIOS errors are fatal errors, most of them relating to the disk I/O. When an error of this type occurs, the user needs to type CTRL-C to reboot the system and proceed on. If the error occurs repeatedly there may be a problem with the disk itself.

BDOS errors usually are not fatal and are likely due to programming mistakes. Correcting the program should clear things up.

CCP errors occurs when the user enters commands. Retyping the command correctly should correct such errors.

7.1 BIOS errors

ERROR No.	Meaning

$20	index hole found, while writing the disk
$21	index hole found, while reading the disk
$22	drive is not ready
$23	track No. does not match
$24	disk is write protected
$25	the I/O devices is not defined
$40	data R/W error
$80	parity error

7.2 BDOS errors

Error No.	Meaning
$02	illegel disk No.
$03	illegel function No.
$04	disk is write protected

$05 duplicate file name

$06 check sum error

$07 file is write protected

$08 file is not open

7.3 CCP errors

. 'NO FILE' means the desired file can not be found.

. 'NO SPACE' means directory or disk space overflow.

. 'FILE EXISTS' means a file already exists by the new name.

. '?' means a syntax error in the command. The '?' preceeds the portion of the command in error.

CHAPTER 8

SYSTEM UTILITY DEVELOPMENT

This system is designed to support a set of utilities such as DSKUTY, STAT, ASM, LOAD, DDT, PIP, ED, SUBMIT, DUMP etc. At present DSKUTY is the only one completely developed. DSKUTY supports disk housekeeping functions, such as disk format, disk copy, specified disk track R/W. The other utilities are to be developed in the future.

Once an assembler and editor become available, all other utilities can be installed using OUP/M operating system. Meanwhile, we present a method for installing utilities in the the OSI work enviroment under which OUP/M was developed. This enviroment consists of a 6502-based microcomputer, a dual disk drive, a console, a printer and the OSI operating system.

Step 1. Design the source program with an origin at $0200.

Step 2. Use the OSI editor to enter the source program.

Step 3. Use the OSI assembler with an offset to assemble the source into a safe location in memory. 'Safe' depends on the length of the object code.

Step 4. Save the object code on the OSI system disk.

Step 5. Load the object code from the disk into memory starting at $0200. Note that the object code must fit between $0200 and $2200.

Step 6. Boot OUP/M and use the SAVE command to store the object code on the OUP/M system disk.

The following is an annotated example to illustrate these steps:

console display	comments
H/D/M ? D	press shift-D to boot OSI system.
A* AS	enter AS to load the editor and assembler.
OSI 6502 ASSEMBLER	
COPYRIGHT 1976 BY OSI	
. I	type I and respond Y to
INIZ ? Y	initialize the workspace.
. 010 ; say hello at the console	enter the source program under the OSI editor.
. 020 * = $0200	set oringin at $0200.
. 030 JMP MAIN	
. 040 BUFF .BYTE 'HELLO$'	

```
. 050       BDOS = $05
. 060       MAIN = *
. 070       LDX  #$09
. 080       LDA  #BUFF
. 090       LDY  #BUFF/256
. 100       JSR  BDOS
. 110       RTS
. 120       .END
. A                                   get listing of the program
 10 ;say hello at the console
 20 0200          JMP MAIN
 30 0203          BUFF .BYTE 'HELLO$'
 40 0204 45
 30 0205 4C
 30 0206 4C
 30 0207 24
 40 0005=         BDOS=$05
 50 0206=         MAIN = *
 60 0208 A209     LDX #$09
 70 020A A903     LDA #BUFF
 80 020C A002     LDY #BUFF/256
 90 020E 200500   JSR BDOS
100 0211 60       RTS
110               .END

. MD000                               set memory offset.
```

```
. A3                          store object code at
                              locations staring at $D200.
. EXIT                        exit to operating system.
. A* SAVE 75,1=D200/1         save object code of 1 page
                              onto track No. 75.
. A* CALL 0200=75,1           load object code to
                              locations starting at $0200.

H/D/M ? D                     push 'reset' and press
                              shift-D to boot OUP/M
                              operating system.

OUP/M   OPERATING   SYSTEM
DIR REM ERA SAVE TYPE USER

>A SAVE 01 HELLO             save the object code under
                              the file name 'HELLO'.
```

CHAPTER 9

DISK READ / WRITE UTILITY

A disk read/write utility is the first utility installed in the system. It will do disk housekeeping such as disk copy, disk format and a specified track read/write.

To invoke this utility, the user should type 'DSKUTY' followed by a carriage return in respond to the CCP prompt 'A>' or 'B>'.

After the user presses 'cr', a main menu will be dispalyed on the screen as follows:

```
-- DISKETTE UTILITY --

SELECT ONE

1) COPY ALL TRACKS

2) COPY OPTION TRACK(S)

3) INITIALIZE OPTION TRACK(S) ( TRACK 0 IS NOT ALLOWED )

4) READ OPTION TRACK(S)
```

5) WRITE OPTION TRACK(S)

6) EXIT TO OUP/M

-- TRACK NUMBER: A 2-DIGIT DECIMAL VALUE

-- ADDRESS & VECTOR: A 4-DIGIT HEXIDECIMAL VALUE

-- PAGE NUMBER: A SINGLE HEXIDECIMAL

After entering a digit, futher prompts will be displayed to obtain additional information from the user.

With these options listed above, back-up copies may be produced and a modified object code of the system itself may be saved or installed on the track No.0 through No.2.

This utility is a limited substitute for the CP/M system generation utility. The user can modify the source program and reassemble it or patch the object code to get a new version of OUP/M.

Appendix B : Listing of Source Programs

B1 : Listing of COLDLD

B2 : Listing of CCP

B3 : Listing of BDOS

B4 : Listing of BIOS

B5 : Listing of DSKUTY

Appendix B1 : Listing of COLDLD

```
10        ;OUP/M COLDLD WRITTEN BY SHAO,JIAN-XIONG
20        ;ON JULY 1982.
30        *=$2200
40        JMP      CDBTLD
50        BIOSA1   .WORD  $2280
60        BIOSA2    .WORD  $DE00
70        BIOSPG   .BYTE $0C
80        MVREG1=$53
90        MVREG2=$55
100       IOBFFG=$39
110       TKNDER=$44
120       TRKCRN=$48
130       DVACIA=$C010
140       FRYCST=$E41B
150       WBOOT=$E477
160       BOOT=$E42F
170       SFTSWT=$F701
180       FRYCCT   .BYTE   $31,$31,$31,$31,$31
190                .BYTE   $62,$62,$62,$62
200                .BYTE   $A0,$A0,$A0,$A0
210                .BYTE   $A0,$A0,$A0,$A0
220 ;
230       CDBTLD=*                ;COLD BOOT LOADER
240 ;
250       CLD                     ;MOVE BIOS AND PART OF
260       LDA      BIOSA1         ;BDOS TO ITS REGULAR POSITION
270       STA      MVREG1
280       LDA      BIOSA1+1
290       STA      MVREG1+1
300       LDA      BIOSA2
310       STA      MVREG2
320       LDA      BIOSA2+1
330       STA      MVREG2+1
340       LDX      BIOSPG
350       LDY      #$00
360       C$BTLD=*
370       LDA      (MVREG1),Y
380       STA      (MVREG2),Y
390       INY
400       BNE      C$BTLD
410       INC      MVREG1+1
420       INC      MVREG2+1
430       DEX
440       BNE      C$BTLD
450       LDA      #$34     ;ADJUST FREQUENCY
460       STA      SFTSWT
470       LDY      #$00
480       LDA      #$03
490       STA      DVACIA
500       LDA      #$38
510       STA      DVACIA
520       STA      DVACIA+1
```

```
530       PHA
540       PLA
550       PHA
560       PLA
570       PHA
580       PLA
590       STX     DVACIA+1
600       AD$TFY=*
610       LDA     DVACIA
620       BMI     A$JTFY
630       INY
640       BNE     AD$TFY
650       A$JTFY=*
660       LDA     FRYCCT,Y
670       STA     FRYCST
680       JSR     BOOT    ;INITIALIZE ALL INTERFACE
690       LDA     #$00
700       STA     IOBFFG
710       STA     TKNDER  ;MARK HEAD POSITION TO 0
720       STA     TRKCRN
730       LDA     #$FF    ;SET COLD BOOT FLAG
740       TAX
750       TXS
760       PHA
770       JMP     WBOOT   ;WBOOT CCP AND REST OF BDOS
```

Appendix B2 : Listing of CCP

```
780          ;OUP/M CCP WRITTEN BY LIO,QI-WEN
790          ;ON AUG, 1982
800         *=$D000
810         JMP     MAIN
820          ;6502 CP/M CCP
830          ;***********************************************
840          STACKB=$FF
850         IBUBCC=$EE80       ;MAX CHAR ADDRESS
860         IBUBC=$EE81       ;BUFF COUNTER
870         IBUFB=$EE82      ;FIRST CHAR IN BUFF
880         IBUFBA=$EF02
890         IBUFBB=$EF04
900 IBUFC    .WORD IBUBC
910 IBUFF    .WORD IBUFB
920         IBUFPA   .WORD IBUFB ;PTR,INIT IBUFB
930 IBUFPB   .WORD IBUFBB
940          FCBB=$EE01
950          FCBBB=FCBB+1
960          FCBBC=FCBB+$10
970          FCBBD=FCBB+$20
980          RETVAL=FCBB+$23
990 FCBP     .WORD FCBB
1000 FCBPB   .WORD FCBBB
1010 FCBPC   .WORD FCBBC
1020 FCBBP   .WORD FCBBD
1030         DSKNUM=FCBB+$42        ;POINTES THE NUMBER OF
1040         WRKPLA=FCBB+$43      ;WORK ELEMENT
1050         WRKPLB=FCBB+$44
1060         WRKPLC=FCBB+$45
1070         WRKPLD=FCBB+$46
1080 WRKPPB .WORD WRKPLB
1090        RBC=$0067
1100        RHL=$0069
1110       RDE=$006D        ;REPLACE Z80 DE,HL,BC
1120         RWK=$006B
1130       RWQ=$0060
1140       RNW=$0062
1150         DMAD=$E965
1160 SUBTBL .WORD SUBPGO
1170         STRTD=$0200
1180 SUBPGO .WORD $2710   ;10,000
1190          .WORD $03E8   ;1,000
1200          .WORD $0064   ;100
1210          .WORD $000A   ;10
1220 MSGAD1   .WORD MSGVER
1230 CMDTAB .BYTE 'DIR ERA TYPESAVEREN USER'
1240;
1250;
1260 TABADD .WORD CMDTAB
1270 ADRTAB   .WORD PDIR,PERA,PTYPE,PSAVE,PREN
1280          .WORD PUSER,PTRANS
1290 TBLAD2 .WORD ADRTAB
```

```
1300 RDERRA  .WORD RDERR
1310 NOFERA  .WORD NOFERR
1320 MSGAD3  .WORD MSGAD1
1330       BDOS=$D93F
1340 MSGAD5  .WORD MSGAD4
1350 MSGAD7  .WORD MSGAD6
1360 MSGDD9  .WORD MSGDD8
1370 MSGBA2  .WORD MSGBA1
1380   ;*********************************************
1390   ;SUBROUTINE
1400   ;*********************************************
1410 CHROUT STA RDE      ;OUTPUT CHAR TO E
1420        LDX #$02
1430        JMP BDOS     ;PRINT A CHARACTER
1440 CHROB  STA WRKPLC    ;FOR SAVING RBC
1450        LDA RBC
1460        STA RWQ
1470        LDA RBC+1
1480        STA RWQ+1
1490        LDA WRKPLC
1500        JSR CHROUT   ;TO PRINT A CHAR
1510        LDA RWQ       ;FOR RESTORE RBC
1520        STA RBC
1530        LDA RWQ+1
1540        STA RBC+1
1550        RTS
1560 CRLF   LDA #$0D     ;'CR' TO A
1570        JSR CHROB    ;OUTPUT 'CR'
1580        LDA #$0A     ;'LF' TO A
1590        JMP CHROB    ;OUTPUT 'LF'
1600 SPACE  LDA #$20      ;' ' TO A
1610        JMP CHROB
1620   ;PRINT STRING UNTIL 'NUL',MESSAGE POINTER IN RDE
1630   ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1640 STRIOA  JSR CRLF     ;OUTPUT 'CR' 'LF'
1650        LDA RBC       ;GET MESSAGE POINTER
1660        STA RHL
1670        LDA RBC+1
1680        STA RHL+1     ;NOW RHL POINTES MESSAGE
1690   ;PRINT STRING ,MESSAGE IN RHL
1700   ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1710 STRIOB LDY #$00      ;CLEAR Y
1720        LDA (RHL),Y   ;GET A CHAR
1730        BNE LOOP1     ;CHAR='NUL'?
1740        RTS           ;YES,END PRINT AND RETURN
1750 LOOP1  INY           ;NO,
1760         STY RWQ
1770        JSR CHROUT    ;OUTPUT A CHAR
1780         LDY RWQ
1790        JMP STRIOB+2 ;CONTINUE
1800 RSTDSK  LDX #$0A     ;RESET  DISK OPERATION
1810;
1820;
1830;
```

```
1840;
1850;
1860;
1870         JMP BDOS
1880 SELDSK LDX #$13      ;SELECT DISK OPERATION
1890         JMP BDOS
1900   ;THE FOLLOWING 8 ROUTINES CALL BDOS THEN
1910   ;RETURN VALUE TO RETVAL ,A+1RETURN
1920   ;*********************************************
1930 CDOSRT LDA RDE
1940         LDY RDE+1
1950 CDOSR1   JSR  BDOS
1960         STA RETVAL  ;RETURN VALUE TO 'RETVAL'
1970         CLC
1980         ADC #$01    ;A+1,THEN RETURN
1990         RTS
2000 OPNFLB LDX #$16  ;OPEN DISK FILE
2010         JMP CDOSRT
2020 OPNFL  LDA #$00
2030      .  STA FCBBD     ;CLEAR SOME PLACE OF FCB
2040         JSR FCBTOD    ;FCBP TO RDE
2050         JMP OPNFLB    ;TO OPEN FILE
2060 CLSFIL LDX #$17      ;CLOSE FILE
2070         JMP CDOSRT  ;
2080 SERFST LDX #$1C      ;SEARCH FOR FIRST
2090       JMP      CDOSRT
2100 SERNXT LDX #$1D     ;SEARH FOR THE NEXT
2110         JMP CDOSR1
2120 SERFIL JSR FCBTOD  ;FCBP TO RDE
2130         JMP SERFST
2140 FCBTOD LDA FCBP     ;FOR FCB TO RDE
2150         STA RDE
2160         LDA FCBP+1
2170         STA RDE+1
2180         RTS
2190 DELFIL LDX #$15     ;DELETE FILE
2200   ;THE FOLLOWING ROUTINE CALLING CDOSB
2210   ;RETURN ZERO IF SUCCESS
2220   ;------------------------------------
2230         LDY #$21
2240 CDOSB  LDA RDE
2250         LDY RDE+1
2260         JSR BDOS
2270         CLC
2280         RTS           ;RETURN VALUE<>0 IF FALSE
2290 RDSEQB LDX #$18     ;READ SEQUENTIAL
2300         JMP CDOSB
2310 RDSEQ  JSR FCBTOD  ;FCB TO RDE
2320         JMP RDSEQB  ;TO READ SEQUENTIAL
2330 WRSEQ  LDX #$19     ;WRITE SEQUENTIAL
2340         JMP CDOSB
2350 MAKFIL LDX #$14     ;MAKE FILE
2360         JMP CDOSRT  ;RETURN VALUE TO 'RETVAL'
2370 RENFIL LDA RDE
```

```
2380        LDY RDE+1
2390        LDX #$1E     ;RENAME FILE
2400        JMP BDOS
2410 GETCOD LDA #$FF
2420        STA RDE      ;$FF TO E
2430 SETCOD LDX  #$11    ;SET/GET ESER CODE
2440        JMP BDOS     ;RETURN USER CODE IF 'GET'
2450;  PRINT PAGE
2460;  ------------
2470 PRNPG1 JSR SPACE
2480        JSR SPACE
2490        LDY #$21      ;PAGE POSITION
2500        LDA (RHL),Y   ;GET PAGE NUMB.
2510        STA FCBB+$25 ;SAVE
2520        INY
2530        LDA (RHL),Y   ;GET PAGE MSB
2540        STA FCBB+$26 ;SAVE
2550        CLC
2560        ROR FCBB+$26
2570        ROR FCBB+$25
2580 PRNPG2 LDY #06      ;IF PAGE 000-999
2590 NXTDIG LDX #00      ;INIT DIGIT COUNT
2600 SUBEM  LDA FCBB+$25 ;FETCH LSBY
2610        SEC
2620        SBC SUBTBL,Y  ;-LSBT OF TAB
2630        STA FCBB+$25 ;RETURN TO MEMORY
2640        LDA FCBB+$26 ;FETCH MSBY
2650        INY
2660        SBC SUBTBL,Y ;-MSBY OF TAB
2670        BCC ADBACK   ;IF RESULT IS '-'
2680        STA FCBB+$26 ;NO
2690        INX
2700        DEY          ;PTR LSBY IN TABLE
2710        JMP SUBEM    ;LOOP
2720 ADBACK DEY          ;PTR LSBY IN TABLE
2730        LDA FCBB+$25 ;FETCH LSBY
2740        ADC SUBTBL,Y ;+LSBY OF TAB
2750        STA FCBB+$25
2760        TXA          ;DIGIT COUNT TO A
2770        ORA #$30     ;CONVERT TO ASCII
2780        STY RNW      ;SAVE
2790        JSR CHROUT   ;OUT DIGIT
2800        LDY RNW      ;RESTORE Y
2810        INY
2820;
2830        INY          ;PTR TO NEXT TABLE
2840;
2850        CPY #$0A
2860;
2870        BCC NXTDIG   ;LOOP
2880        LDA FCBB+$25
2890        ORA DSKNUM
2900        ORA #$30     ;CONVERT TO ASCII
2910        JMP CHROUT   ;PRINT REMAINDER
```

```
2920   ;CHANGE LOWER CASE TO UPPER CASE
2930   ;---------------------------------------
2940 CUPCAS CMP #$61     ;CHAR>=LOWER CASE A ?
2950        BCS LOPU1    ;NO,
2960        RTS          ;YES,RETURN
2970 LOPU1  CMP #$7C     ;CHAR<LOWER CASE Z+1?
2980        BCC LOPU2    ;NO,GO TO LOPU2
2990        RTS          ;YES,RETURN
3000 LOPU2  AND #$5F     ;LOWER CASE TO UPPER CASE
3010        RTS
3020   ;COLD START,($01FF)=0:ROUTINE SHOW OP VERSION
3030   ;WARM START,($01FF)<>0:ROUTINE HANDLE IN BUFF
3040   ;---------------------------------------------
3050 INBUFL LDA $01FF  ;'COLD START'?
3060        CMP #$FF
3070        BNE MJN      ;NO,
3080        JSR CRLF
3090        LDA MSGAD1
3100        STA RHL
3110        LDA MSGAD1+1  ;P
3120        STA RHL+1    ;RHL POINTES 'CP/M VERSION--'
3130        JSR STRIOB   ;PRINT MESSAGE STRING
3140        JSR ENDINB   ;YES TO FILL 0 TO $01FF
3150  MJN   JMP START
3160 INBUFA LDA #$80     ;BUFF SIZE
3170        STA IBUBCC   ;SAVE BUFF SIZE
3180        LDA #IBUBCC ;BUFF ADDRESS
3190        LDY #IBUBCC/256
3200        LDX #$06     ;INPUT BUFF LINE
3210        JSR BDOS
3220        LDA #IBUBC
3230        STA RHL
3240        LDA #IBUBC/256
3250        STA RHL+1    ;RHL POINTES INPUT BUFFER
3260        LDX IBUBC    ;X AS COUNTER FOR NUMB OF CHAR
3270        LDY #$00
3280 LOPIB1 INC RHL
3290        TXA
3300        BEQ BUFEND   ;NUMB.OF CHAR=0,BRANCH
3310        LDA (RHL),Y ;NO,GET A CHAR
3320        JSR CUPCAS   ;LOWER CASE TO UPPER CASE
3330        STA (RHL),Y ;SAVE UPPER CHAR
3340        DEX          ;COUNTER-1=0?
3350        JMP LOPIB1   ;CONTINUE
3360 BUFEND STA (RHL),Y ;0 TO (RHL)
3370        LDA IBUFF
3380        STA IBUFFA
3390        LDA IBUFF+1
3400        STA IBUFFA+1 ;IBUFFA POINTES HEAD OF BUFF
3410        RTS
3420 CSTATD LDX #$09
3430        JSR BDOS
3440        PHA
3450        PLA          ;FLAGE=0?
```

```
3460        BNE LOPCS1    ;NO,BREANCH
3470        RTS           ;YES,RETURN
3480 LOPCS1 LDX #$01      ;CONSOLE INPUT OPERATION
3490        JSR BDOS
3500        PHA
3510        PLA           ;FLAGE Z=0?
3520        RTS
3530 RETDKN LDX #$0C      ;RETURN CURRENT DISK
3540        JMP BDOS
3550 STDMAI LDA #DMAD
3560        STA RDE
3570        LDA #DMAD/256
3580        STA RDE+1     ;RDE POINTES $0080BUF
3590 SETDMA LDX #$12      ;SET DMA OPERATION
3600        LDA RDE
3610        LDY RDE+1
3620        JMP BDOS
3630 ENDINB LDA #$00
3640        STA $01FF     ;0 TO STACK
3650        LDA DSKNUM
3660        JSR SELDSK    ;SELECT CURRENT DISK
3670        RTS
3680 CMDEND JSR CRLF      ;OUTPUT 'CR''LF'
3690        LDA IBUFPB    ;IN BUFF TO RWK REF'3120'
3700        STA RWK
3710        LDA IBUFPB+1  ;
3720        STA RWK+1     ;RWK POINTES IN BUFF
3730        LDY #$00
3740 LOPCN1 LDA (RWK),Y   ;GET A CHAR
3750        CMP #$20      ;CHAR=' '?
3760        BEQ ERENDC    ;YES
3770        PHA
3780        PLA           ;CHAR='0'
3790        BEQ ERENDC    ;Y,
3800        JSR CHROUT    ;OUTPUT CHAR
3810        INC RWK
3820        JMP LOPCN1    ;CONTINUE
3830 ERENDC LDA #$3F      ;'?' TO A
3840        JSR CHROUT    ;OUTPUT '?'
3850        JSR CRLF      ;OUTPUT 'CR' 'LF'
3860        JSR ENDINB    ;TO END COMMAND
3870        JMP START
3880   ;CHECK SPECIAL CHARACTER
3890   ;*************************************************
3900 CHARCK JSR RDRDE     ;READ (RED)
3910        CMP #$00
3920        BEQ CHEND     ;RETURN IF CHAR='NUL'
3930        CMP #$20      ;NO,CHECK CHAR=' '?
3940        BEQ CHEND     ;YES,RETURN
3950        BCC CMDEND    ;GO 'ERROR-HANDLE' IF CHAR<' '
3960        CMP #$3D      ;CHAR='='?
3970        BEQ CHEND     ;YES,RETURN Z=0
3980        CMP #$5F      ;CHAR='<-'?
3990        BEQ CHEND
```

```
4000         CMP #$2E      ;CHAR=','?
4010         BEQ CHEND
4020         CMP #$3A      ;CHAR=':'?
4030         BEQ CHEND
4040         CMP #$3B      ;CHAR=';'?
4050         BEQ CHEND
4060         CMP #$3C      ;CHAR='<'?
4070         BEQ CHEND
4080         CMP #$3E      ;CHAR='>'?
4090         BEQ CHEND
4100 CHEND   RTS
4110 RDRDE   STY WRKPLD+1 ;SAVE Y
4120         LDY #$00
4130         LDA (RDE),Y  ;GET A CHAR
4140         LDY WRKPLD+1 ;RESTORE
4150         RTS
4160   ;SKIP ' '
4170   ;*********************************************
4180 SKPBLK  JSR RDRDE     ;GET A CHAR.
4190         CMP #$00
4200         BEQ KPND      ;'BUFF-END'
4210         CMP #$20      ;CHAR=' '?
4220         BNE KPND      ;NO,RETURN
4230         INC RDE    ; GET NEXT CHAR
4240         JMP SKPBLK ;LOOP
4250 KPND    RTS
4260   ;*********************************************
4270   ;FILL FCB FROM FCB'POINTER'+(A)
4280   ;*********************************************
4290 FIFCBM  CLC
4300         ADC RHL       ;(A)+(L)
4310         STA RHL       ;TO (L)
4320         BCC FIEND     ;RETURN IF FLAG C=0
4330         INC RHL+1     ;NO,(H)+1
4340 FIEND   RTS
4350   ;FILL FCB FROM BEGINNING OF FCB
4360   ;*********************************************
4370 FIFCBB  LDA #$00
4380 FIFCB   PHA
4390         LDA FCBP
4400         STA RHL
4410         LDA FCBP+1
4420         STA RHL+1     ;RHL POINTES FCB
4430         PLA
4440         JSR FIFCBM    ;(RHL)+(A) TO RHL
4450         LDA RHL
4460         STA RWK
4470         LDA RHL+1
4480         STA RWK+1     ;SAVE INITIAL VALUE OF RHL
4490         LDA #$00
4500         STA WRKPLA    ;CLEAR
4510         STA WRKPLD    ;'WRKPLD'AS COUNTER FOR RDE
4520         LDA IBUFPA
4530         STA RDE
```

```
4540        LDA IBUFPA+1
4550        STA RDE+1     ;RDE POINTES INPUT BUFF
4560        JSR SKPBLK    ;TO SKIP ' '
4570        LDA RDE
4580        STA IBUFPB
4590        LDA RDE+1
4600        STA IBUFPB+1 ;SAVE NEW POINTER IN IBUFPB
4610        LDY #$00
4620        INC RDE       ;PRT 2 CHAR
4630        JSR RDRDE     ;GET 2'S CHAR
4640        CMP #$3A      ;':'?
4650        BEQ FIDKNN    ;YES,FILL DSKNUMB
4660        LDA #$00
4670        STA (RWK),Y   ;FILL PCB
4680        LDA DSKNUM    ;CURRENT DISK
4690        STA WRKPLA    ;SAVE REQ.DSKNUM
4700        DEC RDE       ;RET 1'S CHAR IN BUFF
4710        JMP FINAME
4720 FIDKNN DEC RDE       ;PTR 1'S CHAR IN BUFF
4730        JSR RDRDE
4740        SEC
4750        SBC #$40
4760        STA RWQ
4770        STA WRKPLA    ;SAVE REQ.DSKNUM
4780        DEC WRKPLA
4790 MEDJN  STA (RWK),Y   ;FILL 'D'TO PCB
4800        INC RDE
4810        INC RDE       ;PTR 3'CHAR IN BUFF
4820;
4830   ;FILL 'NAME' IN FCB
4840   ;*************************************
4850 FINAME LDX #$08      ;X AS COUNTER FOR 8 CHAR
4860        LDA #$20
4870        STA RNW       ;INIT ' '
4880 LOPFI1 JSR CHARCK    ;CHECK SPECIALCHAR
4890        BEQ FIBLK1    ;TO FILL ' 'IF RET Z=0
4900        INY
4910        CMP #$2A      ;CHAR='*'?
4920        BNE FICHAR    ;NO,TO FILL CHAR
4930        LDA #$3F      ;YES,'?'TO A
4940        STA (RWK),Y
4950        STA RNW
4960        JMP FINEXT
4970 FICHAR STA (RWK),Y   ;A CHAR TO FCB
4980        LDA #$20
4990        STA RNW
5000 FINEXT INC RDE       ;POINTES NEXT CHAR
5010        DEX           ;COUNT X-1=0?
5020        BNE LOPFI1    ;NO,LOOP
5030;
5040   ;TO FIND THE END OF NAME
5050   ;*************************************
5060 FINDSG JSR CHARCK     ;TO CHECK SPECIAL CHAR
5070        BEQ FITYP     ;TO FILL TYPE IF RET Z=0
```

```
5080        INC RDE
5090        JMP FINDSG    ;LOOP
5100;
5110   ;TO FILL ' ' IN FCB
5120   ;*********************************
5130 FIBLK1 PHA
5140 FIBLK  INY
5150        LDA RNW       ;' ' OR ? TO A
5160        STA (RWK),Y   ;' ' TO FCB
5170        DEX           ;COUNTER X-1=0?
5180        BNE FIBLK     ;LOOP
5190        PLA
5200   ;TO FILL TYPE IN FCB
5210   ;*****************************
5220 FITYP  LDX #$20
5230        STX RNW
5240        LDX #$03      ;X AS COUNTER FOR 3 CHAR
5250        CMP #$2E      ;CHAR='>'?
5260        BNE FIBLKT    ;NO,FILL ' 'IN FBC
5270        INC RDE       ;RDE POINTES NEXT CHAR
5280 LOPTP  JSR CHARCK    ;CHECK SPECIAL CHAR
5290        BEQ FIBLKT    ;TO FILL ' 'IF RET Z=0
5300        INY
5310        CMP #$2A      ;CHAR='*'?
5320        BNE FILLCH    ;NO,TO FILL CHAR
5330        LDA #$3F      ;'?' TO A
5340        STA (RWK),Y   ;'?'AS TYPE TO FCB
5350        STA RNW
5360        INC RDE      ;PTR NEXT CHAR
5370        JMP NEXTP     ;TO FILL NEXT CHAR
5380 FILLCH STA (RWK),Y   ;CHAR TO FCB
5390        LDA #$20
5400        STA RNW
5410        INC RDE       ;RDE POINTER NEXT CHAR
5420 NEXTP  DEX           ;COUNTER X-1=0?
5430        BNE LOPTP     ;NO,LOOP
5440   ;TO FIND THE END OF COMMAND
5450   ;*********************************
5460;
5470;
5480;
5490;
5500;
5510;
5520;
5530;
5540;
5550 BLNKJ1 JMP FIND5
5560 FNDEDG JSR CHARCK    ;TO CHECK SPECIAL CHAR.
5570        BNE FIND5     ;IF RETURN Z=0
5580        INC RDE       ;POINTES NEXT
5590        JMP FNDEDG    ;LOOP
5600 FIBLKT INY
5610        LDA RNW       ,' ' OR ? TO A
```

```
5620         STA (RWK),Y  ;' 'TO FCB
5630         DEX          ;COUNTER X-1=0?
5640         BNE FIBLKT   ;NO,LOOP
5650 FIND5   LDX #$03     ;X AS COUNTER FOR 3'NUL'
5660 LOPFEN  INY
5670         LDA #$00     ;'NUL' TO A
5680         STA (RWK),Y  ;TO FCB
5690         DEX          ;COUNTER X-1=0?
5700         BNE LOPFEN   ;NO,LOOP
5710         LDA RDE
5720         STA IBUFPA
5730         LDA RDE+1
5740         STA IBUFPA+1 ;SAVE INPUT BUFFER
5750 CKQUEN  LDY #$00
5760         STY RBC
5770         LDX #$0B     ;X AS COUNTER FOR 11CHAR
5780   ;COMPUTE '?'NUMBER IN FCB
5790   ;*********************************
5800 CHKQUE  INY          ;SCAN FCB
5810         LDA (RWK),Y  ;GET A CHAR FROM FCB
5820         CMP #$3F     ;CHAR='?'?
5830         BNE NEXCH    ;NO,GO TO NEXT CHER
5840         INC RBC      ;C AS COUNTER FOR '?'NUMBER
5850 NEXCH   DEX          ;COUNTER X-1=0?
5860         BNE CHKQUE   ;NO,LOOP
5870         LDA RBC
5880         RTS
5890   ;COMPARE THE COMMAND IN FCB WITH NAME TABLE
5900   ;RETURN A=0 TO 6 WHICH POINTES COMMAND POSITION
5910   ;*********************************************
5920 CMDCMP  LDA TABADD
5930         STA RWK
5940         LDA TABADD+1
5950         STA RWK+1    ;RWK POINTES COMMAND TABLE
5960         LDY #$00
5970         LDX #$00     ;X AS COUNTER FOR 6 COMMANDS
5980 LOPCM   TXA
5990         CMP #$06     ;COUNTER X>=6?
6000         BCS CMPEND   ;YES,JUMP
6010         LDA FCBPB
6020         STA RDE
6030         LDA FCBPB+1
6040         STA RDE+1    ;RDE PTR COMMAND NAME
6050         LDA #$04
6060         STA RHL      ;HL AS COUNTER FOR COMPARE 4 C
6070 CMPBGN  JSR RDRDE    ;GET A CHAR FROM FCB
6080         CMP (RWK),Y  ;EQUAL?
6090         BNE CHGNX1   ;NO,TO CHANGE NEXT COMMAND
6100         INC RDE      ;YES
6110         INY
6120         DEC RHL      ;COUNTER-1=0?
6130         BNE CMPBGN   ;NO,CONTINUE
6140         JSR RDRDE    ;YES,4 CHAR EQUAL
6150         CMP #$20     ;' '?
```

```
6160        BNE CHGNX2    ;NO,COMPARE AGAIN
6170;       LDA RDE
6180;       STA IBUFPA
6190;       LDA RDE+1
6200;       STA IBUFPA+1 ;PTR END OF COMMAND NAME??
6210        TXA           ;YES,GET POSITION OF TABLE
6220 CMPEND RTS
6230 CHGNX1 INY
6240        DEC RHL       ;#4COUNTER -1=0?
6250        BNE CHGNX1    ;NO,LOOP
6260 CHGNX2 INX
6270        JMP LOPCM     ;CONTINUE
6280   ;SEPARATE USER CODE WITH DISK NUMB.
6290   ;SET USER CODE,SELECT CURRENT DISK
6300   ;*************************************
6310 MAIN   PHA
6320        JSR RSTDSK    ;RESET DISK SYSTEM
6330        PLA
6340        STA DSKNUM    ;CURRENT DISK TO 'DSKNUM'
6350        JSR SELDSK    ;TO SELECT CURRENT DISK
6360        JMP INBUFL    ;VERSION SHOW?
6370   ;*********************************************
6380   ;START AFTER ENDING EACH CONSOLE COMMAND
6390   ;*********************************************
6400 START  LDX #$22
6410        JSR BDOS
6420        LDX #$FE
6430        TXS           ;$FE TO SP
6440        LDA DSKNUM
6450        JSR SELDSK
6460        JSR CRLF      ;OUTPUT 'CR''LF'
6470        JSR RETDKN    ;RETURN CURRENT DISK
6480        CLC
6490        ADC #$41      ;CHANGE DISK NUMB.TO CHA
6500        JSR CHROUT    ;OUTPUT 'A' OR 'B'- -
6510        LDA #$3E      ;'>'TO A
6520        JSR CHROUT    ;OUTPUT '>'
6530        JSR INBUFA    ;RECEIVE MESSAGE FROM CONSOLE
6540        LDA IBUBC+$2 ;2'S CHAR IN BUFF
6550        CMP #$3A      ;':'?
6560        BEQ BPSDK
6570 ENTRYA LDA #DMAD
6580        STA RDE
6590        LDA #DMAD/256
6600        STA RDE+1     ;RDE POINTES DMA ADDRESS
6610        JSR SETDMA    ;SET DMA
6620        JSR FIFCBB    ;FILL COMMAND IN FCB,RET '?'CO
6630        BNE CMDND1    ;IF RET'?'COUNT<>0,ERR COMMAND
6640        JSR CMDCMP    ;TO COMPARE COMMAND
6650   ;BASED ON TAB POSITION RETURNED BY 'CMDCMP'
6660   ;,PROGRAM JUMP TO DIFFERENCE ENTRY
6670   ;*********************************************
6680        STA RWK       ;SAVE RETURN VALUE
6690        LDA TBLAD2    ;ENTRY ADDR.TABLE TO  A
```

```
6700        STA RHL
6710        LDA TBLAD2+1 ;
6720;
6730;
6740;
6750;
6760        STA RHL+1     ;RHL POINTES ENTRY ADDR.TABLE
6770        LDA RWK
6780        CLC
6790        ADC RWK       ;(RETURN VALUE)*2
6800        ADC RHL       ;
6810        STA RHL
6820        LDA #$00      ;REMAIN FLAGE C
6830        ADC RHL+1
6840        STA RHL+1     ;TABL.HEAD+(RET.VAL.)*2
6850        LDY #$00
6860        LDA (RHL),Y
6870        STA RWK
6880        INY
6890        LDA (RHL),Y
6900        STA RWK+1
6910        JMP (RWK)     ;TO DIFFERENCE ENTRY
6920 CMDND1 JMP CMDEND    ;END ERR.COMMAND
6930 BPSDK  LDA IBUBC+$3   ;3'S CHAR
6940        CMP #$00       ;'00'?
6950        BNE ENTRYA     ;NOT A: OR B:COMMAND
6960        JMP PSDK       ;A: OR B: COMMAND
6970   ;SOME LOCAL SUBROUTINES
6980   ;************************
6990   ;OUTPUT ERROR MESSAGE
7000   ;~~~~~~~~~~~~~~~~~~~~~
7010 PRTER1 LDA RDERRA    ;MESSAGE ADDR.TO A
7020        STA RBC
7030        LDA RDERRA+1
7040        STA RBC+1     ;RBC POINTES 'MESSAGE'
7050        JMP STRIOA    ;OUTPUT 'DEAD ERROR'
7060 PRTER2 LDA NOFERA    ;MESSAGE ADDR. TO A
7070        STA RBC
7080        LDA NOFERA+1
7090        STA RBC+1     ;RBC POINTES 'MESSAGE'
7100        JMP STRIOA    ;OUTPUT 'NO FILE'
7110   ;'DIGIT' HANDLE FOR PAGE & USER CODE
7120   ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
7130 DIGHD  JSR FIFCBB    ;FILL 'DIGIT' IN FCB
7140;       LDA WRKPLA
7150;       BNE DIGEN1    ;ERR.END
7160        LDA #$00
7170        STA RWQ       ;RWQ FOR SAVING HIGH BITS
7180        LDA FCBPB     ;
7190        STA RHL
7200        LDA FCBPB+1
7210        STA RHL+1     ;RHL POINTES FCB
7220        LDY #$00
7230        LDX #$0B      ;X AS COUNTER FOR 11 CHAR.
```

```
7240 LOPDIG LDA (RHL),Y  ;GET A CHAR
7250        CMP #$20     ;CHAR=' '?
7260        BEQ SKPEN1   ;YES,JUMP TO END
7270        INY          ;NO,
7280        SEC
7290        SBC #$30     ;CHAR-'0'BIAS
7300        CMP #$0A     ;CHAR<=9?
7310        BCS DIGEN1   ;NO, END ERR.COMMAND
7320        STA RWK      ;SAVE LOWER BITS
7330        LDA RWQ      ;HIGH BITS TO A
7340        AND #$E0
7350        BNE DIGEN1   ;TO END ERR.COMMAND IF A<>0
7360        LDA RWQ
7370        CLC
7380        ROR A
7390        ROR A
7400        ROR A
7410        ROR A
7420        ROR A
7430        ROR A        ;(A)LEFT SHIFT 4
7440        ADC RWQ      ;OVERFLOW?
7450        BCS DIGEN1   ;YES,ERR.
7460        ADC RWQ      ;OVERFLOW?
7470        BCS DIGEN1   ;YES,ERR.
7480        ADC RWK      ;ADD LOW BITS
7490        BCS DIGEN1   ;ERR.IF OVERFLOW
7500        STA RWQ      ;SAVE NEW HIGH BITS
7510        DEX          ;COUNTER X-1=0?
7520        BNE LOPDIG   ;NO,LOOP
7530        RTS
7540 DIGEN1 JMP CMDEND   ;TO END ERR.COMMAND
7550 SKPEN1 LDA (RHL),Y  ;GET A CHAR
7560        CMP #$20     ;CHAR=' '?
7570        BNE DIGEN1   ;NO,ERR.
7580        INY
7590        DEX          ;COUNTER X-1=0?
7600        BNE SKPEN1   ;NO,LOOP
7610        LDA RWQ      ;'DIGIT' TO A ,RETURN
7620        RTS
7630   ;MOVE ROUTINE
7640   ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^
7650 MV3CHR LDX #$03     ;X AS COUNTER FOR MOVING 3 BYT
7660 BLKMOV LDY #$00
7670 BLKMO2 LDA (RHL),Y  ;GET CHAR
7680        STA (RDE),Y  ;SAVE CHAR
7690        INY
7700        DEX          ;COUNTER X-1=0?
7710        BNE BLKMO2   ;LOOP
7720        RTS
7730   ;COMPUTE DIR.ADDR.&GET ACHAR.
7740   ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^
7750 DMACUN LDY #DMAD
7760        STY RHL
7770        LDY #DMAD/256
```

```
7780        STY RHL+1     ;RHL POINTES DMA ADDR.
7790        CLC
7800        ADC RBC       ;(C):DIR.RELATIV.POSITION
7810        JSR FIFCBM    ;RHL+A TO RHL
7820        LDY #$00
7830        LDA (RHL),Y   ;GET A CHAR
7840        RTS
7850   ;*****************************************
7860   ; A:,B:,C:----,COMMAND
7870   ;*****************************************
7880 PSDK   LDA IBUBC+$1 ;DISK CHAR
7890        SEC
7900        SBC #$41      ;DISK NUMBER
7910        PHA
7920        JSR SELDSK    ;CHANR-DSK
7930        PLA
7940        STA DSKNUM
7950        JMP START
7960   ;*****************************************
7970   ;DIR COMMAND HANDLE ROUTINE
7980   ;*****************************************
7990 PDIR   JSR FIFCBB    ;FILL FILE NAME IN FCB
8000        LDA WRKPLA
8010        JSR SELDSK
8020        LDA FCBPB
8030        STA RHL
8040        LDA FCBPB+1
8050        STA RHL+1     ;RHL POINTES FCB
8060        LDY #$00
8070        STY FCBB+$0C   ;CLARE SOME PLACE
8080        LDA (RHL),Y   ;GET FIRST CHAR FOR D-NAME
8090        CMP #$20      ;CHAR=' '?
8100        BNE BRNCHB    ;NO,BEGIN TO SEARCH FILE
8110        LDX #$0B      ;YES,X AS COUNTER FOR 11 CHAR
8120 PATCHQ LDA #$3F      ;'?' TO A
8130        STA (RHL),Y   ;'?' TO FCB
8140        INY
8150        DEX           ;COUNTER X-1=0?
8160        BNE PATCHQ    ;NO,LOOP
8170 BRNCHB LDA #$00
8180;
8190        STA RWK       ;CLEAR COUNTER'RWK'
8200        JSR CRLF      ;OUT 'CR''LF'
8210        LDA WRKPLA    ;REQ.DISK
8220        CLC
8230        ADC #$41      ;TO DISK NUMB.
8240        JSR CHROB     ;OUT DISK NAME
8250        LDA #$3A      ;: TO A
8260        JSR CHROB     ;OUT ':'
8270        JSR SERFIL    ;SEARCH FOR THE FIRST
8280        BNE LOPDIR    ;SEARCH SUCCESS,JUMP TO
8290        JSR PRTER2    ;SEARCH FALSE,PRINT'NO FILE'
8300        JMP DIREND    ;TO END COMMAND
8310 LOPDIR LDA RETVAL    ;RET VALUSE TO A
```

```
8320         ROR A
8330         ROR A
8340         ROR A
8350         ROR A
8360         AND #$60
8370         STA RBC        ;(RET VALUES*32) TO RBC
8380         STA RWQ
8390         LDA #$0A       ;CHECK PROTECT
8400         JSR DMACUN     ;COUNT DIR ADDRESS
8410         BMI NXTDR2     ;
8420         LDA RWK
8430         INC RWK        ;COUNTER+1
8440         AND #$03
8450         STA RWQ+1      ;SAVE
8460         BNE WARMR
8470         JSR CRLF
8480          INC RWK
8490          JMP COMMON
8500 WARMR   JSR SPACE      ;OUTPUT ' '
8510          JSR SPACE     ;OUT ' '
8520          JSR SPACE     ;OUT ' '
8530 COMMON  JSR SPACE      ;OUTPUT ' '
8540         LDA RWQ
8550         STA RBC        ;(C):RET.VAL.*32
8560         LDX #$01       ;X AS COUNTER,INITIAL VAL 1
8570         STX RNW        ;SAVE 'X'REG.
8580 ONEDIR  LDA RNW         ;CHAR COUNTER TO A
8590         JSR DMACUN     ;GET CHAR
8600         AND #$7F
8610         JSR CHROB      ;OUT CHAR
8620         LDX RNW
8630         CPX #$08       ;END FILE NAME ?
8640;
8650;
8660;
8670;
8680;
8690         BNE MEDIR2
8700         JSR SPACE      ;INSERT ' '
8710         LDX RNW
8720 MEDIR2  CPX #$0C       ;END TYPE ?
8730         BEQ NXTDR2
8740         INC RNW
8750         BNE ONEDIR
8760 NXTDR2  LDA #$00
8770         JSR DMACUN     ;RHL PTR FCB
8780         LDA RHL
8790         STA RDE
8800         LDA RHL+1
8810         STA RDE+1      ;RDE PTR FCB
8820         LDA FCBB
8830         LDY #$00
8840         STA (RHL),Y    ;WRITE DSK TO FCB'
8850         LDX #$20       ;COUNT PAGE NUMB.
```

```
8860        JSR CDOSRT
8870        JSR PRNPG1    ;PRINT PAGE
8880        JSR CSTATD    ;GET CONSOLE STATUE
8890        BNE DIREND    ;IF STAT,<>0,JUMP
8900        JSR SERNXT    ;SEARCH FOR NEXT
8910        BEQ DIREND
8920        JMP LOPDIR
8930 DIREND LDA RWK       ;COUNTER TO A
8940        JMP PEND
8950   ;*************************************
8960   ;* ERA COMMAND HANDLE ROUTINE        *
8970   ;*************************************
8980 PERA   JSR FIFCBB    ;FILL FILE NAME IN FCB
8990        CMP #$0B      ;RETURN IF RET '?'COUNT,=11
9000        BNE DELEFI    ;NO,GO DELETE FILE
9010        LDA MSGAD3    ;YES,ERR,
9020        STA RBC
9030        LDA MSGAD3+1  ;
9040        STA RBC+1     ;RBC POINTES MESSAGE'ALL(Y/N)?
9050        JSR STRIOA    ;OUTPUT 'ALL (Y/N)?'
9060        JSR INBUFA    ;WAIT FOR ANSWER
9070        LDA IBUBC     ;INPUT BUFF COUNTER TO A
9080        CMP #$01      ;BUFF COUNTER=1?
9090        BNE ERAEN1    ;NO,GO TO END COMMAND
9100        LDA IBUFB     ;YES,GET CHAR
9110        CMP #$59      ;CHAR='Y'?
9120        BNE ERAEN1    ;NO,ERR, JUMP
9130        INC IBUFPA    ;PTR BUFF IN BUFF
9140 DELEFI JSR FCBTOD    ;FCB PTR RDE
9150        JSR DELFIL    ;DELECT FILE ,A=0~3IF SUCCESS
9160        CMP #$FF      ;RET VAL,=$FF IF FALSE
9170        BEQ ERAEN2    ;FALSE,JUMP
9180        JMP PEND      ;SUCCESS,END
9190 ERAEN1 JMP START
9200 ERAEN2 JSR PRTER2    ;OUT 'NO FILE'
9210        JMP PEND
9220   ;*************************************************
9230   ;*  TYPE COMMAND HANDLE ROUTINE                  *
9240   ;*************************************************
9250 PTYPE  JSR FIFCBB    ;FILL FILE NAME IN FCB
9260        BNE TYPEN1    ;IF RET '?'COUNT<>0,GO TO END
9270        LDA WRKPLA
9280        JSR SELDSK
9290        JSR OPNFL     ;OPEN FILE
9300        BEQ ERENDT    ;OPEN FALSE,JUMP
9310        JSR CRLF      ;SUCCESS,OUTPUT 'CR''LF'
9320 WRTTP  LDY #DMAD
9330        STY RHL
9340        LDY #DMAD/256
9350        STY RHL+1     ;RHL POINTES DMAD
9360 TPBEGN JSR RDSEQ     ;READ DISK
9370        BNE TEND      ;IF READ FALSE
9380        LDY #$00
9390 MEDTY  LDA (RHL),Y   ;GET CHAR
```

```
9400         CMP #$1A      ;CHAR=CTL-Z?,EOF?
9410         BEQ BLNKT     ;YES
9420         STY RNW
9430         JSR CHROUT    ;NO,OUTPUT A CHAR
9440         JSR CSTATD    ;GET CONSOLE STATUES
9450         BNE BLNKT     ;IF STATUES<>0,GO END
9460         LDY RNW
9470        INY
9480         CPY #$80      ;TYPE 128 CHAR
9490         BNE MEDTY     ;NO,LOOP
9500         JMP TPBEGN
9510 BLNKT   JMP PEND
9520 TEND    CMP #$01      ;'1'SHOWS EOF
9530         BEQ BLNKT     ;YES
9540         JSR PRTER1    ;NO,OUTPUT'READ ERROR'
9550 ERENDT  NOP
9560 TYPEN1  JMP CMDEND    ;TO END ERR. COMMAND
9570 BLNKS   JMP SAEND
9580   ;*****************************************
9590   ;*   SAVE COMMAND HANDLE ROUTINE          *
9600   ;*****************************************
9610 PSAVE   JSR DIGHD     ;FILL PAGE NUMB. RET NUMBER
9620         PHA
9630         JSR FIFCBB    ;FILL FILE NAME IN FCB,RET'?#
9640         BNE TYPEN1    ;'?'COUNT.<>0,ERR. GO ERR END
9650         LDA WRKPLA
9660         JSR SELDSK
9670         JSR FCBTOD    ;FCB POINTER TO RDE
9680         JSR DELFIL    ;DELECT FILE IF IT EXISTED
9690         JSR FCBTOD    ;
9700         JSR MAKFIL    ;MAKE FILE,RET'0'IF FALSE
9710         BEQ BLNKS     ;FALSE,JUMP TO END
9720         LDA #$00
9730         STA RWK+1
9740         STA FCBBD     ;CLEAR SOME PLACE
9750         CLC
9760         PLA
9770         STA RWK
9780         ADC RWK       ;NUMB.*2
9790         BCC ENTR1     ;LOW BITS NO'C' JUMP
9800         INC RWK+1     ;ADD 'C'
9810 ENTR1   STA RWK       ;PAGE N*2 TO RWK
9820         LDA #STRTD
9830         STA RDE
9840         LDA #STRTD/256
9850         STA RDE+1
9860 SAVBGN  LDA RWK
9870         ORA RWK+1     ;CHECK PAGE NUMB.=0?
9880         BEQ SFINSH    ;YES,END
9890         LDA RWK
9900         SEC
9910         SBC #$01
9920         BCS ENTR2
9930         DEC RWK+1
```

```
 9940 ENTR2  STA RWK
 9950        LDA #$80      ;128 BYTES
 9960        CLC
 9970        ADC RDE       ;+128
 9980        STA RHL
 9990        LDA RDE+1
10000        ADC #$00
10010        STA RHL+1     ;RHL POINTES NEXT RDE A@@R.
10020        JSR SETDMA    ;USINC RDE,SET DMA
10030        JSR FCBTOD    ;R@E POINTES FCB
10040        JSR WRSEQ   ;WRITE SEQUANTIAL,RET 0 IF SUCCE
10050        BNE SAEND         ;WRIPA FALSE(ERRRRRR* AJD
10060        LDA RHL
10070        STA RDE
10080        LDA RHL+1
10090        STA RDE+1     ;RDE POINTES NEW DIA ADDR.
10100        JMP SAVBGN    ;LOOP
10110 SFINSH JSR FCBTOD    ;RDE POINTES FCB
10120        JSR CLSFIL        ;CLOSE BILE
10130        CMP #$00      ;RETURN=0,CLOSE FALSE
10140        BNE SAEND2        ;SUCCASS,JUMPTO--
10150 SAEND  JSR CRLF      ;OUTPUT 'CR''LF'
10160        LDA MSGAD5
10170        STA RHL
10180        LDA MSGAD5+1
10190        STA RHL+1     ;RHH POINPES 'NK SPACE'
10200        JSR STRIOB    ;OUT 'NO SPACE'
10210 SAEND2 JSR STDMAI    ;INITIAL   DMA ADDR.
10220        JMP PEND
10230 ERENDZ JMP CMDEND    ;END ERR.
10240   ;*****************************************
10250   ;*  RENAME COMMAND HANDLE ROUTINE        *
10260   ;*****************************************
10270 PREN   JSR FIFCBB    ;FILL FILE NAME IN FCB,RET '?$
10280        BNE ERENDZ    ;IF RET'?'COUNTER<>0,ERR.
10290        LDA WRKPLA    ;REQUIRED DISK NO.TO A
10300        STA WRKPLC    ;SAVE
10310        LDA FCBP      ;NO REPEAT NAME,
10320        STA RHL
10330        LDA FCBP+1
10340        STA RHL+1     ;RHL POINTES NEW FILE NAME FCB
10350        LDA FCBPC
10360        STA RDE
10370        LDA FCBPC+1
10380        STA RDE+1     ;RDE POINTES TEMP.SAVE ARE
10390        LDX #$10      ;X AS COUNTER FOR MOVING16 CHA
10400        JSR BLKMOV    ;MOVE NEW NAME TO TEMP.AREA
10410        LDA IBUFPA
10420        STA RDE
10430        LDA IBUFPA+1 ;
10440        STA RDE+1     ;RDE POINTES INPUT BUFFER
10450        JSR SKPBLK    ;TO SKIP ' '
10460        CMP #$3D      ;CHAR='='?
10470        BEQ FIOLDN    ;YES,JUMP TO FILL OLD NAME
```

```
10480         CMP #$5F      ;CHAR='<-'?
10490         BNE ERENDN    ;NO,ERR. JUMP
10500 FIOLDN CLC
10510         LDA RDE
10520         ADC #$01
10530         STA IBUFPA
10540         LDA #$00
10550         ADC RDE+1
10560         STA IBUFPA+1 ;MODIFY IN-BUFF PTR
10570         JSR FIFCBB    ;FILL OLD FILE NAME
10580         BNE ERENDN    ,RET '?'COUNT,<>0,ERR.
10590         LDA FCBBC     ;OLD DSK IN FCB
10600         BEQ USNEWN
10610         LDA FCBB
10620         BEQ USNEWN    ;(A)=0,GO,USE NEW FILE DSK
10630        LDA WRKPLA
10640         CMP WRKPLC    ;COMPARE NEW DSK WITH OLD D
10650         BNE ERENDJ    ;ERR.
10660 USNEWN LDA FCBBC
10670         ORA FCBB
10680         STA FCBB
10690         STA FCBBC
10700         LDA FCBPC
10710         STA RDE
10720         LDA FCBPC+1
10730         STA RDE+1    ;RDE PTR NEW FILE
10740         JSR SERFIL+3 ;SEACH NEW FILE
10750         BNE ERENDR
10760         JSR SERFIL    ;SEARCH FOR OLD NAME FILE
10770         BEQ PRTEND    ;SEARCH FALSE,'NO FIRE'
10780         JSR FCBTOD    ;SUCCESS,RDE POINT.FCB
10790         JSR RENFIL    ;RENAME FILE
10800         JMP PEND      ;NORMALLY END
10810 PRTEND JSR PRTER2    ;OUTPUT 'NO FILE'
10820         JMP PEND
10830 ERENDN JMP CMDEND   ;ERR.
10840 ERENDR JSR CRLF      ;OUTPUT 'CR''LF'
10850         LDA MSGAD7
10860         STA RHL
10870         LDA MSGAD7+1 ;
10880         STA RHL+1     ;RHL POINTES'FILE EXISTS'
10890         JSR STRIOB    ;PRINT
10900         JMP PEND
10910 ERENDJ JMP CMDEND   ;END ERR.
10920    ;*****************************************
10930    ;*  USER COMMAND HANDLE ROUTINE          *
10940    ;*****************************************
10950 PUSER   JSR DIGHD     ;FILL 'DIGIT'IN FCB &COUNT
10960         CMP #$10      ;USER CODE>=16?
10970         BCS ERENDJ    ;YES,ERR. TO END
10980         LDX FCBBB
10990         CPX #$20      ;1'S CHAR=' '?
11000         BEQ ERENDJ    ;YES,ERR.
11010         JSR SETCOD    ;SET USER CODE
```

```
11020 BLKTR  JMP PEND
11030   ;*********************************************
11040   ;* TRANSIENT PROGRAM COMMAND HANDLE ROUTINE
11050   ;*********************************************
11060;PTRANS LDA FCBB      ;GET DSK NUMB.IN FCB
11070;
11080 PTRANS LDA WRKPLA
11090;       STA DSKNUM
11100;       DEC DSKNUM    ;TO CHANGE CURRENT DISK
11110;       JSR WRTDKN    ;SAVE CURRENT DISK #0004
11120      JSR SELDSK
11130        LDA #FCBB     ;FCB ADDRESS
11140        STA RWK
11150        LDA #FCBB/256
11160        STA RWK+1     ;RWK PTR FCB
11170        JSR CKQUEN    ;COUNT'?'
11180        BNE ERENDJ    ;JUMP TO END ERR.COMMAND
11190        LDA IBUFPA+1
11200        PHA
11210        LDA IBUFPA
11220        PHA
11230 FICOM  LDA FCBB+9    ;TYPE 1'SCHAR
11240        CMP #$20
11250        BNE ERENDJ
11260        LDA FCBP
11270        CLC
11280        ADC #$09
11290        STA RDE
11300        LDA #$00
11310        ADC FCBP+1
11320        STA RDE+1     ;RDE POINTES (FCBP+8)
11330        LDA MSGDD9
11340        STA RHL
11350        LDA MSGDD9+1
11360        STA RHL+1     ;RHL POINTES 'COM'
11370        JSR MV3CHR    ;FILL 'COM'IN FCB
11380        JSR OPNFL     ;OPEN FILE
11390        BEQ BLKJ6     ;OPEN FALSE,ERR.
11400        LDA #STRTD    ;OPEN SUCCESS,
11410        STA RWK
11420        LDA #STRTD/256
11430        STA RWK+1     ;RWK POINTES INPUT ADDR.
11440 RDDSK  LDA RWK
11450        STA RDE
11460        LDA RWK+1
11470        STA RDE+1     ;RDE POINTES INPUT ADDRESS
11480        JSR SETDMA    ;SET DMA
11490        JSR FCBTOD    ;RDE POINTES FCB
11500        JSR RDSEQB    ;READ SEQUENTIAL,RET0 IF SUCCES
11510        BNE RDEND     ;REAR FALSE,GO END
11520        CLC
11530        LDA RWK       ;GET INPUT ADDR.
11540        ADC #$80      ;+128 BYTES
11550        STA RWK
```

```
11560         LDA #$00
11570         ADC RWK+1
11580         STA RWK+1    ;RWK SAVE NEW INPUT ADDR.
11590         CMP #$B0     ;CHECK OVER LOAD?
11600         BCS BLKJ7    ;OVER,ERR.
11610         JMP RDDSK    ;NO,LOOP
11620 BLKJ6   JMP PRTERB
11630 BLKJ7   JMP PRTERP
11640 RDEND   CMP #$01     ;CHECK EOF?
11650         BNE BLKJ7    ;NO,ERR.
11660         JSR FIFCBB   ;FILL FILENAME FOLLOWING COMMA.
11670         INC WRKPLA
11680         LDA WRKPLA   ;FIRST FILE DSK TO A
11690         STA FCBB     ;FILL
11700         STA RWQ      ;SAVE
11710         LDA #$10     ;FOR FILLING SECOND FILE NAME
11720         JSR FIFCB    ;SECOND FILE NAME TO FCB
11730         LDA RWQ      ;FIRST FILE DISK TO A
11740         STA FCBB+16 ;AS SECOND FILE DISK
11750         LDA #$00
11760         STA FCBB+32 ;CLEAR A BYTE
11770         LDA #$10
11780         STA RDE
11790         LDA #$EF
11800         STA RDE+1    ;RDE PTR $EF10 FOR 2'S FILE NAM
11810         LDA FCBP
11820         STA RHL
11830         LDA FCBP+1
11840         STA RHL+1    ;RHL POINTES FCB
11850         LDX #$21     ;X AS COUNTER FOR 33CHAR
11860         JSR BLKMOV   ;MOVE FROM FCB TO NEW SAVE AREA
11870;
11880;
11890;
11900;
11910;
11920;
11930         PLA
11940         STA RHL
11950         PLA
11960         STA RHL+1    ;RHL POINTES INPUT BUFFER
11970         LDY #$00
11980 FNBLK   LDA (RHL),Y ;GET CHAR
11990         BEQ MVEND
12000         CMP #$20     ;CHAR=' '/
12010         BNE NEWLIN   ;Y,JMP
12020         LDA #$01
12030         JSR FIFCBM   ;RHL+1
12040         JMP FNBLK    ;LOOP
12050 NEWLIN  LDX #$00   ;INITIAL COUNTER X=0
12060         LDA #DMAD
12070         STA RDE
12080         LDA #DMAD/256
12090         STA RDE+1
```

```
12100         INC RDE     ;RDE PTR $E965 FOR SAVING OTHER
12110         LDY #$00
12120 MVSTAR LDA (RHL),Y
12130         STA (RDE),Y ;MOVE A CHAR
12140         BEQ MVEND   ;IF CHAR=0,STOP MOVING
12150         INX         ;COUNTER X+1
12160         INY
12170         JMP MVSTAR ;LOOP
12180 MVEND   TXA
12190        LDY #$00
12200        DEC RDE
12210         STA (RDE),Y ;$E965 SAVE INFO.COUNTER
12220;FROM $E966 TO E966+(E965 COUNT) SAVE ALL PARAME.
12230;FROM $EF10 TO EF30 SAVE SAVE TWO COMMAND INCL.DSK
12240;"""""""""""""""""""""""""""""""""""""""""""""""
12250;***********************************************
12260         JSR CRLF    ;OUTPUT 'CR''LF'
12270         JSR STDMAI ;SET DMA=DMAD
12280 EXECUT JSR STRTD   ;GO TO EXECUTE PROGRAM
12290         LDA DSKNUM ;CURRENT DISK
12300         JSR SELDSK ;SELECT CURRENT DISK
12310         JMP START
12320 PRTERB NOP
12330         JMP CMDEND ;ERR.END
12340 PRTERP JSR CRLF    ;OUTPUT 'CR''LF'
12350         LDA MSGBA2
12360         STA RHL
12370         LDA MSGBA2+1 ;
12380         STA RHL+1   ;RHL POINTES 'BAD LOAD'
12390         JSR STRIOB ;PRINT
12400         JMP PEND
12410 PEND    JSR FIFCBB ;FILL REMAIN SOMETHING
12420         LDA FCBBB   ;GET CHAR FROM FCB
12430         SEC
12440         SBC #$20    ;SUB ' 'VALUE
12450;         ORA WRKPLA
12460         BNE PENER   ;(A)<>0,ERR.END
12470         JMP START   ;END NORMALLY
12480 PENER   JMP CMDEND ;
12490 MSGVER .BYTE 'O.U.6502 CP/M VERSION',$0D,$0A
12500         .BYTE ' DIR ERA TYPE SAVE REN B:',$00
12510;
12520;
12530;
12540;
12550;
12560 RDERR   .BYTE 'READ ERROR',$00
12570 NOFERR .BYTE 'NO FILE',$00
12580 MSGADI .BYTE 'ALL (Y/N)?',$00
12590 MSGAD4 .BYTE 'NO SPACE',$00
12600 MSGAD6 .BYTE 'FILE EXISTS',$00
12610 MSGDD8 .BYTE 'COM'
12620 MSGBA1 .BYTE 'BAD LOAD',$00
12630         .END
```

Appendix B3 : Listing of BODS

```
12640       ;OUP/M BDOS WRITTEN BY SHAO, JIAN - XIONG
12650       ;ERROR NO. LISTING
12660       ;$02  DISK NO. TOO LARGE
12670       ;$03  FUNCTION NO. TOO LARGE
12680       ;$04  DISK WRITING PROTECTION
12690       ;$05  DUPLICTED FILE WHEN MAKE FILE
12700       ;$06  CHECK AREA NOT MATCH
12710       ;$07  FILE WRITING PROTECTION
12720       ;$08  FILE NOT OPEN
12730       *=$D93A
12740       START=$00             ;WARM BOOT ADDR.
12750       IOBYTE=$03            ;I/O BYTE
12760       CREG=$08              ;WORK STORAGE
12770       DEREG=$09
12780       HLREG=$0B
12790       HLREG1=$0D
12800       USERSP=$10
12810       RTNFLG=$18            ;RETURN FLAG
12820       TEMP=$1F
12830       CRTDRN=$25            ;'DR' RESERVATION
12840       SCNPTS=$38            ;SCREEN LINE START POINTER
12850       CHRNUB=$3E            ;CHAR. NOS IN INPUT BUFFER
12860       CSBUFS=$3F            ;OFFSET OF LAST CHAR TO
12870                             ;STARING ADDRESS OF THE BUFFER
12880       COUNT=$3D
12890       BRHTAB=$DBC3          ;BDOS
12900       ADCOFS=$DC80
12910       STRTFG=$DFDA
12920       RETURN=$E3D8
12930       R$TURN=$E3EE
12940       WBOOT=$E477           ;BIOS
12950       CSLTET=$E560
12960       CLINN=$E57A
12970       CLOUTT=$E591
12980       LITOUT=$E5A9
12990       ERROR=$E928
13000       BANBAC    .BYTE    $00  ;NO. COUNTER OF BACK SPACE
13010       PRTFLG    .BYTE    $00  ;PRINTER ON/OFF FLAG
13020       SCNPNT    .BYTE    $00  ;SCREEN LINE COUNTER
13030       PRECHR    .BYTE    $00  ;PREVIOUS CHARACTER
13040       WMBCOT    .BYTE    $2D  ;
13050 ;
13060       BDOS=*
13070 ;
13080       STX     CREG          ;STORE PARAMETERS
13090       STA     DEREG
13100       STY     DEREG+1
13110       LDA     #$00          ;INIT. SOME LOCATIONS FOR
13120       STA     CRTDRN        ;RETURN ROUTINE
13130       STA     RTNFLG
13140       STA     RTNFLG+1
13150       STA     RTNFLG+2
```

```
13160       LDA     #RETURN-1/256
13170       PHA                     ;SET RETURN ROUTINE ADDR.
13180       LDA     #RETURN-1
13190       PHA
13200       LDA     CREG            ;CHECK FUNCTION NO. LEGALITY
13210       CMP     #$29
13220       BCC     B$OS
13230       LDA     #$03            ;IF ILLEGAL, ERR!
13240       JMP     ERROR
13250       B$OS=*
13260       ASL     A               ;GET DESIRED ROUTIN
13270       LDX     #BRHTAB         ;ADDRESS FROM BRHTAB
13280       STX     HLREG
13290       LDX     #BRHTAB/256
13300       STX     HLREG+1
13310       LDY     #$00
13320       STY     TEMP
13330       JSR     ADCOFS
13340       LDA     (HLREG),Y
13350       STA     HLREG1
13360       INY
13370       LDA     (HLREG),Y
13380       STA     HLREG1+1
13390       LDA     DEREG
13400       JMP     (HLREG1)        ;JMPE TO DESIRED ROUTINE
13410       REV1*=*+40
13420 ;
13430 ; ************************************************************
13440 ; *                                                          *
13450 ; *  FUNCTION  0 :  SYSTEM  RESET                            *
13460 ; *                                                          *
13470 ; ************************************************************
13480 ; *                                                          *
13490 ; *  ENTRY PARAMETERS :                                      *
13500 ; *       REGISTER  X  : $00                                 *
13510 ; *                                                          *
13520 ; ************************************************************
13530 ;
13540       SYMRST=*
13550 ;
13560       JMP     WBOOT
13570 ;
13580       INCHR=*             ;GET A CHARACTER FROM CONSOLE
13590 ;
13600       LDA    PRECHR
13610       BEQ    I$CHR
13620       LDX    #$00
13630       STX    PRECHR
13640       RTS
13650       I$CHR=*
13660       LDX    #$00
13670       STX    PRECHR
13680       JMP    CLINN
13690       RESV2*=*+4
```

```
13700 ;
13710 ; ********************************************************
13720 ; *                                                      *
13730 ; *  FUNCTION  1 :  CONSOLE INPUT                        *
13740 ; *                                                      *
13750 ; ********************************************************
13760 ; *                                                      *
13770 ; *  ENTRY PARAMETERS :                                  *
13780 ; *      REGISTER  X  : $01                              *
13790 ; *                                                      *
13800 ; *  RETURNED VALUE :                                    *
13810 ; *      REGISTER A : ASCII CHARACTER                    *
13820 ; *                                                      *
13830 ; ********************************************************
13840 ;
13850       CSLIN=*
13860 ;
13870       JSR     INCHR       ;GET A CHAR FROM CONSOLE
13880       JSR     C$LIN       ;HANDLING CONTROL CHARACTER
13890       BCC     CS$IN       ;IF CONTROL CHAR EXCEPT
13900                           ;'CR','LF','BA','TAB',THEN RETURN
13910       PHA
13920       JSR     CSLOUT
13930       PLA
13940       CS$IN=*
13950       JMP     STRTFG
13960       C$LIN=*
13970       CMP     #$0D        ;'CR'?
13980       BNE     CSLIN1
13990       RTS
14000       CSLIN1=*
14010       CMP     #$0A        ;'LF'?
14020       BNE     CSLIN2
14030       RTS
14040       CSLIN2=*
14050       CMP     #$09        ;'TAB'?
14060       BNE     CSLIN3
14070       RTS
14080       CSLIN3=*
14090       CMP     #$08        ;'BA'?
14100       BNE     CSLIN4
14110       RTS
14120       CSLIN4=*
14130       CMP     #$20
14140       RTS
14150 ;
14160       INTPHD=*            ;INTERUPTING CHAR HANDLING
14170 ;
14180       LDA     PRECHR      ;CHECK IF THERE IS A PREVIOUS CHAR
14190       BNE     INTPH$
14200       JSR     CSLTET      ;TEST IF CONSOLE IS READY
14210       AND     #$01
14220       BEQ     I$TPHD
14230       JSR     CLINN       ;IF READY,READ A CHAR.
```

```
14240       CMP     #$13        ;CONTROL-S?
14250       BNE     IN$PHD
14260       INT$HD=*
14270       JSR     CLINN       ;IF IT IS CONTROL-S,WAIT UNTIL NEXT
14280                           ;CHAR COMING IN.
14290       CMP     #$03        ;CONTROL-C?
14300       BNE     INTP$D
14310       JMP     $0000       ;IF IS,TO WARM BOOT
14320       INTP$D=*
14330       LDA     $00         ;IF ANOTHER CHARACTER,RETURN
14340       RTS
14350       IN$PHD=*
14360       STA     PRECHR      ;WHEN FIRST RENDING IS NOT CTRL-S
14370                           ;STORE IT TO PREV. CHAR. LOCATION
14380       INTPH$=*
14390       LDA     #$FF
14400       I$TPHD=*
14410       RTS
14420 ;
14430       OUTCHR=*            ;OUTPUT A CHARACTER
14440 ;
14450       PHA
14460       LDA     BANBAC      ;TO SEE IF THIS IS FROM 'BA'
14470       BNE     O$TCHR      ;IF IS,NOT OUTPUT TO CSL
14480       JSR     INTPHD         ;HANDLING INTERUPTING CHAR.
14490       PLA
14500       PHA
14510       JSR     CLOUTT      ;OUTPUT TIS CHAR TO CONSOLE
14520       PLA
14530       PHA
14540       LDX     PRTFLG      ;TO SEE IF PRINTER
14550       BEQ     O$TCHR
14560       JSR     LITOUT      ;IF ON,ALSO SENT TO PRINTER.
14570       O$TCHR=*
14580       PLA
14590       CMP     #$7F        ;IF IS 'DEL' ,RETURN,THAT IS, NOT
14600                           ;INCREASING SCNPNT
14610       BNE     OU$CHR
14620       RTS
14630       OU$CHR=*
14640       INC     SCNPNT      ;IF IT LESS THAN $7F,GREATER THEN
14650                           ;EQUAL TO $20, SCNPNT INCREASED
14660                           ;BY 1
14670       CMP     #$20
14680       BCC     OUT$HR
14690       RTS
14700       OUT$HR=*
14710       DEC     SCNPNT      ;IF IT IS CTRL CHAR.,THAT IS, LESS
14720                           ;THAN $20
14730       LDX     SCNPNT      ;THEN, NOTCHANGE SCNPNT
14740       BNE     OUTC$R
14750       RTS
14760       OUTC$R=*
14770       CMP     #$08        ;IF IS 'BA',SCNPNT DECREASED BY 1
```

```
14780       BNE     OUTCH$
14790       DEC     SCNPNT
14800       RTS
14810       OUTCH$=*
14820       CMP     #$0A        ;IF IS 'LF',SCNPNT SHOULD BE SET
14830                           ;TO ZERO
14840       BNE     OUTC$$
14850       LDA     #$00
14860       STA     SCNPNT
14870       OUTC$$=*
14880       RTS
14890       CSLOU$=*
14900       JSR     C$LIN       ;DISTINGUISH CTRL & NO-CTRL CHAR
14910       BCS     CSLOUT      ;EXCEPT 'CR','LF','BA','TAB'
14920       PHA
14930       LDA     #$5E        ;IF CTRL CHAR, OUTPUT '^', FOLLOWED
14940                           ;BY A CHAR CORRES, THIS DESIRED
14950                           ;CONTROL CHARACTER
14960       JSR     OUTCHR
14970       PLA
14980       ORA     #$40
14990 ;
15000 ; ************************************************************
15010 ; *                                                          *
15020 ; *  FUNCTION  2 :  CONSOLE OUTPUT                           *
15030 ; *                                                          *
15040 ; ************************************************************
15050 ; *                                                          *
15060 ; *  ENTRY PARAMETERS :                                      *
15070 ; *       REGISTER  X  : $02                                 *
15080 ; *       REGISTER  A  : ASCII CHARACTER                     *
15090 ; *                                                          *
15100 ; ************************************************************
15110 ;
15120       CSLOUT=*
15130 ;
15140       CMP     #$09        ;CHECK FOR 'TAB'
15150       BEQ     C$LOUT
15160       JMP     OUTCHR
15170       C$LOUT=*
15180       LDA     #$20        ;IF IS 'TAB',OUTPUT SOME SPACE TO
15190       JSR     OUTCHR      ;FIT THE NEEDS OF THE 'TAB'
15200       LDA     SCNPNT
15210       AND     #$07
15220       BNE     C$LOUT
15230       RTS
15240       RESV*=*+1
15250 ;
15260 ;
15270 ;
15280 ;
15290 ;
15300 ;
15310 ;
```

```
15320 ; ************************************************************
15330 ; *                                                          *
15340 ; *  FUNCTION  3 :  LIST  OUTPUT                             *
15350 ; *                                                          *
15360 ; ************************************************************
15370 ; *                                                          *
15380 ; *  ENTRY PARAMETERS :                                      *
15390 ; *      REGISTER  X  : $03                                  *
15400 ; ************************************************************
15410 ;
15420       LITOTT=*
15430 ;
15440       JMP     LITOUT
15450 ;
15460 ; ************************************************************
15470 ; *                                                          *
15480 ; *  FUNCTION  4 : DIRECT CONSOLE I/O                        *
15490 ; *                                                          *
15500 ; ************************************************************
15510 ; *                                                          *
15520 ; *  ENTRY PARAMETERS :                                      *
15530 ; *      REGISTER  X  : $04                                  *
15540 ; *      REGISTER  A  : $FF (INPUT) OR CHAR(OUTPUT)          *
15550 ; *                                                          *
15560 ; *  RETURNED VALUE :                                        *
15570 ; *      REGISTER A : CHARACTER OR STATUS                    *
15580 ; *                                                          *
15590 ; ************************************************************
15600 ;
15610       DRCSIO=*
15620 ;
15630       TAX
15640       INX                 ;CHECK FOR INPUT OR OUTPUT
15650       BEQ     D$CSIO
15660       JMP     CLOUTT      ;OUTPUT THIS CHARACTER
15670       D$CSIO=*
15680       JSR     CSLTET      ;TEST IF CONSOLE IS READY INPUT
15690       AND     #$01
15700       BEQ     DR$SIO      ;IF NOT READY,RETURN
15710       JSR     CLINN       ;INPUT A CHARACTER
15720       DR$SIO=*
15730       JMP     STRTFG
15740       REESV*=*+3
15750 ;
15760 ;
15770 ;
15780 ;
15790 ;
15800 ;
15810 ;
15820 ;
15830 ;
15840 ;
15850 ;
```

```
15860 ; ************************************************************
15870 ; *                                                          *
15880 ; *  FUNCTION  5 :  BUFFER OUTPUT                            *
15890 ; *                                                          *
15900 ; ************************************************************
15910 ; *                                                          *
15920 ; *  ENTRY PARAMETERS :                                      *
15930 ; *      REGISTER  X  : $05                                  *
15940 ; *      REGISTER  Y,A: STARTING ADDRESS OF THE BUFFER       *
15950 ; *                                                          *
15960 ; ************************************************************
15970 ;
15980       OUTBUF=*
15990 ;
16000       LDY     #$00
16010       OU$BUF=*
16020       LDA     (DEREG),Y
16030       CMP     #$24        ;'$'?
16040       BNE     O$TBUF      ,IF IS, END
16050       RTS
16060       O$TBUF=*
16070       JSR     CSLOUT      ;IF NOT,OUTPUT THIS CHARACTER UNTIL
16080                           ;$ IS MET.
16090       INY
16100       BNE     OU$BUF
16110       INC     DEREG+1
16120       BNE     OU$BUF
16130 ;
16140       BKSPAC=*            ;BACK SPACE ONE CHARACTER
16150 ;
16160       JSR     B$SPAC
16170       LDA     #$20
16180       JSR     CLOUTT
16190       B$SPAC=*
16200       LDA     #$08
16210       JMP     CLOUTT
16220 ;
16230       NWLNHD=*            ;POSITION CURSOR TO THE LOCA. OF NEW
16240                           ;LINE CORRESPONDING TO THE ORIGINAL
16250                           ;POSITION IN CURRENT LINE
16260       LDA     #$23        ;OUTPUT # TO ENDED CURRENT LINE
16270       JSR     OUTCHR
16280       JSR     CRLFHD      ;CURSOR TO A NEW LINE
16290       NW$NHD=*
16300       LDA     SCNPNT      ;POSITION CURSOR TO THE LOCA.
16310                           ;CORRESPONDING THE ORIGINAL ONE
16320       CMP     SCNPTS
16330       BCC     N$LNHD
16340       RTS
16350       N$LNHD=*
16360       LDA     #$20
16370       JSR     OUTCHR
16380       JMP     NW$NHD
16390 ;
```

```
16400       CRLFHD=*            ;CR AND LF HANDLING
16410 ;
16420       LDA     #$0D
16430       JSR     OUTCHR
16440       LDA     #$0A
16450       JMP     OUTCHR
16460 ;
16470 ; ************************************************************
16480 ; *                                                          *
16490 ; *  FUNCTION  6 :  READ CONSOLE BUFFER                      *
16500 ; *                                                          *
16510 ; ************************************************************
16520 ; *                                                          *
16530 ; *  ENTRY PARAMETERS :                                      *
16540 ; *      REGISTER  X  : $06                                  *
16550 ; *      REGISTER  Y,A: STARTING ADDRESS OF THIS BUFFER      *
16560 ; *                                                          *
16570 ; *  RETURNED VALUE :                                        *
16580 ; *      CONSOLE CHARACTERS IN THE BUFFER                    *
16590 ; *                                                          *
16600 ; ************************************************************
16610 ;
16620       RDCSBF=*
16630 ;
16640       CLD
16650       LDA     SCNPNT      ;STORE THE INITVALUE OF SCREEN
16660                           ;LINE POINTER TO SCREEN STAR POINTER
16670       STA     SCNPTS
16680       LDA     #$00
16690       LDY     #$01
16700       STA     CHRNUB      ;KEEP COUNTER OF CURRENT CHAR.S
16710       STY     CSBUFS      ;AND OFFSET OF THE LAST CHAR TO START
16720                           ;ADDRESS OF THE BUFFER.
16730       RD$SBF=*
16740       JSR     INCHR       ;GET A CHARACTER
16750       AND     #$7F
16760       CMP     #$0D        ;'CR'?
16770       BEQ     RD$$$F
16780       CMP     #$0A        ;'LF'?
16790       BNE     RDC$$$
16800       RD$$$F=*
16810       JMP     RDC$$F
16820       RDC$$$=*
16830       CMP     #$08        ;'BA'?
16840       BNE     RDC$BF
16850       LDA     CHRNUB      ;'BA' HANDLING
16860       BEQ     RD$SBF
16870       DEC     CHRNUB      ;COUNTER DECREASE BY 1
16880       LDA     SCNPNT      ;STORE SCREEN LINE POINTER
16890       STA     BANBAC      ;TO BACK SPACE NUMBER ACCOUNTER
16900       JMP     RD$$BF
16910       RDC$BF=*
16920       NOP
16930       CMP     #$7F        ;'DEL'?
```

```
16940       BNE     RDCS$F
16950       LDX     CHRNUB
16960       BEQ     RD$SBF
16970       DEC     CHRNUB          ;NUMBER ACCOUNTER AND BUFFER
16980       DEC     CSBUFS          ;OFFSET DECREASE BY 1 TO ELIMMINATE
16990                               ;THE LAST WORD
17000       JMP     RD$SB$
17010       RDCS$F=*
17020       CMP     #$05            ;CONTROL-E?
17030       BNE     RDCSB$
17040       JSR     CRLFHD          ;GIVING A 'CR','LF'COMMAND
17050       LDA     #$00            ;SET SCREEN START POINTER TO ZERO
17060       STA     SCNPTS
17070       JMP     RD$SBF          ;GET NEXT CHARACTER
17080       RDCSB$=*
17090       CMP     #$10            ;CONTROL-P?
17100       BNE     R$$SBF
17110       LDA     #$80            ;REVERSE THE PRINTER FLAG
17120       EOR     PRTFLG
17130       STA     PRTFLG
17140       JMP     RD$SBF          ;GET NEXT CHARACTER
17150       R$$SBF=*
17160       CMP     #$18            ;'CONTROL-X'?
17170       BNE     R$C$BF
17180       RDCSB1=*
17190       LDA     SCNPTS          ;KEEP BACKING SPACE UNTIL THE START
17200                               ;POSITION OF CURRENT LINE ARRIVED
17210       CMP     SCNPNT
17220       BCS     RDCSBF
17230       DEC     SCNPNT
17240       JSR     BKSPAC
17250       JMP     RDCSB1
17260       R$C$BF=*
17270       CMP     #$15            ;'CONTROL U'?
17280       BNE     R$CS$F
17290       JSR     NWLNHD          ;CREATE A NEW LINE
17300       JMP     RDCSBF
17310       R$CS$F=*
17320       CMP     #$12            ;'CONTROL-R'?
17330       BNE     R$CSB$
17340       RD$$BF=*
17350       JSR     NWLNHD          ;NEW LINE HANDLING
17360       LDY     #$01            ;START TO WRITE A NEW LINE FROM THE
17370                               ;FIRST CHARCACTER OF CURRENT LINE IN
17380                               ;CONSOLE BUFFER
17390       STY     CSBUFS
17400       LDA     CHRNUB
17410       STA     TEMP
17420       RDCSB2=*
17430       BEQ     RD$S$F
17440       INC     CSBUFS          ;INCRE. OFFSET TO WRITE NEXT CHAR
17450       LDY     CSBUFS
17460       LDA     (DEREG),Y
17470       JSR     CSLOU$          ;WRITE NEXT CHARACTER
```

```
17480       DEC     TEMP
17490       JMP     RDCSB2
17500       RD$S$F=*
17510       LDA     BANBAC          ;IF AL CHAR HAVE RETYPED,CHECK FOR
17520       BNE     RDCSB5
17530       JMP     RD$SBF          ;IF BANBAC=0
17540       RDCSB5=*
17550       SEC
17560       SBC     SCNPNT          ;TO FIND THE DIFFERENCE NO. BETWEEN
17570                               ;CURRENT ACCOUNTER AND PREVIOUS
17580                               ;ACCOUNTER
17590       STA     BANBAC
17600       RDCSB3=*
17610       JSR     BKSPAC          ;BACK SPACE UNTIL CURRENT ACCOUNTER
17620       DEC     BANBAC          ;MATCHING PREVIOUS ONE
17630       BNE     RDCSB3
17640       JMP     RD$SBF
17650       R$CSB$=*
17660       INC     CSBUFS          ;SET UFFSET TO CURRENT CHAR.
17670       LDY     CSBUFS
17680       STA     (DEREG),Y       ;STORE THE CHAR INTO CORRESPONDING
17690                               ;LOCATIONN
17700       INC     CHRNUB          ;INCREASING CHR NUMBER 1
17710       RD$SB$=*
17720       PHA
17730       JSR     CSLOU$          ;TYPE IT
17740       PLA
17750       CMP     #$03            ;CHECK FOR CONTROL-C
17760       BNE     RDCSB4
17770       LDA     CHRNUB
17780       CMP     #$01
17790       BNE     RDCSB4
17800       JMP     $0000           ;IF IS CTRL-C AND IN STARTING
17810                               ;POSITION,GO TO WARM BOOT
17820       RDCSB4=*
17830       NOP
17840       NOP
17850       LDY     #$00            ;TO COMPARE CHAR NO. WITH MAX NO.
17860       LDA     (DEREG),Y
17870       CMP     CHRNUB
17880       BEQ     RDC$$F
17890       JMP     RD$SBF          ;IF LESS THAN,GET NEXT CHAR.
17900       RDC$$F=*
17910       LDY     #$01            ;STORE CHR NO. TO CONSOLE BUFFER
17920                               ;WHEN RETURN
17930       LDA     CHRNUB
17940       STA     (DEREG),Y
17950       JMP     CRLFHD
17960 ;
17970 ;
17980 ;
17990 ;
18000 ;
18010 ;
```

```
18020 ; ************************************************************
18030 ; *                                                          *
18040 ; *  FUNCTION  7 :  GET INPUT/OUTPUT BYTE                    *
18050 ; *                                                          *
18060 ; ************************************************************
18070 ; *                                                          *
18080 ; *  ENTRY PARAMETERS :                                      *
18090 ; *      REGISTER  X  : $07                                  *
18100 ; *                                                          *
18110 ; *  RETURNED VALUE :                                        *
18120 ; *      REGISTER A : INPUT/OUTPUT BYTE VALUE                *
18130 ; *                                                          *
18140 ; ************************************************************
18150 ;
18160       GTIOBY=*
18170 ;
18180       LDA     IOBYTE
18190       JMP     STRTFG
18200 ;
18210 ; ************************************************************
18220 ; *                                                          *
18230 ; *  FUNCTION  8 :  SET INPUT/OUTPUT BYTE                    *
18240 ; *                                                          *
18250 ; ************************************************************
18260 ; *                                                          *
18270 ; *  ENTRY PARAMETERS :                                      *
18280 ; *      REGISTER  X  : $08                                  *
18290 ; *      REGISTER  A  : INPUT/OUTPUT BYTE VALUE              *
18300 ; *                                                          *
18310 ; ************************************************************
18320 ;
18330       STIOBY=*
18340 ;
18350       STA     IOBYTE
18360       RTS
18370 ;
18380 ; ************************************************************
18390 ; *                                                          *
18400 ; *  FUNCTION  9 :  GET CONSOLE STATUS                       *
18410 ; *                                                          *
18420 ; ************************************************************
18430 ; *                                                          *
18440 ; *  ENTRY PARAMETERS :                                      *
18450 ; *      REGISTER  X  : $09                                  *
18460 ; *                                                          *
18470 ; *  RETURNED VALUE :                                        *
18480 ; *      REGISTER A : CONSOLE STATUS                         *
18490 ; *                                                          *
18500 ; ************************************************************
18510 ;
18520       GTCLST=*
18530 ;
18540       JSR     INTFHD
18550       JMP     STRTFG
```

```
18560       *=$DBC0
18570       START=$00
18580       CREG=$08
18590       DEREG=$09
18600       HLREG=$0B
18610       HLREG1=$0D
18620       FLAG=$0F            ;BIT3:'1' FOR BUILD UP CHECK AREA
18630                           ;BIT4:'1' FOR R/W SECTOR
18640                           ;BIT5:'1' FOR ADD MAP BIT
18650                           ;BIT6:'1' FOR WRITE
18660                           ;BIT7:'1' FOR DIR ENTRY FINISH
18670       DIRBFA=$12          ;DIR BUFFER
18680       MAPBFA=$14          ;MAP AREA
18690       CHKBFA=$16          ;CHECK AREA
18700       RTNFLG=$18          ;THREE RETURN FLAGS
18710       DRERNB=$1C          ;DIR ENTRY#
18720       DREROF=$1D          ;DIR ENTRY OFFSET
18730       TEMP=$1F
18740       BLKNUB=$21          ;BLOCK NUMBER
18750       TRKNUB=$22          ;TRACK NUMBER
18760       TEMP1=$23
18770       PRVDSK=$24          ;PREVIOUS DISK#
18780       CRTDRN=$25           ;RESERVED FOR DR OF FCB
18790       CMERCT=$27          ;NUMBER OF MATCHING FCB WITH DIR
18800       DERDSK=$29          ;DESIRED DISK#
18810       RTNFG1=$2A          ;AXULIRARY RETURN FLAG
18820       FLAG2=$2C
18830       CRRCNB=$2D          ;CR
18840       TLRCNB=$2E          ;RC
18850       EXNTNB=$2F          ;EX
18860       SECNUB=$30          ;SECTOR# WITHIN A TRACK
18870       TEMP2=$37
18880       BKRBLK=$31          ;BACK BLOCK#
18890       ADVBLK=$33          ;ADVANCED BLOCK#
18900       BKOSFG=$35          ;BLOCK OFFSET FLAG
18910       BLKOFS=$36          ;BLOCK OFFSET
18920       IOBFFG=$39          ;'FLUSH' FLAG
18930       CRTDSK=$3A          ;CURRENT DISK NO.
18940       CBADRV=$55          ;RESERVED FOR SERCH NEXT
18950       FLAG1=$58
18960       SYMRST=$D9A6        ;BDOI
18970       CSLIN=$D9C0
18980       CSLOUT=$DA5C
18990       LITOTT=$DA71
19000       DRCSIO=$DA74
19010       OUTBUF=$DA8B
19020       RDCSBF=$DACD
19030       GTIOBY=$DBB2
19040       STIOBY=$DBB7
19050       GTCLST=$DBBA
19060       PAMTAB=$E41F        ;BIOS
19070       LITOUT=$E5A9
```

```
19080       HOMEHD=$E5F5
19090       DKSEL1=$E643
19100       SETTR1=$E665
19110       SETSC1=$E668
19120       SETBL1=$E66B
19130       STDMA1=$E66E
19140       WRTTRK=$E822
19150       RWDBS1=$E887
19160       ERROR=$E928
19170       SIOBDK=$E90A
19180       DFBUFA=$E965
19190       DIRBUF=$EB65
19200       MAPAR0=$ED65
19210       MAPAR1=$EDA5
19220       CHKAR0=$EDE5
19230       CHKAR1=$EDF3
19240       FLCTBK=$EE01
19250       IOBUF=$B000
19260       USERCD    .BYTE $01        ;USER CODE
19270       WPTVCT    .BYTE $00        ;R/O VECTOR
19280       LOGVTR    .BYTE $00        ;LOGIN VECTOR
19290       ;BRANCH TABLE
19300       BRHTAB    .WORD  SYMRST,CSLIN,CSLOUT,LITOTT,DRCSIO
19310                 .WORD  OUTBUF,RDCSBF,GTIOBY,STIOBY,GTCLST
19320                 .WORD  RSDKSM,GTLGVT,GTCRDK,GTMPLT,SETWPT
19330                 .WORD  GTROVT,GTDKPA,SGTUCD,SETDMA,SELDSK
19340                 .WORD  MAKFIL,DELFIL,OPNFIL,CLSFIL,RDRCSQ
19350                 .WORD  WTRCSQ,RDRCRM,WTRCRM,SHFIST,SHNEXT
19360                 .WORD  RENAME,CHATRB,CPFISZ,STRDRC,SIODSK
19370 ;
19380 ; ************************************************************
19390 ; *                                                          *
19400 ; *  FUNCTION  10 :  RESET DISK SYSTEM                       *
19410 ; *                                                          *
19420 ; ************************************************************
19430 ; *                                                          *
19440 ; *  ENTRY PARAMETERS :                                      *
19450 ; *     REGISTER  X  : $0A                                   *
19460 ; *                                                          *
19470 ; ************************************************************
19480 ;
19490       RSDKSM=*
19500 ;
19510       LDA     #$00        ;INITIALIZE WRITING PROTECT VECTOR
19520       STA     LOGVTR      ;& DISK LOG VECTOR
19530       STA     WPTVCT
19540       STA     DERDSK      ;SELECT DISK A
19550       JSR     S$LDSK
19560       LDA     #DFBUFA     ;SELECT DEFAULT BUF FOR DMA
19570       LDX     #DFBUFA/256
19580       JMP     STDMA1
19590 ;
19600 ;
19610 ;
```

```
19620 ; ************************************************************
19630 ; *                                                          *
19640 ; *  FUNCTION  11 :  GET LOGIN VECTOR                        *
19650 ; *                                                          *
19660 ; ************************************************************
19670 ; *                                                          *
19680 ; *  ENTRY PARAMETERS :                                      *
19690 ; *      REGISTER  X  : $0B                                  *
19700 ; *                                                          *
19710 ; *  RETURNED VALUE :                                        *
19720 ; *      REGISTER A : LOGIN VECTOR                           *
19730 ; *                                                          *
19740 ; ************************************************************
19750 ;
19760       GTLGVT=*
19770 ;
19780       LDA     LOGVTR
19790       JMP     STRTFG
19800 ;
19810 ; ************************************************************
19820 ; *                                                          *
19830 ; *  FUNCTION  12 :  GET CURRENT DISK NUMBER                 *
19840 ; *                                                          *
19850 ; ************************************************************
19860 ; *                                                          *
19870 ; *  ENTRY PARAMETERS :                                      *
19880 ; *      REGISTER  X  : $0C                                  *
19890 ; *                                                          *
19900 ; *  RETURNED VALUE :                                        *
19910 ; *      REGISTER A : CURRENT DISK NUMBER                    *
19920 ; *                                                          *
19930 ; ************************************************************
19940 ;
19950       GTCRDK=*
19960       LDA     CRTDSK
19970       JMP     STRTFG
19980 ;
19990 ; ************************************************************
20000 ; *                                                          *
20010 ; *  FUNCTION  13 :  GET MAP ADDRESS OF CURRENT DISK         *
20020 ; *                                                          *
20030 ; ************************************************************
20040 ; *                                                          *
20050 ; *  ENTRY PARAMETERS :                                      *
20060 ; *      REGISTER  X  : $0D                                  *
20070 ; *                                                          *
20080 ; *  RETURNED VALUE :                                        *
20090 ; *    REGISTER Y,A : MAP ADDRESS OF CURRENT DISK            *
20100 ; *                                                          *
20110 ; ************************************************************
20120 ;
20130       GTMPLT=*
20140 ;
20150       LDA     MAPBFA
```

```
20160       STA     RTNFLG+1
20170       LDA     MAPBFA+1
20180       STA     RTNFLG+2
20190       RTS
20200 ;
20210 ; ************************************************************
20220 ; *                                                          *
20230 ; *  FUNCTION  14 :  SET WRITING PROTECT VECTOR               *
20240 ; *                                                          *
20250 ; ************************************************************
20260 ; *                                                          *
20270 ; *  ENTRY PARAMETERS :                                      *
20280 ; *      REGISTER  X  : $0E                                  *
20290 ; *                                                          *
20300 ; ************************************************************
20310 ;
20320       SETWPT=*
20330 ;
20340       LDA     CRTDSK          ;GET BIT PATTERN FOR CURRENT
20350       JSR     BNYBIT          ;DISK
20360       ORA     WPTVCT
20370       STA     WPTVCT          ;SET R/O FOR CURRENT DISK
20380       RTS
20390 ;
20400 ; ************************************************************
20410 ; *                                                          *
20420 ; *  FUNCTION  15 :  GET READ ONLY VECTOR                    *
20430 ; *                                                          *
20440 ; ************************************************************
20450 ; *                                                          *
20460 ; *  ENTRY PARAMETERS :                                      *
20470 ; *      REGISTER  X  : $0F                                  *
20480 ; *                                                          *
20490 ; *  RETURNED VALUE :                                        *
20500 ; *      REGISTER A : READY ONLY VECTOR                      *
20510 ; *                                                          *
20520 ; ************************************************************
20530 ;
20540       GTROVT=*
20550 ;
20560       LDA     WPTVCT
20570       JMP     STRTFG
20580 ;
20590 ;
20600 ;
20610 ;
20620 ;
20630 ;
20640 ;
20650 ;
20660 ;
20670 ;
20680 ;
20690 ;
```

```
20700 ; ************************************************************
20710 ; *                                                          *
20720 ; *  FUNCTION  16 :  GET PARAMETER TABLE ADDRESS OF          *
20730 ; *                  CURRENT DISK                            *
20740 ; *                                                          *
20750 ; ************************************************************
20760 ; *                                                          *
20770 ; *  ENTRY PARAMETERS :                                      *
20780 ; *      REGISTER  X  : $10                                  *
20790 ; *                                                          *
20800 ; *  RETURNED VALUE :                                        *
20810 ; *      REGISTER Y,A : PARAMETER TABLE ADDRESS              *
20820 ; *                                                          *
20830 ; ************************************************************
20840 ;
20850       GTDKPA=*
20860 ;
20870       LDA     CRTDSK
20880       LDX     #$03
20890       JSR     ASLDSR
20900       LDX     #PAMTAB        ;GET CORRES. ADDR.
20910       STX     HLREG
20920       LDX     #PAMTAB/256
20930       STX     HLREG+1
20940       LDX     #$00
20950       STX     TEMP
20960       JSR     ADCOFS
20970       LDA     HLREG          ;RETURN THE ADDR.
20980       STA     RTNFLG+1
20990       LDA     HLREG+1
21000       STA     RTNFLG+2
21010       RTS
21020 ;
21030 ; ************************************************************
21040 ; *                                                          *
21050 ; *  FUNCTION  17 :  SET/GET USERCODE                        *
21060 ; *                                                          *
21070 ; ************************************************************
21080 ; *                                                          *
21090 ; *  ENTRY PARAMETERS :                                      *
21100 ; *      REGISTER  X  : $11                                  *
21110 ; *      REGISTER  A  : $FF  (GET) OR USERCODE (SET)         *
21120 ; *                                                          *
21130 ; *  RETURNED VALUE :                                        *
21140 ; *      REGISTER A : CURRENT CODE OR (NO VALUE)             *
21150 ; *                                                          *
21160 ; ************************************************************
21170 ;
21180       SGTUCD=*
21190       CMP     #$FF           ;CHECK WETHER SET OR GET
21200       BNE     S$TUCD
21210       LDA     USERCD         ;GET IT
21220       JMP     STRTFG
21230       S$TUCD=*
```

```
21240       AND     #$1F
21250       STA     USERCD          ;GET IT
21260       RTS
21270 ;
21280 ; ***************************************************************
21290 ; *                                                             *
21300 ; *  FUNCTION  18 :  SET DMA ADDRESS                            *
21310 ; *                                                             *
21320 ; ***************************************************************
21330 ; *                                                             *
21340 ; *  ENTRY PARAMETERS :                                         *
21350 ; *      REGISTER  X  : $12                                     *
21360 ; *      REGISTER  Y,A: DMA ADDRESS                             *
21370 ; *                                                             *
21380 ; ***************************************************************
21390 ;
21400       SETDMA=*
21410 ;
21420       LDA     DEREG
21430       LDX     DEREG+1
21440       JMP     STDMA1
21450 ;
21460       ASLDSR=*                ;VALUE*(2**(X))
21470                               ;WHERE VALUE IS IN A & TEMP
21480 ;
21490       ASL     A
21500       ROL     TEMP
21510       DEX
21520       BNE     ASLDSR
21530       RTS
21540 ;
21550       ADCOFS=*                ;ADD OFFSET TO THE ADDR.
21560 ;
21570       CLC
21580       ADC     HLREG
21590       STA     HLREG
21600       LDA     TEMP
21610       ADC     HLREG+1
21620       STA     HLREG+1
21630       RTS
21640 ;
21650       GETENT=*                ;GET DESIRED DIR ENTRY ADDR.
21660 ;
21670       LDA     DIRBFA
21680       STA     HLREG
21690       LDA     DIRBFA+1
21700       STA     HLREG+1
21710       LDA     DREROF+1
21720       STA     TEMP
21730       LDA     DREROF
21740       JMP     ADCOFS
21750 ;
21760       BNYBIT=*                ;ENTRY:A BINARY
21770                               ;EXIT:A BIT PATTERN
```

```
21780 ;
21790       TAX
21800       LDA     #$01
21810       B$YBIT=*
21820       DEX
21830       BMI     BN$BIT
21840       ASL     A
21850       JMP     B$YBIT
21860       BN$BIT=*
21870       RTS
21880 ;
21890       ;FOR THE INPUT PHYSICAL BLOCK NO.IN A &TEMP
21900       ;CACULATE THE CORRS. BYTE OFFSET FROM BIT MAP
21910       ;AND STORE IT TO Y,THE BIT PATTERN WITHIN THE
21920       ;BYTE AND STORE IT TO A
21930 ;
21940       GTMAPB=*
21950 ;
21960       PHA                 ;LOW BYTE R-SHIFT 3 TIMES
21970       LSR     A
21980       LSR     A
21990       LSR     A
22000       LDX     #$05
22010       G$MAPB=*
22020       ASL     TEMP        ;HIGH BYTE L-SHIFT 5 TIMES
22030       DEX
22040       BNE     G$MAPB
22050       ORA     TEMP        ;THE RESULT: PHY. BLOCK NO./ 8
22060       TAY
22070       PLA
22080       AND     #$07        ;GET THE CORRES. BIT PATTERN
22090       STA     TEMP+1
22100       LDA     #$07
22110       SEC
22120       SBC     TEMP+1
22130       JMP     BNYBIT
22140 ;
22150       GTBKNB=*            ;GET PHYSICAL BLOCK NO.
22160                           ;AND STORE IN A & TEMP
22170 ;
22180       LDA     #$00
22190       STA     TEMP
22200       LDA     (HLREG1),Y  ;GET LOGICAL BLOCK NO.
22210       LDX     #$01        ;AND *2
22220       JSR     ASLDSR
22230       TAX
22240       TYA
22250       LSR     A           ;ODD OR EVEN?
22260       BCS     G$BKNB      ;ODD, -1
22270       JSR     G$BKNB      ;EVEN, -2
22280       G$BKNB=*
22290       CPX     #$00
22300       BNE     GT$KNB
22310       DEC     TEMP
```

```
22320       GT$KNB=*
22330       DEX
22340       TXA
22350       RTS
22360 ;
22370       MDFMPB=*                ;BUILD OR MODIFY BIT MAP
22380 ;
22390       JSR     GTBKNB          ;GET PHSICAL BLOCK NO.
22400       JSR     GTMAPB
22410       PHA
22420       LDA     #$20            ;ADD OR DELETE?
22430       BIT     FLAG
22440       BEQ     M$FMPB
22450       PLA
22460       ORA     (MAPBFA),Y      ;SET CORRES. BIT
22470       STA     (MAPBFA),Y
22480       RTS
22490       M$FMPB=*
22500       PLA
22510       EOR     #$FF
22520       AND     (MAPBFA),Y      ;RESET CORRES. BIT
22530       STA     (MAPBFA),Y
22540       RTS
22550 ;
22560       MDFMAP=*                ;BUILD OR MODIFY BIT MAP FOR
22570                               ;WHOLE ENTRY"
22580 ;
22590       LDA     HLREG           ;DIR ADDR. STORE TO HLREG1
22600       STA     HLREG1
22610       LDA     HLREG+1
22620       STA     HLREG1+1
22630       LDY     #$0F
22640       MD$MAP=*
22650       INY
22660       CPY     #$20            ;16 BYTES DONE?
22670       BNE     M$FMAP          ;NO, KEEP DOING
22680       RTS
22690       M$FMAP=*
22700       LDA     (HLREG1),Y      ;GET LOGICAL BLOCK NO.
22710       BEQ     MD$MAP          ;IF 0, SKIP IT
22720       STY     TEMP1
22730       JSR     MDFMPB          ;IF NO 0, MODIFY CORRE. BIT MAP
22740       LDY     TEMP1
22750       JMP     MD$MAP
22760 ;
22770       ADDDIR=*                ;GET THE CHECKSUM OF THE DIR
22780                               ;BLOCK AND STORE IT TO X & TEMP
22790                               ;EXIT:X,TEMP SUM OF WHOLE DIR
22800 ;
22810       LDX     #$02
22820       LDY     #$00
22830       LDA     #$00
22840       A$DDIR=*
22850       CLC                     ;ADD UP ALL BYTES IN THE BLOCK
```

```
22860       ADC     (HLREG),Y
22870       BCC     AD$DIR
22880       INC     TEMP
22890       AD$DIR=*
22900       INY
22910       BNE     A$DDIR
22920       INC     HLREG+1
22930       DEX
22940       BNE     A$DDIR          ;TWO PAGE DONE?
22950       TAX
22960       RTS
22970 ;
22980       DIRCHK=*                ;BUILD OR CHECK CHECKSUM
22990 ;
23000       LDA     BLKNUB          ;GET OFFSET FROM CHECKSUM AREA
23010       LDX     #$01            ;FOR DESIRED DIR BLOCK
23020       JSR     ASLDSR
23030       PHA
23040       LDA     #$00             ;SET DIR BUFFER ADDR.
23050       STA     TEMP
23060       LDA     DIRBFA
23070       STA     HLREG
23080       LDA     DIRBFA+1
23090       STA     HLREG+1
23100       JSR     ADDDIR          ;GET CKECKSUM FOR THIS DIR BLOCK
23110       PLA
23120       TAY
23130       TXA
23140       LDA     #$08
23150       BIT     FLAG            ;BUILD OR CHECK?
23160       BNE     D$RCHK
23170       TXA                     ;CHECKING CHECKSUM
23180       CMP     (CHKBFA),Y
23190       BNE     DI$CHK
23200       INY
23210       LDA     TEMP
23220       CMP     (CHKBFA),Y
23230       BNE     DI$CHK
23240       RTS
23250       DI$CHK=*
23260       JSR     SETWPT          ;IF NOT MATCH, SET CORRES. DISK
23270       LDA     #$06            ;TO R/O AND DISPALY ERR
23280       JMP     ERROR
23290       D$RCHK=*
23300       TXA                     ;BUILDING CHECKSUM
23310       STA     (CHKBFA),Y
23320       INY
23330       LDA     TEMP
23340       STA     (CHKBFA),Y
23350       RTS
23360 ;
23370       DSKPRM=*                ;GET CORRES. DISK PARAMS.
23380                               ;& STORE THEM TO DIRBFA,MAPBFA
23390                               ;AND CHKBFA
```

```
23400 ;
23410       LDA     CRTDSK
23420       JSR     DKSEL1
23430       LDY     #$05
23440       D$KPRM=*
23450       LDA     (HLREG),Y
23460       STA     DIRBFA,Y
23470       DEY
23480       BPL     D$KPRM
23490       RTS
23500 ;
23510       RWDRBK=*                ;READ OR WRITE DIR BLOCK
23520 ;
23530       JSR     R$DRBK          ;SET DIR BUFFER FOR DMA
23540       JSR     RWDBS1          ;R/W A DIR BLOCK
23550       LDA     #DFBUFA         ;RECOVE DMA TO DEFAULT BUFFER
23560       LDX     #DFBUFA/256
23570       JMP     STDMA1
23580       R$DRBK=*
23590       LDA     DIRBFA
23600       LDX     DIRBFA+1
23610       JMP     STDMA1
23620 ;
23630       RDDIRB=*                ;READ DIR BLOCKS TO DIR BUFFER
23640                               ;AND BUILD UP CHECKSUM
23650 ;
23660       INC     DRERNB          ;SET CURRENT ENTRY NO.
23670       LDA     DRERNB
23680       CMP     #$70            ;112 ENTRIES DONE?
23690       BCC     R$DIRB
23700       LDA     #$80            ;IF DONE, SET FLAG DONE
23710       ORA     FLAG
23720       STA     FLAG
23730       RTS
23740       R$DIRB=*
23750       AND     #$0F            ;GET CORRES. ENTRY OFFSET
23760       LDX     #$05            ;AWAY FROM THE DIR BUFFER
23770       LDY     #$00
23780       STY     TEMP
23790       JSR     ASLDSR
23800       STA     DREROF          ;STORE IT
23810       LDA     TEMP
23820       STA     DREROF+1
23830       ORA     DREROF          ;FIRST ENTRY?
23840       BEQ     BDD$RB
23850       BIT     FLAG1
23860       BMI     BDD$RB
23870       RTS                     ;IF NO, RETURN
23880       BDD$RB=*
23890       STA     FLAG1
23900       LDA     DRERNB
23910       AND     #$F0
23920       LSR     A
23930       LSR     A
```

```
23940       LSR     A
23950       LSR     A
23960       STA     BLKNUB          ;SET BLOCK NO.
23970       JSR     SETBL1
23980       LDA     #$03
23990       STA     TRKNUB          ;SET DIR TRACK NO.
24000       JSR     SETTR1
24010       JSR     RWDRBK          ;READ THIS DIR BLOCK
24020       JMP     DIRCHK          ;BUILD UP CHECKSUM
24030 ;
24040       BUDMAP=*                ;BUILD UP BIT MAP FOR
24050                               ;CURRENT DISK'
24060 ;
24070       LDA     #$00            ;INIT. BIT MAP TO 0
24080       LDY     #$3F
24090       B$DMAP=*
24100       STA     (MAPBFA),Y
24110       DEY
24120       BPL     B$DMAP
24130       LDA     #$FF            ;INIT. ENTRY COUNTER
24140       STA     DRERNB
24150       LDA     #$28            ;SET FLAG INDICATING READ BLOCK
24160       STA     FLAG            ;,BUILD BIT AMAP & CHECKSUM
24170       BUD$AP=*
24180       JSR     RDDIRB            ;READ A DIR BLOCK
24190       BIT     FLAG            ;112 ENTRIES DONE?
24200       BPL     BU$MAP
24210       RTS                     ;YES, RETURN
24220       BU$MAP=*
24230       JSR     GETENT          ;GET CORRES. ENTRY ADDR.
24240       LDA     #$E5
24250       LDY     #$00
24260       CMP     (HLREG),Y
24270       BEQ     BUD$AP          ;SKIP THE EMPTY ONE
24280       JSR     MDFMAP          ;BUILD UP THE BIT MAP FOR THE ENTRY
24290       JMP     BUD$AP
24300 ;
24310       LGNMAP=*                ;LOG IN CURRENT DISK
24320 ;
24330       JSR     DSKPRM          ;GET DISK PARA. FOR CURRENT DISK
24340       LDA     CRTDSK
24350       JSR     BNYBIT          ;GET BIT PATTERN
24360       BIT     LOGVTR
24370       BEQ     L$GMAP
24380       RTS
24390       L$GMAP=*
24400       ORA     LOGVTR
24410       STA     LOGVTR          ;LOG IN CURRENT DISK
24420       JMP     BUDMAP          ;BUILD UP BIT MAP & CHECKSUM
24430       RESV11*=*+3
24440 ;
24450 ;
24460 ;
24470 ;
```

```
24480 ;
24490 ;
24500 ; ***************************************************************
24510 ; *                                                             *
24520 ; *  FUNCTION  19 :  SELECT DISK                                *
24530 ; *                                                             *
24540 ; ***************************************************************
24550 ; *                                                             *
24560 ; *  ENTRY PARAMETERS :                                         *
24570 ; *      REGISTER  X  : $13                                     *
24580 ; *      REGISTER  A  : SELECT DISK                             *
24590 ; *                                                             *
24600 ; ***************************************************************
24610 ;
24620       SELDSK=*
24630 ;
24640       STA     DERDSK          ;CHECK LEGALITY
24650       CMP     #$02
24660       BCC     SE$DSK
24670       LDA     #$02
24680       JMP     ERROR
24690       SE$DSK=*
24700       CMP     CRTDSK          ;=CURRENT DISK?
24710       BNE     S$LDSK          ;IF NOT, SELECT IT AS CURRENT
24720       RTS
24730       S$LDSK=*
24740       JSR     SIOBDK          ;SEND LAST TRACK IN BUF TO DISK
24750       LDA     DERDSK
24760       STA     CRTDSK
24770       JMP     LGNMAP          ;BUILD UP BIT MAP & CHECKSUM FOR IT
24780 ;
24790       MDFYS2=*                ;SET FCB.S2 TO INDICATE FCB HAS BEEN
24800                               ;ACTIVED
24810 ;
24820       LDY     #$0E
24830       LDA     #$80
24840       STA     (DEREG),Y
24850       RTS
24860 ;
24870       FCBINZ=*                ;SELSCT DISK ACCORDING TO FCB
24880                               ;PUT USER'S CODE TO FCB.DR
24890 ;
24900       LDA     #$FF            ;INIT. RETURN FLAG
24910       STA     RTNFLG
24920       LDY     #$00
24930       LDA     (DEREG),Y       ;CURRENT DISK IS DESIRED?
24940       AND     #$1F
24950       BEQ     F$BINZ
24960       STA     DERDSK          ;IF NOT, STORE IT AS DESIRED
24970       DEC     DERDSK
24980       LDA     CRTDSK          ;KEEP CURRENT DISK
24990       STA     PRVDSK
25000       LDA     (DEREG),Y
25010       STA     CRTDRN
```

```
25020       AND     #$E0
25030       STA     (DEREG),Y
25040       LDA     DERDSK          ;SELECT DESIRED DISK
25050       JSR     SELDSK
25060       F$BINZ=*
25070       LDA     USERCD          ;STORE USERCODE TO FCB.DR
25080       LDY     #$00
25090       ORA     (DEREG),Y
25100       STA     (DEREG),Y
25110       RTS
25120 ;
25130       TETWPT=*                ;TEST FOR DISK WRITING PROTECT
25140 ;
25150       JSR     T$TWPT
25160       BNE     TE$WPT
25170       RTS
25180       TE$WPT=*
25190       LDA     #$04
25200       JMP     ERROR
25210       T$TWPT=*
25220       LDA     CRTDSK
25230       JSR     BNYBIT
25240       BIT     WPTVCT
25250       RTS
25260 ;
25270       FBDRMH=*                ;MATCHING FCB WITH DIR
25280 ;
25290       LDA     #$FF            ;INIT. ENTRY COUNTER
25300       STA     RTNFG1
25310       FBD$$$=*
25320       STA     DRERNB
25330       STX     CMERCT          ;KEEP MATCH NUMBERS
25340       FBDR$$=*
25350       LDA     #$00            ;SET CHECKING,DIR NOT FINISH
25360       STA     FLAG            ;AND READING BLOCK FLAG
25370       FB$RMH=*
25380       JSR     RDDIRB          ;READ A DIR BLOCK
25390       BIT     FLAG            ;112 ENTRIES DONE?
25400       BMI     FBD$$H
25410       JSR     GETENT          ;IF NOT, GET DESIRED ENTRY ADDR.
25420       LDX     CMERCT
25430       DEX                     ;TEST FOR MAKING FILE
25440       BEQ     FBD$MH
25450       INX
25460       LDY     #$FF
25470       F$DRMH=*
25480       INY
25490       DEX                     ;ALL DESIRED CHARS MATCHED?
25500       BMI     FBDR$H          ;YES, FILE NAME FOUND
25510       CPY     #$0D
25520       BEQ     F$DRMH
25530       LDA     (DEREG),Y   ;GET CURRENT CHAR AND MATCH IT
25540       JMP     F$$RMH
25550       FBD$MH=*
```

```
25560       LDA     #$E5            ;IF MAKE FILE, MATCH $E5
25570       LDY     #$00            ;TO FIND AN EMPTY ENTRY
25580       F$$RMH=*
25590       CMP     #$3F            ;'?'?
25600       BEQ     F$DRMH
25610       SEC
25620       SBC     (HLREG),Y
25630       AND     #$7F
25640       BEQ     F$DRMH          ;NEXT BYTE EXPECTED
25650       BNE     FB$RMH          ;NEXT ENTRY EXPECTED
25660       FBDR$H=*
25670       LDA     DRERNB          ;IF MATCH FOUND, SET MATCKED
25680       STA     RTNFLG+1        ;ENTRY NO.
25690       STA     RTNFG1
25700       RTS
25710       FBD$$H=*
25720       LDA     #$FF            ;IF NOT, SET $FF
25730       STA     RTNFLG+1
25740       RTS
25750 ;
25760       PTFBDK=*                ;PUT FCB TO DISK
25770       JSR     GETENT          ;GET DESIRED ENTRY ADDR,.
25780       LDY     #$00
25790       P$FBDK=*
25800       LDA     (DEREG),Y       ;COPY DESIRED CHARS. FROM FCB
25810       STA     (HLREG),Y       ;TO DIR ENTRY
25820       INY
25830       DEX
25840       BNE     P$FBDK
25850       PT$BDK=*
25860       LDA     #$48
25870       ORA     FLAG            ;SET WRITING BLOCK FLAG
25880       STA     FLAG
25890       JSR     DIRCHK
25900       LDA     BLKNUB          ;SET DIR BLOCK
25910       JSR     SETBL1
25920       LDA     #$03            ;SET DIR TRACK NO.
25930       JSR     SETTR1
25940       JMP     RWDRBK          ;WRITE DIR BLOCK TO DISK
25950 ;
25960       MKFILE=*
25970 ;
25980       JSR     TETWPT          ;DISK WRITE PROTECTED
25990       LDA     #$10            ;ORIGINAL MAKING?
26000       BIT     FLAG2
26010       BEQ     M$FILE
26020       LDX     #$0C
26030       JSR     FBDRMH          ;YES, CHECK FOR DUPLICATION
26040       BIT     FLAG
26050       BMI     M$FILE
26060       LDA     #$05            ;IF DUPLICATED, ERR
26070       JMP     ERROR
26080       M$FILE=*
26090       LDX     #$01
```

```
26100       JSR     FBDRMH          ;GO TO FIND AN EMPTY DIR
26110       BIT     FLAG            ;MATCHED?
26120       BPL     MK$ILE
26130       RTS
26140       MK$ILE=*
26150       LDY     #$0D
26160       LDA     #$00
26170       MKF$LE=*
26180       STA     (DEREG),Y       ;RESET REST BYTE IN FCB TO 0
26190       INY
26200       CPY     #$20
26210       BNE     MKF$LE
26220       LDX     #$20
26230       JSR     PTFBDK          ;WRITE IT BACK TO DIR TRACK
26240       JMP     MDFYS2          ;SET BIT 7 OF S2 TO INDICATE ACTIVED
26250 ;
26260 ; ************************************************************
26270 ; *                                                          *
26280 ; *  FUNCTION  20 :  MAKE FILE                               *
26290 ; *                                                          *
26300 ; ************************************************************
26310 ; *                                                          *
26320 ; *  ENTRY PARAMETERS :                                      *
26330 ; *      REGISTER  X  : $14                                  *
26340 ; *      REGISTER  Y,A: FCB ADDRESS                          *
26350 ; *                                                          *
26360 ; *  RETURNED VALUE :                                        *
26370 ; *      REGISTER A : $FF  DIRECTORY SPACE OVERFLOW           *
26380 ; *                   DIRECTORY ENTRY NUMBER IF SUCCESS      *
26390 ; *                                                          *
26400 ; ************************************************************
26410 ;
26420       MAKFIL=*
26430 ;
26440       JSR     FCBINZ
26450       LDA     #$10            ;SET BIT 4 OF FLAG2 TO INDICATE
26460       STA     FLAG2           ;ORIGINAL MAKING
26470       JMP     MKFILE
26480 ;
26490       SETRTN=*
26500 ;
26510       LDA     RTNFG1
26520       JMP     STRTFG
26530 ;
26540       TETFWT=*                ;TEST FOR FILE WRITE PROTECT
26550 ;
26560       JSR     GETENT          ;GET DIR ENRTRY ADDR.
26570       LDY     #$09            ;GET DIR.T1 AND CHECK IT'S BIT 7
26580       LDA     (HLREG),Y
26590       TE$FWT=*
26600       ASL     A
26610       BCS     T$TFWT
26620       RTS
26630       T$TFWT=*
```

```
26640       LDA     #$07
26650       JMP     ERROR
26660 ;
26670       DLFILE=*
26680 ;
26690       JSR     TETWPT          ;DISK WRITE PROTECTED
26700       LDX     #$0C            ;FIND DESIRED DIR ENTRY
26710       JSR     FBDRMH
26720       DL$ILE=*
26730       BIT     FLAG
26740       BPL     D$FILE
26750       RTS
26760       D$FILE=*
26770       JSR     TETFWT          ;FILE WRITE PROTECTED
26780       JSR     GETENT          ;GET DESIRED ENTRY ADDR.
26790       LDY     #$00            ;SET $E5 TO INDICATE DELETED
26800       LDA     #$E5
26810       STA     (HLREG),Y
26820       JSR     MDFMAP          ;MODIFY MAP
26830       JSR     PT$BDK          ;PUT DIR TO DISK
26840       JSR     FBDR$$
26850       JMP     DL$ILE          ;KEEP DOING, TILL ALL DONE
26860       RESV12*=*+3
26870 ;
26880 ; ************************************************************
26890 ; *                                                          *
26900 ; *  FUNCTION  21 :  DELETE FILE                             *
26910 ; *                                                          *
26920 ; ************************************************************
26930 ; *                                                          *
26940 ; *  ENTRY PARAMETERS :                                      *
26950 ; *      REGISTER  X  : $15                                  *
26960 ; *      REGISTER  Y,A: FCB ADDRESS                          *
26970 ; *                                                          *
26980 ; *  RETURNED VALUE :                                        *
26990 ; *      REGISTER A : $FF FILE NOT FOUND                     *
27000 ; *                   DIRECTORY ENTRY NUMBER IF SUCCESS      *
27010 ; *                                                          *
27020 ; ************************************************************
27030 ;
27040       DELFIL=*
27050 ;
27060       JSR     FCBINZ
27070       JSR     DLFILE
27080       JMP     SETRTN
27090 ;
27100       CPDRFB=*            ;COPY DIR ENTRY TO FCB
27110 ;
27120       JSR     GETENT
27130       LDY     #$00
27140       C$DRFB=*
27150       LDA     (HLREG),Y
27160       STA     (DEREG),Y
27170       INY
```

```
27180       CPY     #$20
27190       BNE     C$DRFB
27200       RTS
27210 ;
27220       OPFILE=*
27230 ;
27240       LDX     #$0D            ;FIND DESIRED DIR ENTRY
27250       JSR     FBDRMH
27260       BIT     FLAG
27270       BPL     O$FILE
27280       RTS
27290       O$FILE=*
27300       JSR     CPDRFB          ;COPY DIR ENTRY TO FCB
27310       JMP     MDFYS2
27320 ;
27330 ; ************************************************************
27340 ; *                                                          *
27350 ; *  FUNCTION  22 :  OPEN FILE                               *
27360 ; *                                                          *
27370 ; ************************************************************
27380 ; *                                                          *
27390 ; *  ENTRY PARAMETERS :                                      *
27400 ; *      REGISTER  X  : $16                                  *
27410 ; :      REGISTER  Y,A: FCB ADDRESS                          *
27420 ; *                                                          *
27430 ; *  RETURNED VALUE :                                        *
27440 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
27450 ; *                   DIRECTORY ENTRY NUMBER IF SUCCESS      *
27460 ; *                                                          *
27470 ; ************************************************************
27480 ;
27490       OPNFIL=*
27500 ;
27510       JSR     FCBINZ
27520       JMP     OPFILE
27530 ;
27540       CLFILE=*
27550 ;
27560       LDA     #$00
27570       STA     RTNFLG+1
27580       JSR     TETWPT          ;DISK WRITE PROTECTED
27590       LDY     #$0E
27600       LDA     (DEREG),Y
27610       ASL     A
27620       ASL     A
27630       BCS     C$FILE
27640       RTS
27650       C$FILE=*
27660       LDX     #$0D            ;FIND A MATCHED ENTRY
27670       JSR     FBDRMH
27680       BIT     FLAG
27690       BPL     CL$ILE
27700       RTS
27710       CL$ILE=*
```

```
27720       JSR     GETENT          ;GET THE ADDR.
27730       LDY     #$10
27740       CLFI$$=*
27750       ;COPY FILE LOCATION AREA IN FCB TO THE DIR ENTRY'
27760       ;NOTE: IF (CORRS. BYTES NOT MATCH) AND (NONE OF '
27770       ; THEM ARE 0, THEN ERR. OTHERWISE, COPY NONE 0 BYTE'
27780       ;TO 0 BYTE IF ONE OF THEM IS 0; OR LEAVE MATCHED'
27790       ;PAIR ALONE'
27800       LDA     (DEREG),Y       ;COPY FILE LOCATION AREA IN FCB
27810       BNE     CLF$LE
27820       LDA     (HLREG),Y
27830       STA     (DEREG),Y
27840       CLF$LE=*
27850       LDA     (HLREG),Y
27860       BNE     CLFI$E
27870       LDA     (DEREG),Y
27880       STA     (HLREG),Y
27890       CLFI$E=*
27900       LDA     (DEREG),Y
27910       CMP     (HLREG),Y
27920       BNE     CLFIL$
27930       INY
27940       CPY     #$20
27950       BNE     CLFI$$
27960       LDY     #$0F            ;COPY 'RC' FROM FCB TO DIR ENTRY
27970       LDA     (DEREG),Y
27980       STA     (HLREG),Y
27990       JMP     PT$BDK          ;PUT DIR BLOCK TO DISK
28000       CLFIL$=*
28010       DEC     RTNFLG+1        ;IF ERR, SET $FF TO INDICATE NOT SUECC.
28020       RTS
28030 ;
28040 ; ************************************************************
28050 ; *                                                          *
28060 ; *  FUNCTION  23 :  CLOSE FILE                              *
28070 ; *                                                          *
28080 ; ************************************************************
28090 ; *                                                          *
28100 ; *  ENTRY PARAMETERS :                                      *
28110 ; *      REGISTER  X  : $17                                  *
28120 ; *      REGISTER  Y,A: FCB ADDRESSS                         *
28130 ; *                                                          *
28140 ; *  RETURNED VALUE :                                        *
28150 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
28160 ; *                   DIRECTORY ENTRY NUMBER IF SUCCESS      *
28170 ; *                                                          *
28180 ; ************************************************************
28190 ;
28200       CLSFIL=*
28210 ;
28220       JSR     FCBINZ
28230       JMP     CLFILE
28240 ;
28250       STRTFG=*            ;SET RETURN FLAG
```

```
28260 ;
28270       STA     RTNFLG+1
28280       RTS
28290 ;
28300       FLEDRT=*                ;FILE END RETURN
28310 ;
28320       LDA     #$01
28330       JMP     STRTFG
28340 ;
28350       GETREC=*                ;GET RC,EX,CR AND STORE THEM
28360 ;
28370       LDY     #$0C
28380       LDA     (DEREG),Y
28390       STA     EXNTNB
28400       LDY     #$0F
28410       LDA     (DEREG),Y
28420       STA     TLRCNB
28430       LDY     #$20
28440       LDA     (DEREG),Y
28450       STA     CRRCNB
28460       RTS
28470 ;
28480       TETOPN=*        ;;TEST FOR FCB OPENING
28490 ;
28500       LDY     #$0E
28510       LDA     (DEREG),Y
28520       ASL     A
28530       BCS     T$TOPN
28540       LDA     #$08
28550       JMP     ERROR
28560       T$TOPN=*
28570       RTS
28580 ;
28590       GTBKOF=*                ;KNOWING FCB,RC TO GET CURRENT
28600                               ;POSITION IN FCB FILE LOCATION AREA'
28610                               ;TO Y AND SECTOR# WITHIN THIS BLOCK
28620                               ;TO TEMP2
28630 ;
28640       LDA     #$03
28650       AND     CRRCNB
28660       STA     TEMP2
28670       LDA     CRRCNB
28680       LSR     A
28690       LSR     A
28700       CLC
28710       ADC     #$10
28720       TAY
28730       RTS
28740 ;
28750       GTRLBS=*                ;GET PHYSICAL BLOCK NO. AND
28760                               ;SECTOR NO. WITHIN THE BLOCK
28770 ;
28780       JSR     GTBKOF          ;GET CURRENT POSITION AND
28790                               ;SECTOR NO. WITIHIN THE BLOCK
```

```
28800       LDA     (DEREG),Y     ;GET LOGICAL BLOCK NO.
28810       BEQ     G$RLBS        ;SKIP 0
28820       LDX     DEREG
28830       STX     HLREG         ;CACULATE PHYSICAL BLOCK NO.1
28840       LDX     DEREG+1
28850       STX     HLREG1+1
28860       JSR     GTBKNB
28870       LDX     #$01
28880       G$RLBS=*
28890       RTS
28900 ;
28910       GTTKSC=*              ;GET TRACK# AND PHYSICAL SECTOR#
28920                             ;WITHIN THIS TRACK
28930                             ;ENTRY PHYSICAL BLOCK# IN A,TEMP
28940                             ;EXIT X TRACK#,A PHYSICAL SECTOR#
28950 ;
28960       LDX     #$FF
28970       G$TKSC=*
28980       SEC
28990       INX                    ;BLOCK# DIV 7 TO GET TRACK NO.
29000       SBC     #$07
29010       BCS     G$TKSC
29020       DEC     TEMP
29030       BPL     G$TKSC
29040       ADC     #$07           ;BLOCK# MOD 7
29050       ASL     A
29060       ASL     A              ;(BLOCK# MOD 7)*4
29070       CLC
29080       ADC     TEMP2          ;ADDING THE SECTOR NO. WITHIN
29090                              ;THE BLOCK TO GET SECTOR NO. '
29100                              ;WITHON THE TRACK
29110       PHA
29120       TXA
29130       SED
29140       ADC     #$04
29150       CLD
29160       TAX                    ;PUT TRACKNO. TO X
29170       PLA                    ;PUT SECTOR NO. TO A
29180       RTS
29190 ;
29200       RWSCDF=*              ;READ A SECTOR INTO DEFAULT BUFFER
29210 ;
29220       STX     TRKNUB
29230       JSR     SETSC1
29240       TXA
29250       JSR     SETTR1        ;SET TRACK NO.
29260       JMP     RWDBS1
29270 ;
29280       MDFFCB=*              ;MODIFY FCB TO FIT NEXT OPERATION
29290 ;
29300       BIT     FLAG2         ;RAN. OR SEQ. MODE?
29310       BPL     M$FFCB
29320       INC     CRRCNB
29330       M$FFCB=*
```

```
29340       LDA     CRRCNB
29350       LDY     #$20
29360       STA     (DEREG),Y     ;STORE TO FCB.CR
29370       LDA     TLRCNB
29380       LDY     #$0F
29390       STA     (DEREG),Y     ;STORE TO FCB.RC
29400       RTS
29410 ;
29420       HDNTEN=*              ;DEAL WITH NEXT EXTENT ENTRY
29430 ;
29440       JSR     CLFILE        ;CLOSE THE CURRENT EXTENT
29450       LDX     RTNFLG+1      ;$FF MEANS NOT SUCCES.
29460       INX
29470       BNE     H$NTEN        ;SUCCES! CONTINUE.
29480       RTS
29490       H$NTEN=*
29500       LDY     #$0C
29510       LDA     #$01
29520       CLC
29530       ADC     (DEREG),Y     ;INCREAMENT 'EX' BY 1
29540       STA     (DEREG),Y
29550       LDX     #$0D          ;FIND A CORRES. 'EX' IN DIR
29560       JSR     FBDRMH
29570       BIT     FLAG          ;FOUND?
29580       BPL     H$$TEN
29590       BIT     FLAG2         ;IF NOT, CHHECK R OR W?
29600       BVC     HDN$EN
29610       JMP     FLEDRT        ;IF R, ERR
29620       HDN$EN=*
29630       JSR     MKFILE        ;IF W, CREATE A NEW 'EX'
29640       BIT     FLAG
29650       BPL     HDNT$N
29660       LDA     #$03
29670       JMP     STRTFG
29680       H$$TEN=*
29690       JSR     O$FILE        ;COPY CORRES. BYTES IN DIR TO FCB
29700       HDNT$N=*
29710       JSR     GETREC        ;GET 'EX','RC','CR' AND STORE THEM
29720       LDA     #$00
29730       JMP     STRTFG        ;SET SUCCES. FLAG
29740 ;
29750       RDRCS1=*
29760 ;
29770       JSR     TETOPN        ;FILE OPENED?
29780       LDA     #$E0          ;SET FLAG2 TO INDICATE SEQ. R
29790       STA     FLAG2
29800       RDRCS$=*
29810       JSR     GETREC
29820       LDA     CRRCNB        ;'CR'<'RC'?
29830       CMP     TLRCNB
29840       BCC     R$RCSQ        ;YES, GO AHEAD TO READ
29850       CMP     #$40          ;NO, CHECK 'CR'=$40?
29860       BEQ     RD$CSQ
29870       JMP     FLEDRT        ;IF NOT,ERR
```

```
29880       RD$CSQ=*
29890       JSR     HDNTEN          ;CLOSE CURRENT 'EX', OPEN NEXT
29900       LDA     #$00            ;RESET CRRCNB
29910       STA     CRRCNB
29920       LDA     RTNFLG+1        ;SUCCES.
29930       BEQ     R$RCSQ
29940       JMP     STRTFG
29950       R$RCSQ=*
29960       JSR     GTRLBS          ;GET PHY. BLOCK NO.
29970       BNE     RDRC$Q
29980       LDA     #$02            ;IF CORRES. LOG. BLO. NO.=0, ERR
29990       JMP     STRTFG
30000       RDRC$Q=*
30010       JSR     GTTKSC          ;GET TRACK NO. & SECTOR NO.
30020       LDY     #$10
30030       STY     FLAG
30040       JSR     RWSCDF          ;READ A SECTOR
30050       JMP     MDFFCB          ;MODIFY BIT MAP
30060 ;
30070 ; ************************************************************
30080 ; *                                                          *
30090 ; *  FUNCTION  24 :  READ SEQUENTIALLY                       *
30100 ; *                                                          *
30110 ; ************************************************************
30120 ; *                                                          *
30130 ; *  ENTRY PARAMETERS :                                      *
30140 ; *      REGISTER  X  : $18                                  *
30150 ; *      REGISTER  Y,A: FCB ADDRESS                          *
30160 ; *                                                          *
30170 ; *  RETURNED VALUE :                                        *
30180 ; *      REGISTER A : $00  SUCCESS                           *
30190 ; *                   $01  FILE END                          *
30200 ; *                   $02  RECORD NOT IN DISK                *
30210 ; *                                                          *
30220 ; ************************************************************
30230 ;
30240       RDRCSQ=*
30250 ;
30260       JSR     FCBINZ
30270       JMP     RDRCS1
30280 ;
30290       FNDEPT=*                ;FIND EMPTY BLOCK,ENTRY A,TEMP
30300                               ;PHY. BLOCK# OF THE ONE BEFORE IT
30310                               ;EXIT A,TEMP EMPTY PHY. BLOCK#
30320 ;
30330       STA     BKRBLK          ;SET BK. & AD. COUNTER
30340       STA     ADVBLK
30350       LDA     TEMP
30360       STA     BKRBLK+1
30370       STA     ADVBLK+1
30380       LDA     #$01            ;SET CURRENT POINTER ODD OR EVEN FLAG
30390       AND     BLKOFS
30400       STA     BKOSFG
30410       F$$EPT=*
```

```
30420       LDA     BKRBLK          ;BK. SEARCK REACH BOTTOM?
30430       ORA     BKRBLK+1
30440       BEQ     F$DEPT          ;YES,GO TO AD. SEARCH
30450       FN$$PT=*
30460       LDA     BKRBLK          ;BK. SEARCH
30470       BNE     FN$EPT
30480       DEC     BKRBLK+1
30490       FN$EPT=*
30500       DEC     BKRBLK          ;BK. COUNTER MATCH THE E/O FLAG SET BEF?
30510       LDA     BKRBLK
30520       AND     #$01
30530       CMP     BKOSFG
30540       BNE     F$$EPT          ;NO, SKIP IT
30550       LDA     BKRBLK          ;YES, GET CORRES. BYTE & BIT PATTERN
30560       LDX     BKRBLK+1        ;IN BIT MAP AREA
30570       STX     TEMP
30580       JSR     GTMAPB
30590       TAX
30600       AND     (MAPBFA),Y      ;OCCUPIED?
30610       BNE     F$DEPT          ,YES, TO AD. SEARCH
30620       TXA                     ;NO,SET BIT MAP
30630       ORA     (MAPBFA),Y
30640       STA     (MAPBFA),Y
30650       LDA     BKRBLK+1        ;BK. COUNTER AS FOUND PHY. BLOCK
30660       STA     TEMP            ;NO. THEN RETURN
30670       LDA     BKRBLK
30680       RTS
30690       F$DEPT=*
30700       LDA     ADVBLK          ;AD. SEARCH TOP REACHED?
30710       CMP     #$FF
30720       BNE     FND$PT
30730       LDA     ADVBLK+1
30740       CMP     #$01
30750       BEQ     FNDE$T          ;YES, TO CHECK IF BA. BOTTOM REACHED?
30760       FND$PT=*
30770       INC     ADVBLK          ;INCREMENT AD. COUNTER BY 1
30780       BNE     FNDE$$
30790       INC     ADVBLK+1
30800       FNDE$$=*
30810       LDA     ADVBLK          ;THE SAME PROCESS AS IN BA. SEARCH
30820       AND     #$01
30830       CMP     BKOSFG
30840       BNE     F$DEPT
30850       LDA     ADVBLK
30860       LDX     ADVBLK+1
30870       STX     TEMP
30880       JSR     GTMAPB
30890       TAX
30900       AND     (MAPBFA),Y
30910       BNE     F$$EPT
30920       TXA
30930       ORA     (MAPBFA),Y
30940       STA     (MAPBFA),Y
30950       LDA     ADVBLK+1
```

```
30960       STA     TEMP
30970       LDA     ADVBLK
30980       RTS
30990       FNDE$T=*
31000       LDX     BKRBLK+1        ;BA. BOTTOM ALSO REACHED?
31010       INX
31020       BNE     FN$$PT          ;NO, TO BA. SEARCH
31030       LDA     #$A8            ;REPORT MAP OVERFLOW
31040       STA     FLAG2
31050       RTS
31060 ;
31070       RELFOM=*                ;PHY. BLOCK# TO LOGICAL ONE
31080                               ;EXIT X LOGICAL BLOCK#
31090 ;
31100       LDX     TEMP
31110       STX     TEMP1
31120       PHA
31130       INC     BLKOFS          ;GET CURRENT POSITION IN
31140       LDA     BLKOFS          ;FCB FILE LOCATION AREA
31150       LSR     A               ;EVEN OR ODD?
31160       PLA
31170       ADC     #$01            ;EVEN,ADD 1;ODD,ADD 2
31180       BCC     R$LFOM          ;TO PHY. BLOCK#
31190       INC     TEMP1
31200       R$LFOM=*
31210       LSR     A               ;DIVIDED BY 2
31220       LSR     TEMP1
31230       BCC     RE$FOM
31240       ORA     #$80
31250       RE$FOM=*
31260       TAX
31270       DEC     BLKOFS
31280       RTS
31290 ;
31300       WTRCS1=*
31310 ;
31320       JSR     TETOPN          ;OPENED?
31330       LDA     #$A0            ;SET FALG2 TO INDICATE SEQ. W
31340       STA     FLAG2
31350       WTRCS$=*
31360       JSR     TETWPT          ;DISK W PROTECTED?
31370       LDY     #$09
31380       LDA     (DEREG),Y
31390       JSR     TE$FWT          ;FILE W PROTECTED?
31400       JSR     GETREC
31410       LDA     CRRCNB          ;'CR'<$40
31420       CMP     #$40
31430       BCC     W$RCSQ
31440       JMP     FLEDRT
31450       W$RCSQ=*
31460       JSR     GTBKOF          ;GET CUR. RECORD POSI. IN
31470                               ;FILE LOCATION AREA'
31480       STY     BLKOFS
31490       LDA     (DEREG),Y       ;GET LOG. BLOCK NO.
```

```
31500       BEQ     WTR$$$          ;IF 0,GO TO FIND EMPTY BLOCK
31510       JSR     GTRLBS          ;NO,GET CORRES. PHY. BLOCK NO.
31520       WTRC$$=*
31530       JSR     GTTKSC          ;GET TRACK# & SECTOR#
31540       LDY     #$50
31550       STY     FLAG            ;SET 'WRITE' TO FLAG
31560       JSR     RWSCDF          ;WRITE A RECORD
31570       LDY     #$0E
31580       LDA     #$C0
31590       STA     (DEREG),Y       ;SET 'WRITTEN' FLAG IN FCB.S2
31600       LDA     CRRCNB          ;'CR'<'RC'
31610       CMP     TLRCNB
31620       BCC     WTR$$Q          ;GO AHEAD WITH NO INCRESE 'RC'
31630       STA     TLRCNB          ;STORE 'CR' TO 'RC'
31640       INC     TLRCNB          ;INCRESE 'RC' BY 1
31650       WTR$$Q=*
31660       CMP     #$3F            ;'CR'=$3F?
31670       BNE     WT$$SQ          ;NO,MODIFY FCB & RETURN
31680       BIT     FLAG2           ;OTHERWISE, CHECK RANDOM
31690       BPL     WT$$SQ
31700       JSR     MDFFCB          ;IF SEQ. CLOSE THIS EXTENT &
31710       JSR     HDNTEN          ;OPEN NEXT
31720       LDA     RTNFLG+1
31730       BNE     W$$CSQ          ;IF UNSUCC., LEAVE 'CR'=$3F ALONE
31740       LDA     #$FF            ;IF  SUCC., SET 'CR'=$FF
31750       STA     CRRCNB
31760       W$$CSQ=*
31770       LDA     #$00
31780       STA     RTNFLG+1
31790       WT$$SQ=*
31800       JMP     MDFFCB          ;MODIFY FCB, 'CR' WILL BE $40 OR $00
31810       WTR$$$=*                ;FIND AN EMPTY BLOCK
31820       STA     TEMP
31830       LDA     #$01
31840       CPY     #$10            ;FIRST POINTER?
31850       BEQ     WT$CDSQ         ;YES, SET BASS PHY. BLOCK# TO $01
31860       DEY
31870       LDA     (DEREG),Y       ;NO, GET PHY. BLOCK# BEFORE IT
31880       BEQ     WT$CSQ          ;0?
31890       LDA     DEREG           ;NO, CONVERT IT TO BASS BLOCK#
31900       STA     HLREG1
31910       LDA     DEREG+1
31920       STA     HLREG1+1
31930       JSR     GTBKNB
31940       WT$CSQ=*
31950       JSR     FNDEPT          ;TO FIND AN EMPTY BLOCK
31960       TAX
31970       LDA     #$08            ;MAP OVERFLOW?
31980       BIT     FLAG2
31990       BEQ     WTR$SQ
32000       LDA     #$02
32010       JMP     STRTFG
32020       WTR$SQ=*
32030       TXA
```

```
32040       PHA
32050       JSR     RELFOM        ;NO,CHANGE FOUND# TO CORRES. LOGICAL#
32060       LDY     BLKOFS
32070       NOP
32080       NOP
32090       NOP
32100       TXA
32110       STA     (DEREG),Y     ;STORE  TO COORES POINTER
32120       PLA                   ;WRITE  THE RECORD TO FIRST
32130       JMP     WTRC$$        ;SECTOR OF FOUND BLOCK
32140 ;
32150 ;
32160 ;
32170 ; ************************************************************
32180 ; *                                                          *
32190 ; *  FUNCTION  25 :  WRITE SEQUENTIALLY                      *
32200 ; *                                                          *
32210 ; ************************************************************
32220 ; *                                                          *
32230 ; *  ENTRY PARAMETERS :                                      *
32240 ; *      REGISTER  X  : $19                                  *
32250 ; *      REGISTER  Y,A: FCB ADDRESS                          *
32260 ; *                                                          *
32270 ; *  RETURNED VALUE :                                        *
32280 ; *      REGISTER A : $00  SUCCESS                           *
32290 ; *                   $01  CAN'T CLOSE OR OPEN NEXT          *
32300 ; *                        EXTENT                            *
32310 ; *                   $02  MAP OVERFLOW                      *
32320 ; *                                                          *
32330 ; ************************************************************
32340 ;
32350        WTRCSQ=*
32360 ;
32370       JSR     FCBINZ
32380       JMP     WTRCS1
32390 ;
```

```
32400 ;
32410       FNDCRN=*                ;FIND CURRENT LOGIC RECORD#
32420                               ;AND EXTENT
32430 ;
32440       LDY     #$21            ;GET R0
32450       LDA     (DEREG),Y
32460       AND     #$3F            ;CACULATE DESIRED RECORD#
32470       LDY     #$20
32480       STA     (DEREG),Y       ;STORE TO FCB.CR
32490       INY
32500       LDA     (DEREG),Y       ;GET R0 AGAIN
32510       STA     TEMP1
32520       INY
32530       LDA     (DEREG),Y       ;GET R1
32540       STA     EXNTNB          ;CACULATE THE 'EX' DESIRED
32550       ROL     TEMP1
32560       ROL     EXNTNB
32570       ROL     TEMP1
32580       ROL     EXNTNB
32590       LDA     EXNTNB          ;DESIRED 'EX' = CURRENT 'EX'?
32600       LDY     #$0C
32610       CMP     (DEREG),Y
32620       BEQ     FN$CRN
32630       JSR     CLFILE          ;IN NOT, CLOSE CURRENT EXTENT
32640       LDX     RTNFLG+1        ;SUCCESS?
32650       LDA     #$03
32660       INX
32670       BEQ     FNDCR$
32680       LDA     EXNTNB          ;YES,STORE DESIRED TO FCB.CR
32690       LDY     #$0C
32700       STA     (DEREG),Y
32710       JSR     OPFILE          ;OPEN THIS EXTENT
32720       LDX     RTNFLG+1        ;SUCCESS?
32730       INX
32740       BNE     FN$CRN
32750       LDA     #$04            ;IF NOT, CHECK R OR W
32760       BIT     FLAG2
32770       BVS     FNDCR$
32780       JSR     MKFILE          ;IF W, CREATE A NEW NEXT
32790       LDA     #$05
32800       LDX     RTNFLG+1        ;SUCCESS?
32810       INX
32820       BEQ     FNDCR$          ;IF NOT REPORT ERR
32830       FN$CRN=*
32840       LDA     #$00
32850       FNDCR$=*
32860       JMP     STRTFG
32870 ;
32880       RDRCR1=*
32890 ;
32900       JSR     TETOPN
32910       LDA     #$60            ;SET FLAG TO INDICATE RANDOM R
```

```
32920         STA     FLAG2
32930         JSR     FNDCRN          ;FOND 'EX' & 'CR' DESIRED
32940         BNE     R$RCRM
32950         JMP     RDRCS$          ;IF SUCCESS, READ THE RECORD
32960         R$RCRM=*
32970         RTS
32980 ;
32990 ; ************************************************************
33000 ; *                                                          *
33010 ; *  FUNCTION  26 :  READ RANDOM                             *
33020 ; *                                                          *
33030 ; ************************************************************
33040 ; *                                                          *
33050 ; *  ENTRY PARAMETERS :                                      *
33060 ; *      REGISTER  X  : $1A                                  *
33070 ; *      REGISTER  Y,A: FCB ADDRESS                          *
33080 ; *                                                          *
33090 ; *  RETURNED VALUE :                                        *
33100 ; *      REGISTER A : $00  SUCCESS                           *
33110 ; *                   $01  READ UNWRITTEN DATA               *
33120 ; *                   $03  CAN'T CLOSE CURRENT EXTENT        *
33130 ; *                   $04  SEEK TO UNWRITTEN EXTENT          *
33140 ; *                                                          *
33150 ; ************************************************************
33160 ;
33170       RDRCRM=*
33180 ;
33190         JSR     FCBINZ
33200         JMP     RDRCR1
33210 ;
33220         WTRCR1=*
33230 ;
33240         JSR     TETOPN          ;DISK W PROTECTED?
33250         LDA     #$20            ;SET FLAG TO INDICATE RANDOM W
33260         STA     FLAG2
33270         JSR     FNDCRN          ;FIND 'EX' & 'CR' DESIRED
33280         BNE     W$RCRM
33290         JMP     WTRCS$          ;IF SUCCESS, WRITE THIS RECORD
33300         W$RCRM=*
33310         RTS
33320 ;
33330 ;
33340 ;
33350 ;
33360 ;
33370 ;
33380 ;
33390 ;
33400 ;
33410 ;
33420 ;
33430 ;
33440 ;
33450 ;
```

```
34000       STA     CBADRV+2
34010       LDA     #DFBUFA     ;IF FOUND,COPY DIR BUFFER TO
34020                           ;DEFAULT BUFFER
34030       STA     HLREG1
34040       LDA     #DFBUFA/256
34050       STA     HLREG1+1
34060       LDA     DIRBFA
34070       STA     HLREG
34080       LDA     DIRBFA+1
34090       STA     HLREG+1
34100       LDA     #$00
34110       STA     TEMP
34120       LDA     RTNFLG+1
34130       AND     #$0C
34140       LDX     #$05
34150       JSR     ASLDSR
34160       JSR     ADCOFS      ;GET THE ADDR. OF MATCHED RECORD
34170       LDY     #$00
34180       SH$IST=*
34190       LDA     (HLREG),Y   ;COPY MATCHED DIR TO DEFAULT BUFFER
34200       STA     (HLREG1),Y
34210       INY
34220       CPY     #$80
34230       BNE     SH$IST
34240       LDA     RTNFLG+1    ;GET THE OFFSET AWAY FROM THE START
34250       AND     #$03
34260       STA     RTNFLG+1
34270       RTS
34280 ;
34290 ; ***************************************************************
34300 ; *                                                             *
34310 ; *  FUNCTION  29 :  SEARCH FOR NEXT                            *
34320 ; *                                                             *
34330 ; ***************************************************************
34340 ; *                                                             *
34350 ; *  ENTRY PARAMETERS :                                         *
34360 ; *      REGISTER  X  : $1D                                     *
34370 ; *                                                             *
34380 ; *  RETURNED VALUE :                                           *
34390 ; *      REGISTER A : $FF  FILE NOT FOUND                       *
34400 ; *                   NUMBER OF DESIRED ENTRY WITHIN            *
34410 ; *                   DIRECTORY BLOCK                           *
34420 ; *                                                             *
34430 ; ***************************************************************
34440 ;
34450       SHNEXT=*
34460 ;
34470       LDA     CBADRV      ;GET RESERVED ADR
34480       STA     DEREG
34490       LDA     CBADRV+1
34500       STA     DEREG+1
34510       JSR     FCBINZ
34520       LDA     #$FF
34530       STA     FLAG1
```

```
33460 ; ************************************************************
33470 ; *                                                          *
33480 ; *  FUNCTION  27 :  WRITE RANDOM                            *
33490 ; *                                                          *
33500 ; ************************************************************
33510 ; *                                                          *
33520 ; *  ENTRY PARAMETERS :                                      *
33530 ; *      REGISTER  X  : $1B                                  *
33540 ; *      REGISTER  Y,A: FCB ADDRESS                          *
33550 ; *                                                          *
33560 ; *  RETURNED VALUE :                                        *
33570 ; *      REGISTER A : $00  SUCCESS                           *
33580 ; *                   $02  MAP OVERFLOW                      *
33590 ; *                   $03  CAN'T CLOSE CURRENT EXTENT        *
33600 ; *                   $05  DIRECTORY OVERFLOW                *
33610 ; *                                                          *
33620 ; ************************************************************
33630 ;
33640       WTRCRM=*
33650 ;
33660       JSR     FCBINZ
33670       JMP     WTRCR1
33680 ;
33690 ; ************************************************************
33700 ; *                                                          *
33710 ; *  FUNCTION  28 :  SEARCH FOR FIRST                        *
33720 ; *                                                          *
33730 ; ************************************************************
33740 ; *                                                          *
33750 ; *  ENTRY PARAMETERS :                                      *
33760 ; *      REGISTER  X  : $1C                                  *
33770 ; *      REGISTER  Y,A: FCB ADDRESS                          *
33780 ; *                                                          *
33790 ; *  RETURNED VALUE :                                        *
33800 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
33810 ; *                   NUMBER OF DESIRED ENTRY WHITHIN        *
33820 ; *                   DIRECTORY BLOCK                        *
33830 ; *                                                          *
33840 ; ************************************************************
33850 ;
33860       SHFIST=*
33870       JSR     FCBINZ      ;INITIALIZE FCB
33880       LDA     DEREG       ;RESERV FCB ADR FOR
33890                           ;SERCH NEXT
33900       STA     CBADRV
33910       LDA     DEREG+1
33920       STA     CBADRV+1
33930       LDX     #$0D        ;FIND DESIRED DIR
33940       JSR      FBDRMH
33950       BIT     FLAG        ;IF NOT FOUND,RETURN WITH A=FF
33960       BPL     S$FIST
33970       RTS
33980       S$FIST=*
33990       LDA     DRERNB
```

```
34540       STA     RTNFG1
34550       LDX     #$0D
34560       LDA     CBADRV+2
34570       JSR     FBD$$$          ;GO AHEAD TO FIND NEXT USING
34580                               ;PREVIOUS FCB AND THE PREVIOUS FOUND
34590                               ; ENTRY
34600       BIT     FLAG
34610       BPL     S$FIST          ;IF FOUND, GO TO COPY THEM
34620       RTS
34630 ;
34640       TETDUP=*
34650 ;
34660       CLC
34670       LDA     DEREG           ;CHANGE ADDR. TO FCB LOCATION AREA
34680       ADC     #$10
34690       STA     DEREG
34700       BCC     T$TDUP
34710       INC     DEREG+1
34720       T$TDUP=*
34730       LDX     #$0C
34740       JSR     FBDRMH          ;TRY TO FIND DUPLICATE
34750       BIT     FLAG
34760       BMI     TE$DUP
34770       PLA                     ;IF FOUND,REPORT ERR
34780       PLA
34790       LDA     #$80
34800       JMP     STRTFG
34810       TE$DUP=*
34820       CLC
34830       LDA     DEREG           ;NO, RECOVE THE FCB ADDR.
34840       ADC     #$F0
34850       STA     DEREG
34860       LDA     DEREG+1
34870       ADC     #$FF
34880       STA     DEREG+1
34890       RTS
34900 ;
34910       RENAMM=*
34920 ;
34930       JSR     TETWPT          ;DISK W PROTECTED?
34940       LDX     #$0C
34950       JSR     FBDRMH          ;TRY TO FIND THE OLD FILE
34960       RE$AMM=*
34970       BIT     FLAG            ;FOUND?
34980       BPL     R$NAMM
34990       RTS
35000       R$NAMM=*
35010       JSR     TETFWT          ;FILE W PROTECTED
35020       JSR     GETENT          ;GET DIR ENTRY
35030       LDA     #$FF
35040       STA     TEMP
35050       LDA     #$F0
35060       JSR     ADCOFS          ;SUBTRACT FROM $10
35070       LDX     #$0C
```

```
35080       LDY     #$10
35090       JSR     P$FBDK          ;MODIFY DIR DR-T3 ACCORDING TO FCB
35100                               ;DO-D1 AND WRITE IT BACK TO DISK DIR.
35110       JSR     FBDR$$          ;FIND NEXT EXTENT AND DEAL WITH IT
35120                               ;UNTIL ALL FILES ARE DEALT.
35130       JMP     RE$AMM
35140 ;
35150 ; ************************************************************
35160 ; *                                                          *
35170 ; *  FUNCTION  30 :  RENAME FILE                             *
35180 ; *                                                          *
35190 ; ************************************************************
35200 ; *                                                          *
35210 ; *  ENTRY PARAMETERS :                                      *
35220 ; *      REGISTER  X  : $1E                                  *
35230 ; *      REGISTER  Y,A: FCB ADDRESS                          *
35240 ; *                                                          *
35250 ; *  RETURNED VALUE :                                        *
35260 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
35270 ; *                   $80 THE NEW NAME IS ALREADY IN DIR.    *
35280 ; *                   DIR CODE FROM $00-$6F IF SUCCESS  ;    *
35290 ; *                                                          *
35300 ; ************************************************************
35310 ;
35320       RENAME=*
35330 ;
35340       JSR     FCBINZ
35350       LDY     #$00            ;COPY 'DR' IN OLD FILE TO 'DR' IN
35360       LDA     (DEREG),Y       ;NEW FILE
35370       LDY     #$10
35380       STA     (DEREG),Y
35390       JSR     TETDUP          ;TEST DUPLICATE
35400       JSR     RENAMM          ;RENAME!
35410       JMP     SETRTN
35420 ;
35430       CHATRT=*
35440 ;
35450       JSR     TETWPT
35460       LDX     #$0C            ;TRY TO FIND A ENTRY IN DIR
35470       JSR     FBDRMH
35480       CH$TRT=*
35490       BIT     FLAG
35500       BPL     C$ATRT
35510       RTS
35520       C$ATRT=*
35530       LDX     #$0C            ;COPY FCB TO DIR TO CHANGE THE
35540       JSR     PTFBDK          ;ATTR.
35550       JSR     FBDR$$          ;KEEP FINDING
35560       JMP     CH$TRT
35570 ;
35580 ;
35590 ;
35600 ;
35610 ;
```

```
35620 ; ************************************************************
35630 ; *                                                          *
35640 ; *   FUNCTION  31 :  CHANGE ATTRIBUTES                      *
35650 ; *                                                          *
35660 ; ************************************************************
35670 ; *                                                          *
35680 ; *   ENTRY  PARAMETERS  :                                   *
35690 ; *       REGISTER   X   :  $1F                              *
35700 ; *       REGISTER   Y,A :  FCB ADDR.                        *
35710 ; *                                                          *
35720 ; *   RETURNED  VALUE    :  $FF FILE NOT FOUND               *
35730 ; *                         DIR NO. IF SUCCESS               *
35740 ; *                                                          *
35750 ; ************************************************************
35760 ;
35770       CHATRB=*
35780 ;
35790       JSR     FCBINZ
35800       JSR     CHATRT
35810       JMP     SETRTN
35820 ;
35830 ; ************************************************************
35840 ; *                                                          *
35850 ; *  FUNCTION  32 :  COMPUTE FILE SIZE                       *
35860 ; *                                                          *
35870 ; ************************************************************
35880 ; *                                                          *
35890 ; *  ENTRY PARAMETERS :                                      *
35900 ; *      REGISTER  X  : $20                                  *
35910 ; *      REGISTER  Y,A: FCB ADDRESS                          *
35920 ; *                                                          *
35930 ; *  RETURNED VALUE :                                        *
35940 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
35950 ; *                   OTHERWISE R0,R1 CONTAIN FILE SIZE      *
35960 ; *                                                          *
35970 ; ************************************************************
35980 ;
35990       CPFISZ=*
36000 ;
36010       JSR     FCBINZ
36020       LDA     #$00          ;INIT. R0 &R1
36030       LDY     #$21
36040       STA     (DEREG),Y
36050       INY
36060       STA     (DEREG),Y
36070       LDX     #$0C          ;GO AHEAD TO FIND THE FILE
36080       JSR     FBDRMH
36090       CPF$SZ=*
36100       BIT     FLAG          ;FOUND?
36110       BPL     C$FISZ
36120       RTS
36130       C$FISZ=*
36140       JSR     GETENT        ;YES,SET HLREG ADDRESS OF THIS ENTRY
36150       LDY     #$0F
```

```
36160       LDA     (HLREG),Y   ;GET THE NUMBER OF TOTAL RECORDS
36170       CLC
36180       LDY     #$21        ;BE ADDED TO SUN STORED IN R0,R1
36190       ADC     (DEREG),Y
36200       STA     (DEREG),Y
36210       BCC     CP$ISZ
36220       INY
36230       LDA     #$00
36240       ADC     (DEREG),Y
36250       STA     (DEREG),Y
36260       CP$ISZ=*
36270       JSR     FBDR$$      ;KEPT FINDING UNTIL WHOLE DIRECTORY IS
36280                           ;SEARCHED
36290       JMP     CPF$SZ
36300 ;
36310 ; ************************************************************
36320 ; *                                                          *
36330 ; *  FUNCTION  33 :  SET RANDOM RECORD                       *
36340 ; *                                                          *
36350 ; ************************************************************
36360 ; *                                                          *
36370 ; *  ENTRY PARAMETERS :                                      *
36380 ; *      REGISTER  X  : $21                                  *
36390 ; *      REGISTER  Y,A: FCB ADDRESS                          *
36400 ; *                                                          *
36410 ; *  RETURNED VALUE :                                        *
36420 ; *      REGISTER A : $FF  FILE NOT FOUND                    *
36430 ; *                   OTHERWISE R0,R1 CONTAIN NUMBER OF      *
36440 ; *                   DESIRED RECORD WITH R0 BEING THE       *
36450 ; *                   LEAST SIGNIFICANT BYTE                 *
36460 ; *                                                          *
36470 ; ************************************************************
36480 ;
36490       STRDRC=*
36500 ;
36510       LDY     #$0C        ;GET 'EX'
36520       LDA     (DEREG),Y
36530       STA     TEMP1
36540       LDY     #$20        ;GET CR
36550       LDA     (DEREG),Y
36560       LSR     TEMP1       ;RIGHT SHIFT 'EX' ONCE
36570       BCC     S$RDRC
36580       NOP                 ;THAT MEANS NUMBER
36590                           ;OF RECORDS SOULD BE ADDED BY $40
36600       ADC     #$3F
36610       S$RDRC=*
36620       LSR     TEMP1       ;RIGHT SHIFT 'EX' AGAIN
36630       BCC     ST$DRC
36640       NOP                 ;NUBMER OF RECORDS
36650                           ;SHOULD BE ADDED BY $80
36660       ADC     #$7F
36670       ST$DRC=*
36680       LDY     #$21        ;NOW PUT THE LOWER BYTE OF TOTAL NUMBER
36690       STA     (DEREG),Y   ;TO THE R0
```

```
36700       INY
36710       LDA     TEMP1           ;THE HIGHER BYTE TO R1
36720       STA     (DEREG),Y
36730       RTS
36740 ;
36750 ;
36760 ;
36770 ;
36780 ;
36790 ;
36800 ;
36810 ;
36820 ;
36830 ; ************************************************************
36840 ; *                                                          *
36850 ; *  FUNCTION  34 :  SENT I/O BUFFER BACK TO DISK            *
36860 ; *                                                          *
36870 ; ************************************************************
36880 ; *                                                          *
36890 ; *  ENTRY PARAMETERS :                                      *
36900 ; *       REGISTER  X  : $22                                 *
36910 ; *                                                          *
36920 ; ************************************************************
36930 ;
36940       SIODSK=*
36950 ;
36960       JMP     SIOBDK
36970 ;
36980       RETURN=*
36990 ;
37000       LDA     RTNFLG          ;DISK NEEDS TO CHANGE BACK?
37010       BEQ     R$TURN
37020       LDA     #$00
37030       TAY
37040       STA     (DEREG),Y
37050       LDA     CRTDRN          ;STORE BACK THE 'DR'
37060       BEQ     R$TURN
37070       STA     (DEREG),Y
37080       LDA     PRVDSK          ;GET RESERVED DISK NO.
37090       STA     DERDSK
37100       JSR     SELDSK          ;CHANGE BACK TO CURRENT DISK
37110       R$TURN=*
37120       LDY     RTNFLG+2        ;RETURN THE FLAG OR ADDR. NEEDED
37130       LDA     RTNFLG+1
37140       RTS
37150       .END
```

Appendix B4 : Listing of BIOS

```
37160       ;OUP/M BIOS WRITTEN BY SHAO,JIAN-XIONG
37170       ;ON JULY 1982
37180       ;ERROR NO. LISTING
37190       ;$20  INDEX HOLE FOUND WHILE WRITING
37200       ;$21  INDEX HOLE FOUND WHILE READING
37210       ;$22  DRIVE NOT READY
37220       ;$23  TRACK NOT MATCH
37230       ;$24  DISK WRITING PROTECTION
37240       ;$25  I/O DEVICE NOT DEFINED
37250       ;$40  BUFFER AND DISK NOT MATCH
37260       ;$80  PARITY ERROR
37270       *=$E41A             ;OPBIOS START ADDR.
37280       START=$00
37290       IOBYTE=$03          ;I/O BYTE
37300       COUNTER=$04           ;PAGE NO. COUNTER
37310       DEREG=$09           ;WORK STORAGES
37320       HLREG=$0B
37330       HLREG1=$0D
37340       FLAG=$0F
37350       TEMP=$1F
37360       IOBFFG=$39          ;'FLUSH' FLAG
37370       CRTDSK=$3A          ;CURRENT DISK NO.
37380       BLKNB1=$42          ;BLOCK#
37390       SECNB1=$43          ;SECTOR#
37400       TKNDER=$44          ;PREVIOUS TRACK# WHICH THEN
37410                           ;CHANGE TO DESIRED TRACK#
37420       TRKDRC=$45          ;DESIRED TRACK#
37430       DMAREG=$46          ;DMA ADDRESS
37440       TRKCRN=$48          ;HEAD POSITION TRACK#
37450       TKTMAX=$49          ;MAX TIMES FOR MATCHING TRACK#
37460       ERORFG=$4A          ;ERROR FLAG
37470       STATFG=$4B          ;STATE FLAG
37480       IOBUFA=$4C
37490       PAGNUB=$4E          ;PAGE COUNT
37500       ERTMAX=$4F          ;R/W MAX TIMES
37510       DVACIA=$C010        ;DISK I/O PORT
37520       DVPIA1=$C000        ;DISK CONTROL
37530       DVPIA2=$C002        ;DISK CONTROL
37540       CLACIA=$FC00        ;CONSOLE I/O PORT
37550       PTACIA=$CF02        ;PRINTER I/O PORT
37560       SLACIA=$CF00        ;SERIAL COMMUNICATION I/O PORT
37570       USACI1=$FB03        ;USER USING I/O PORT
37580       USACI2=$F400        ;USER USING I/O PORT
37590       IOBUF=$B000         ;DISK I/O BUFFER ADDR.
37600       CCPSAT=$D000        ;CCP ENTRY NO. 1
37610       CCPST1=$D3F5        ;CCP ENTRY NO. 2
37620       BDOS=$D93F          ;BDOS ENTRY
37630       WHBCOT     .BYTE $28
37640       FRYCST     .BYTE $62
37650       DKCONT     .BYTE $40,$00
37660       DIKCRT     .BYTE  $00
37670       PAMTAB   .WORD DIRBUF,MAPARO,CHKARO,$0000  ;DIAK PARA.
```

```
37680              .WORD DIRBUF,MAPAR1,CHKAR1,$0000  ;HEAD TABLE
37690 ;
37700 ; ************************************************************
37710 ; *                                                          *
37720 ; *  NUMBER  0 :  INITIALIZE  INPUT/OUTPUT PORT              *
37730 ; *                                                          *
37740 ; ************************************************************
37750 ;
37760       BOOT=*
37770 ;
37780       LDA     #$80          ;SET IOBYTE TO INDICATE SELECTING
37790                             ;PRINTER IN LIST FIELD AND
37800       STA     IOBYTE        ;SELECTING CONSOLE IN CONSLE FIELD
37810       LDY     #$00          ;SET DVPIA1+1 BIT 2 TO ZERO,INDICAT
37820                             ;DVPIA1 IS DIRECTION REGISTER NOW.
37830       STA     DVPIA1+1
37840       LDY     #$40          ;SET DVPIA1 BIT 6   OUTPUT AND REST
37850                             ;INPUT
37860       STY     DVPIA1
37870       LDY     #$04          ;SET BACK INDICATING DVPIA1 IS
37880                             ;CONTROL GATE NOW
37890       STY     DVPIA1+1
37900       LDA     #$03          ;INITIAIZE CONSOLE PORT
37910       STA     CLACIA
37920       LDY     #$11
37930       STY     CLACIA
37940       STA     PTACIA        ;INITIALIZE PRINTER PORT
37950       STY     PTACIA
37960       STA     SLACIA        ;INITIALIZE SERIAL COMMUN. PORT
37970       STY     SLACIA
37980       LDA     USACI1+3      ;INITIALIZE USER PORT1
37990       LDX     #$FF
38000       STX     USACI1+2
38010       LDY     #$00          ;INITIALIZE USER PORT2
38020       STY     USACI2+1
38030       STY     USACI2
38040       STY     USACI2+3
38050       STX     USACI2+2
38060       LDA     #$04
38070       STA     USACI2+1
38080       STA     USACI2+3
38090       RTS
38100 ;
38110 ; ************************************************************
38120 ; *                                                          *
38130 ; *  NUMBER  1 :  WARM  BOOT                                 *
38140 ; *                                                          *
38150 ; ************************************************************
38160 ;
38170       WBOOT=*
38180 ;
38190       NOP
38200       LDA     #CCPSAT       ;SET STARTING ADDR. OF CCP FOR DMA
38210       LDX     #CCPSAT/256
```

```
38220       JSR     STDMA1
38230       LDA     #$00            ;SELECT DRIVE A
38240       STA     DIKCRT
38250       JSR     DIKSEL
38260       LDA     #$01            ;SELECT TRACK 1
38270       STA     TRKDRC
38280       LDA     #$00            ;SELECT SECTOR 00
38290       STA     SECNB1
38300       LDA     #$10            ;SET READ SECTOR FLAG
38310       STA     FLAG
38320       W$OOT=*
38330       JSR     RWDBS1          ;READ A SECTOR FROM DISK
38340       CLC
38350       LDA     #$80            ;NO INCREASE DMAREG TO GET NEXT
38360                               ;SECTOR
38370       ADC     DMAREG
38380       STA     DMAREG
38390       BCC     WB$OT
38400       INC     DMAREG+1
38410       WB$OT=*
38420       INC     SECNB1          ;SET SECTOR NUMBER WITHIN ONE TRACK
38430       LDA     SECNB1
38440       CMP     #$1C            ;ONE TRACK IS DONE?
38450       BNE     W$OOT           ;IF NOT,KEPT READING
38460       LDA     #DFBUFA         ;RECOVE DEFAULT BUFFER
38470       LDX     #DFBUFA/256
38480       JSR     STDMA1
38490       LDA     #$4C
38500       STA     START+5         ;SET JMP BDOS AT$05
38510       STA     START           ;SET JMP WBOOT AT $00
38520       LDA     #WBOOT
38530       STA     START+1
38540       LDA     #WBOOT/256
38550       STA     START+2
38560       LDA     #BDOS
38570       STA     START+6
38580       LDA     #BDOS/256
38590       STA     START+7
38600       LDA     #$00
38610       STA     IOBFFG
38620       JMP     CCPSAT
38630 ;
38640       CONTST=*                ;CONSOLE TEST FOR RECEIVE READY
38650 ;
38660       LDA     CLACIA          ;TEST BIT 0 OF CLACIA FOR READY
38670       AND     #$01
38680       BNE     C$LTST
38690       RTS                     ;IF NOT READY,RETURN WITH A=$00
38700       C$LTST=*
38710       LDA     #$FF            ;IF READY,RETURN WITH A=$FF
38720       RTS
38730 ;
38740       CONIN=*                 ;READ A CHARACTER FROM CONSOLE
38750 ;
```

```
38760       JSR     CONTST          ;TEST FOR CONSOLE'S READY,IF NOT
38770       BEQ     CONIN           ;KEPT TESTING
38780       LDA     CLACIA+1        ;IF READY,READ A CHARACTER
38790       AND     #$7F            ;MASK OFF IT'S BIT 7
38800       RTS
38810 ;
38820       CONTOT=*                ;CONSOLE TEST FOR TRANSMIT READY
38830 ;
38840       LDA     CLACIA          ;TEST BIT1 OF CLACIA FOR READY
38850       AND     #$02
38860       BNE     C$NTOT
38870       RTS                     ;IF NOT READY,RETURN WITH A=$00
38880       C$NTOT=*
38890       LDA     #$FF            ;IF READY,RETURN WITH A=$FF
38900       RTS
38910 ;
38920       CONOUT=*                ;OUTPUT   A CHARACTER TO CONSOLE
38930 ;
38940       JSR     CONTOT          ;KEPT TESTING UNTIL IS READY
38950       BEQ     CONOUT
38960       STX     CLACIA+1        ;SENT A CHARACTER TO CONSOLE
38970       RTS
38980 ;
38990       PRTTST=*                ;PRINTER TEST FOR TRANSMIT READY
39000 ;
39010       LDA     PTACIA          ;TEST BIT1 OF PTACIA FOR READY
39020       AND     #$02
39030       BNE     P$TTST
39040       RTS                     ;IF NOT READY,RETURN WITH A=$00
39050       P$TTST=*
39060       LDA     #$FF            ;IF READY, RETURN WITH A=$FF
39070       RTS
39080 ;
39090       PRTOUT=*                ;PRINTER OUTPUT
39100 ;
39110       JSR     PRTTST          ;KEPT TESTING,UNTIL IS READY
39120       BEQ     PRTOUT
39130       STX     PTACIA+1        ;OUTPUT A CHARACTER
39140       LDA     PTACIA          ;TEST FOR RECEIVING READY
39150       AND     #$01
39160       BNE     P$TOUT
39170       RTS                     ;IF NOT,RETURN
39180       P$TOUT=*
39190       LDA     PTACIA+1        ;IF READY,READ A CHARACTER
39200       CMP     #$13            ;CONTROL-S
39210       BNE     PR$OUT          ;IF NOT CONTROL-S RETURN
39220       RTS
39230       PR$OUT=*
39240       LDA     PTACIA          ;IF CONTROL-S WAIT UNTL CONTROL-Q
39250       AND     #$01            ;IS RECEIVED
39260       BEQ     PR$OUT
39270       CMP     #$11
39280       BNE     PR$OUT
39290       RTS
```

```
39300 ;
39310       SRLTST=*                ;SERIAL COMMUN. RECEIVE READY TEST
39320 ;
39330       LDA     SLACIA          ;THE BIT 0 OF SLACIA FOR READY
39340       AND     #$01
39350       BNE     S$LTST          ;NOT,RETURN WITH A=$00
39360       RTS
39370       S$LTST=*
39380       LDA     #$FF            ;READY,RETURN WITH A=$FF
39390       RTS
39400 ;
39410       SRLIN=*
39420 ;
39430       JSR     SRLTST          ;KEPT TESTING,UNTIL IS READY
39440       BEQ     SRLIN
39450       LDA     SLACIA+1        ;READ A CHARACTER
39460       AND     #$7F            ;MASK OFF IT'S BIT 7
39470       RTS
39480 ;
39490       SRLTOT=*                ;SERIAL COMMUN. TRANSMIT READY TEST
39500 ;
39510       LDA     SLACIA          ;TEST BIT 1 OF SLACIA FOR READY
39520       AND     #$02
39530       BNE     S$LTOT
39540       RTS                     ;IF NOT,RETURN WITH A=$00
39550       S$LTOT=*
39560       LDA     #$FF            ;IF READY,RETURN WITH A=$FF
39570       RTS
39580 ;
39590       SRLOUT=*
39600 ;
39610       JSR     SRLTOT          ;KEPT TESTING,UNTIL SLACIA IS
39620                               ;READY
39630       BEQ     SRLOUT
39640       STX     SLACIA+1        ;OUTPUT A CHARACTER
39650       RTS
39660 ;
39670 ;
39680       OUTBFF=*
39690 ;
39700       LDY     #$00
39710       OU$BFF=*
39720       LDA     (DEREG),Y
39730       CMP     #$24
39740       BNE     O$TBFF
39750       RTS
39760       O$TBFF=*
39770       TAX
39780       JSR     CONOUT
39790       INY
39800       BNE     OU$BFF
39810 ;
39820 ;
39830 ;
```

```
39840 ; ************************************************************
39850 ; *                                                          *
39860 ; *  NUMBER  2 :  SAMPLE THE STATUS OF THE CURRENTLY         *
39870 ; *               ASSIGNED CONSOLE DEVICE                    *
39880 ; *                                                          *
39890 ; ************************************************************
39900 ; *                                                          *
39910 ; *  RETURNED VALUE :                                        *
39920 ; *      REGISTER A : $FF  READY TO READ                     *
39930 ; *                   $00  NOT READY                         *
39940 ; *                                                          *
39950 ; ************************************************************
39960 ;
39970       CSLTET=*
39980 ;
39990       LDA     IOBYTE        ;GET THE CONSLE FIELD OF IOBYTE
40000       AND     #$03
40010       BNE     C$LTET
40020       JMP     CONTST        ;IF #$00 TO CONSOLE TEST
40030       C$LTET=*
40040       CMP     #$01
40050       BNE     CS$TET
40060       JMP     PRTTST        ;IF #$01 TO PRINT TEST
40070       CS$TET=*
40080       CMP     #$03
40090       BNE     CSL$ET
40100       JMP     USRTST        ;IF #$03 TO USER TEST
40110       CSL$ET=*
40120       JMP     SRLTST        ;IF #$02 TO SERIAL TEST
40130 ;
40140 ; ************************************************************
40150 ; *                                                          *
40160 ; *  NUMBER  3 :  READ FROM THE CURRENT ASSIGNED CONSLE      *
40170 ; *               DEVICE                                     *
40180 ; *                                                          *
40190 ; ************************************************************
40200 ; *                                                          *
40210 ; *  RETURNED VALUE  :                                       *
40220 ; *      REGISTER A  :  CHARACTER TO BE READ IN              *
40230 ; *                                                          *
40240 ; ************************************************************
40250 ;
40260       CLINN=*
40270 ;
40280       LDA     IOBYTE        ;GET THE CONSOLE FIELD OF IOBYTE
40290       AND     #$03
40300       BNE     C$INN
40310       CLIN$=*
40320       JMP     CONIN         ;IF #$00 TO CONSOLE IN
40330       C$INN=*
40340       CMP     #$01
40350       BNE     CL$NN
40360       JMP     SRLIN         ;IF #$01 TO SERIAL COMMUNICATON IN
40370       CL$NN=*
```

```
40380       CMP     #$03
40390       BNE     CLIN$           ;IF #$02 TO CONSOLE IN
40400       JMP     USRIN           ;IF #$30 TO USERIN
40410 ;
40420 ; ************************************************************
40430 ; *                                                          *
40440 ; *  NUMBER  4 :  OUTPUT CHARACTER TO THE CURRENTLY          *
40450 ; *               ASSIGNED CONSOLE DEVICES                   *
40460 ; *                                                          *
40470 ; ************************************************************
40480 ; *                                                          *
40490 ; *  ENTRY PARAMETERS :                                      *
40500 ; *      REGISTER A : CHARACTER TO BE OUTPUT                 *
40510 ; *                                                          *
40520 ; ************************************************************
40530 ;
40540       CLOUTT=*
40550 ;
40560       TAX
40570       LDA     IOBYTE          ;GET CONSOLE FIELD OF IOBYTE
40580       AND     #$03
40590       BNE     C$OUTT
40600       CLOU$T=*
40610       JMP     CONOUT          ;IF #$00 TO CONSLE OUT
40620       C$OUTT=*
40630       CMP     #$01
40640       BNE     CL$UTT
40650       JMP     SRLOUT          ;IF #$01 TO SERIAL COMMUNICATON OUT
40660       CL$UTT=*
40670       CMP     #$03
40680       BNE     CLOU$T          ;IF #$02 TO CONSOLE OUT
40690       JMP     USROUT          ;IF #$03 TO USER OUT
40700 ;
40710 ; ************************************************************
40720 ; *                                                          *
40730 ; *  NUMBER  5  :  OUTPUT CHARACTER TO THE CURRENTLY         *
40740 ; *                ASSIGNED LIST DEVICE                      *
40750 ; *                                                          *
40760 ; ************************************************************
40770 ; *                                                          *
40780 ; *  ENTRY PARAMETERS :                                      *
40790 ; *      REGISTER A :  CHARACTER TO BE OUTPUT                *
40800 ; *                                                          *
40810 ; ************************************************************
40820 ;
40830       LITOUT=*
40840 ;
40850       TAX
40860       LDA     IOBYTE          ;GET LIST FIELD OF IOBYTE
40870       AND     #$C0
40880       BNE     L$TOUT
40890       LITO$T=*
40900       JMP     CONOUT          ;IF #$00 TO CONSLE OUT
40910       L$TOUT=*
```

```
40920       CMP     #$01
40930       BNE     LI$OUT
40940       JMP     SRLOUT          ;IF #$01 TO SERIAL COMMUN. OUT
40950       LI$OUT=*
40960       CMP     #$03
40970       BEQ     LITO$T          ;IF #$03 TO CONSOLE
40980       JMP     PRTOUT          ;IF #$02 TO PRINTER OUT
40990 ;
41000 ; ********************************************************
41010 ; *                                                      *
41020 ; *  NUMBER  6 :  OUTPUT A CHARACTER TO THE CURRENTLY    *
41030 ; *               ASSIGNED PUNSH DEVICE                  *
41040 ; *                                                      *
41050 ; ********************************************************
41060 ; *                                                      *
41070 ; *  ENTRY PARAMETERS :                                  *
41080 ; *      REGISTER A : A CHARACTER TO BE OUTPUT           *
41090 ; *                                                      *
41100 ; ********************************************************
41110 ;
41120       PUHOUT=*            ;RESERVE FOR FURTHER USE
41130 ;
41140 ; ********************************************************
41150 ; *                                                      *
41160 ; *  NUMBER  7 :  READ A CHARACTER FROM THE CURRENTYLY    *
41170 ; *               ASSIGNED READER DEVICE                 *
41180 ; *                                                      *
41190 ; ********************************************************
41200 ; *                                                      *
41210 ; *  RETURNED VALUE :                                    *
41220 ; *      REGISTER A :  A CHARACTER BE READ IN            *
41230 ; *                                                      *
41240 ; ********************************************************
41250 ;
41260       READIN=*            ;RESERVE FOR FURTHER USE
41270 ;
41280       USRTST=*            ;RESERVE FOR FURTHER USE
41290       USROUT=*            ;AS ABOVE
41300       USRIN=*             ;AS ABOVE
41310       LDA     #$25
41320       JMP     ERROR
41330 ;
41340       DELAYB=*            ;DELAY ROUTINE
41350 ;
41360       INC     DELYMR
41370       INC     DELYMR
41380       INC     DELYMR
41390       INC     DELYMR
41400       RTS
41410       DELYMR*=*+1
41420 ;
41430 ;
41440       DELYMS=*            ;DELAY 1 MS
41450 ;
```

```
41460       JSR     DELAYB
41470       DEY
41480       BNE     DELYMS
41490       RTS
41500 ;
41510       DLYMS1=*            ;DELAY (X) MS
41520 ;
41530       LDX     #$0C
41540       D$YMS1=*
41550       LDY     FRYCST
41560       JSR     DELYMS
41570       DEX
41580       BNE     D$YMS1
41590       DL$MS1=*
41600       RTS
41610 ;
41620       DELYUS=*            ;DELAY (X)*100 US
41630 ;
41640       LDA     FRYCST
41650       D$LYUS=*
41660       BIT     $00
41670       SEC
41680       SBC     #$05
41690       BCS     D$LYUS
41700       DEX
41710       BNE     DELYUS
41720       RTS
41730 ;
41740 ; ************************************************************
41750 ; *                                                          *
41760 ; *  NUMBER  8 :  POSITION  HEAD  TO  TRACK  0               *
41770 ; *                                                          *
41780 ; ************************************************************
41790 ;
41800       HOMEHD=*            ;HOME HEAD TO ZERO
41810 ;
41820       JSR     STPIN
41830       JSR     DLYMS1
41840       STY     TRKCRN
41850       STY     TKNDER
41860       H$MEHD=*
41870       LDA     #$02
41880       BIT     DVPIA1
41890       BEQ     DLYMS1
41900       JSR     STPOUT
41910       BEQ     H$MEHD
41920 ;
41930       ASLDS1=             ;(A) * 2'S POWER OF (X)
41940 ;
41950       ASL     A
41960       ROL     TEMP
41970       DEX
41980       BNE     ASLDS1
41990       RTS
```

```
42000 ;
42010       ADCOF1=*
42020 ;
42030       CLC
42040       ADC     HLREG
42050       STA     HLREG
42060       LDA     TEMP
42070       ADC     HLREG+1
42080       STA     HLREG+1
42090       RTS
42100 ;
42110       ADCIOF=*
42120 ;
42130       LDX     #IOBUF
42140       STX     HLREG
42150       LDX     #IOBUF/256
42160       STX     HLREG+1
42170       JMP     ADCOF1
42180 ;
42190       DIKSEL=             ;GET DESIRED DISK PARAMETERS
42200 ;
42210       LDX     DIKCRT
42220       LDA     DKCONT,X
42230       STA     DVPIA1
42240       LDA     #$FF
42250       STA     DVPIA2
42260       D$KSEL=*
42270       LDA     #$10
42280       CPX     #$01
42290       BEQ     DI$SEL
42300       LDA     #$01
42310       DI$SEL=*
42320       BIT     DVPIA1
42330       RTS
42340 ;
42350 ; ************************************************************
42360 ; *                                                          *
42370 ; *  NUMBER  9 :  SELECT  DISK                               *
42380 ; *                                                          *
42390 ; ************************************************************
42400 ; *                                                          *
42410 ; *  ENTRY PARAMETERS :                                      *
42420 ; *      REGISTER A : DISK NUMBER                            *
42430 ; *                                                          *
42440 ; ************************************************************
42450 ;
42460       DKSEL1=*
42470 ;
42480       STA     DIKCRT          ;STORE DESIRED DISK NO.
42490       LDX     #$03            ;*8 TO GET OFFSET
42500       JSR     ASLDS1
42510       LDX     #PAMTAB
42520       STX     HLREG
42530       LDX     #PAMTAB/256
```

```
42540       STX     HLREG+1
42550       LDX     #$00
42560       STX     TEMP
42570       JSR     ADCOF1          ;GET DESIRED ENTRY ADDR.
42580       JSR     DIKSEL          ;GET PARA. & TEST DISK READY
42590       BNE     D$SEL1
42600       JMP     HOMEHD          ;READY! HOME HEAD
42610       D$SEL1=*
42620       JMP     TRKPS1          ;IF DISK NOT READY,ERR
42630 ;
42640 ; ************************************************************
42650 ; *                                                          *
42660 ; *  NUMBER  10 :  SET  TRACK  NUMBER                        *
42670 ; *                                                          *
42680 ; ************************************************************
42690 ; *                                                          *
42700 ; *  ENTRY PARAMETERS :                                      *
42710 ; *      REGISTER A :  TRACK NUMBER                          *
42720 ; *                                                          *
42730 ; ************************************************************
42740 ;
42750       SETTR1=*
42760 ;
42770       STA     TRKDRC
42780       RTS
42790 ;
42800 ; ************************************************************
42810 ; *                                                          *
42820 ; *  NUMBER  11 :  SET  SECTOR  NUMBER                       *
42830 ; *                                                          *
42840 ; ************************************************************
42850 ; *                                                          *
42860 ; *  ENTRY PARAMETERS :                                      *
42870 ; *      REGISTER A :  SECTOR  NUMBER                        *
42880 ; *                                                          *
42890 ; ************************************************************
42900 ;
42910       SETSC1=*
42920 ;
42930       STA     SECNB1
42940       RTS
42950 ;
42960 ; ************************************************************
42970 ; *                                                          *
42980 ; *  NUMBER  12 :  SET  BLOCK  NUMBER                        *
42990 ; *                                                          *
43000 ; ************************************************************
43010 ; *                                                          *
43020 ; *  ENTRY PARAMETERS :                                      *
43030 ; *      REGISTER A :  BLOCK  NUMBER                         *
43040 ; *                                                          *
43050 ; ************************************************************
43060 ;
43070       SETBL1=*
```

```
43080 ;
43090       STA     BLKNB1
43100       RTS
43110 ;
43120 ; ***********************************************************
43130 ; *                                                         *
43140 ; *  NUMBER  13 :  SET  DMA                                 *
43150 ; *                                                         *
43160 ; ***********************************************************
43170 ; *                                                         *
43180 ; *  ENTRY PARAMETERS :                                     *
43190 ; *      REGISTER A,X :  DMA  ADDRESS                       *
43200 ; *                                                         *
43210 ; ***********************************************************
43220 ;
43230       STDMA1=*
43240 ;
43250       STA     DMAREG
43260       STX     DMAREG+1
43270       RTS
43280 ;
43290       SEKIHL=*                ;SEEK INDEX HOLE
43300 ;
43310       LDA     DVPIA1
43320       BMI     SEKIHL
43330       S$KIHL=*
43340       LDA     DVPIA1
43350       BPL     S$KIHL
43360       RTS
43370 ;
43380       STPOUT=*                ;STEP OUT ONE TRACK
43390 ;
43400       LDA     DVPIA2          ;SET BIT 2 OF DVPIA2
43410       ORA     #$04
43420       BNE     STEP
43430       STPIN=*                 ;STEP IN ONE TRACK
43440       LDA     #$FB
43450       AND     DVPIA2          ;RESET BIT 2 OF DVPIA2
43460       STEP=*
43470       STA     DVPIA2          ;STEP IN OR OUT ACCORDINGLY
43480       JSR     DL$MS1          ;DELAY!
43490       AND     #$F7            ;SENT A PULSE TO STEP HEAD
43500       STA     DVPIA2
43510       JSR     DL$MS1
43520       JSR     DL$MS1
43530       ORA     #$08
43540       STA     DVPIA2
43550       JSR     DL$MS1          ;DELAY!
43560       LDX     #$08
43570       JMP     D$YMS1          ;DELAY!
43580 ;
43590       SIHSDK=*                ;SEEK HOLE ,SET ACIA
43600 ;
43610       JSR     SEKIHL
```

```
43620       LDA     #$03
43630       STA     DVACIA
43640       LDA     #$58
43650       STA     DVACIA
43660       RTS
43670 ;
43680       LDHEAD=*                ;LOAD HEAD
43690 ;
43700       LDA     #$7F            ;RESET BIT 7 OF DVPIA2
43710       AND     DVPIA2
43720       L$HEAD=*
43730       STA     DVPIA2          ;LOAD OR UNLOAD HEAD ACCORDINGLY
43740       LDX     #$28            ;DELAY!
43750       JMP     D$YMS1
43760 ;
43770       UNLDHD=*                ;UNLOAD HEAD
43780       LDA     #$80            ;SET BIT 7 OF DVAPIA2
43790       ORA     DVPIA2
43800       BNE     L$HEAD
43810 ;
43820       WRTDIK=*                ;WRITE A BYTE TO DISK
43830 ;
43840       LDA     DVPIA1          ;INDEX HOLE FOUND?
43850       BPL     W$TDIK
43860       LDA     DVACIA          ;READY?
43870       LSR     A
43880       LSR     A
43890       BCC     WRTDIK          ;IF NO,KEEP TESTING!
43900       STX     DVACIA+1        ;WRITE A BYTE TO DISK
43910       RTS
43920       W$TDIK=*
43930       LDA     #$20            ;IF INDEX HOLE FOUND, ERR!
43940       JMP     ERROR
43950 ;
43960       REDDIK=*                ;READ ABYTE FROM DISK
43970 ;
43980       LDA     DVPIA1          ;INDEX HOLE FOUND?
43990       BPL     R$DDIK          ;FOUND, ERROR!
44000       LDA     DVACIA          ;READY?
44010       LSR     A
44020       BCC     REDDIK          ;IF NO, KEEP TESTING
44030       LDA     DVACIA+1        ;READ IN A CHAR.
44040       RTS
44050       R$DDIK=*
44060       LDA     #$21            ;INDEX HOLE FOUND,ERROR!
44070       JMP     ERROR
44080 ;
44090       TRKPSA=*                ;PUT HEAD TO DESIRED TRACK
44100 ;
44110       LDX     DIKCRT          ;DESIRED DISK READY?
44120       JSR     D$KSEL
44130       BEQ     T$KPSA
44140       TRKPS1=*
44150       LDA     #$22            ;NOT READY,ERROR!
```

```
44160       JMP     ERROR
44170       T$KPSA=*
44180       LDA     TKNDER          ;CURRENT DESIRED TRACK =
44190       CMP     TRKCRN          ;CURRENT HEAD POSITION?
44200       BEQ     TR$PSA          ;IF EQUAL, FINISH STEPING
44210       BCS     TRK$SA          ;IF GREATER,STEP IN ONE TRACK
44220       JSR     STPOUT          ;OIF LESS,STEP OUT ONE TARACK
44230       LDA     #$99            ;AND SET 'A' TO 99 INDICATING
44240       BCC     TRKP$A          ;SUBTRACTING ONE TRACK
44250       TRK$SA=*
44260       JSR     STPIN
44270       TXA
44280       TRKP$A=*
44290       SED
44300       ADC     TRKCRN          ;+ (OR -) ONE TO (OR FROM)
44310       STA     TRKCRN          ;CURRENT HEAD POSITION
44320       CLD
44330       JMP     T$KPSA          ;KEEP STEPING UNTIL DONE
44340       TR$PSA=*
44350       CMP     #$43            ;CHECK CURRENT TRACK RANGE
44360       LDA     DVPIA2          ;IF IN NO.43-76, RESET BIT 7
44370       AND     #$BF            ;OF DVPIA2 TO ADJUST CURRENT
44380       LDY     #$00
44390       NOP
44400       BCS     TRKPS$
44410       LDA     #$40            ;IF IN NO.0-42, SET BIT 7
44420       ORA     DVPIA2          ;OF DVACIA
44430       TRKPS$=*
44440       STA     DVPIA2
44450       RTS
44460 ;
44470       TRKPSH=*
44480 ;
44490       JSR     TRKPSA          ;POSITION HEAD
44500       LDA     #IOBUF          ;SET DISK I/O BUFFER ADDR.
44510       STA     IOBUFA
44520       LDA     #IOBUF/256
44530       STA     IOBUFA+1
44540       JMP     LDHEAD          ;LOAD HEAD
44550 ;
44560       TKNMTH=*                ;MATCH TRACK#
44570 ;
44580       LDA     #$05            ;SET MAX. TRACK READ TIMES
44590       STA     TKTMAX
44600       TKNM$H=*
44610       JSR     SIHSDK          ;SEEK INDEX HOLE, SET DVACIA
44620       T$NMTH=*
44630       JSR     REDDIK          ;READ A BYTE INTO 'A'
44640       TK$MTH=*
44650       CMP     #$43            ;FIND TRACK IDENTIFICATION
44660       BNE     T$NMTH          ;NO. $43 & $57
44670       JSR     REDDIK
44680       CMP     #$57
44690       BNE     TK$MTH
```

```
44700       JSR     REDDIK          ;CHECK TRACK NO.
44710       CMP     TKNDER
44720       BNE     TKN$TH
44730       RTS                     ;IF MATCH,RETURN!
44740       TKN$TH=*                ;IF NOT,HOME HEAD AND TRY AGAIN
44750       DEC     TKTMAX          ;MAX. TIME EXCEED?
44760       BEQ     TKNMT$
44770       LDA     TKNDER          ;NO, SAVE CURRENT DESIRED NO.
44780       PHA
44790       JSR     HOMEHD          ;HOME HEAD!
44800       PLA
44810       STA     TKNDER          ;RECOVE TRACK NO.
44820       JSR     T$KPSA          ;POSITION HEAD AGAIN
44830       JMP     TKNM$H          ;MATCHT THEM AGAIN!
44840       TKNMT$=*
44850       LDA     #$23            ;MAX. EXHAUSTED, ERROR!
44860       JMP     ERROR
44870 ;
44880       WRTTKS=*                ;WRITE A TRACK TO DISK
44890 ;
44900       LDA     #$20            ;INDEX HOLE FOUND!
44910       BIT     DVPIA1
44920       BNE     W$TTKS
44930       LDA     #$24            ;IF FOUND, ERROR!
44940       JMP     ERROR
44950       W$TTKS=*
44960       JSR     TKNMTH          ;CHECK IF TRACK NO. ARE MATCHED
44970       LDX     #$04            ;DELAY!
44980       JSR     DELYUS
44990       LDA     #$FE            ;ENABLE ERASING
45000       AND     DVPIA2
45010       STA     DVPIA2
45020       LDX     #$02            ;DELAY!
45030       JSR     DELYUS
45040       LDA     #$FD            ;ENABLE WRITING
45050       AND     DVPIA2
45060       STA     DVPIA2
45070       LDX     #$08            ;DELAY!
45080       JSR     DELYUS
45090       LDX     #$76            ;WRITE START FLAG $76
45100       JSR     WRTDIK
45110       LDX     #$0E            ;SET 14 PAGES
45120       STX     PAGNUB
45130       LDY     #$00
45140       WR$TKS=*
45150       LDA     (IOBUFA),Y      ;GET A BYTE FROM DISK I/O
45160       TAX                     ;BUFFER AND WRITE IT
45170       JSR     WRTDIK          ;TO THE DISK
45180       INY
45190       BNE     WR$TKS
45200       INC     IOBUFA+1
45210       DEC     PAGNUB          ;KEEP DOING UNTIL 14 PAGE DONE
45220       BNE     WR$TKS
45230       LDX     #$47            ;WRITE END FLAG
```

```
45240       JSR     WRTDIK
45250       LDX     #$53
45260       JSR     WRTDIK
45270       LDA     DVPIA2          ;DISABLE WRITING
45280       ORA     #$01
45290       STA     DVPIA2
45300       LDX     #$05            ;DELAY!
45310       JSR     DELYUS
45320       LDA     DVPIA2          ;DISABLE ERASING
45330       ORA     #$02
45340       STA     DVPIA2
45350       LDA     #IOBUF/256
45360       STA     IOBUFA+1        ;RECOVE I/O BUFFER ADDR.
45370       RTS
45380 ;
45390       REDTKS=*                ;READ A TRACK
45400 ;
45410       JSR     TKNMTH          ;MATCH TRACK NO.
45420       R$DTKS=*
45430       JSR     REDDIK          ;FIND DATA AREA START FLAG
45440       CMP     #$76
45450       BNE     R$DTKS
45460       LDX     #$0E            ;SET TOTAL PAGE NO. 14
45470       LDY     #$00
45480       STY     ERORFG          ;INIT ERR FLAG
45490       LDA     #$01
45500       RE$TKS=*
45510       BIT     DVACIA          ;READY?
45520       BEQ     RE$TKS
45530       LDA     DVACIA+1        ;READ A BYTE
45540       BVC     RED$KS          ;PARITY ERR?
45550       LDA     #$80            ;IF IS, SET ERR FLAG
45560       STA     ERORFG
45570       R$$TKS=*
45580       CLC                     ;RESET CARRY TO INDICATE ERR
45590       RTS
45600       RED$KS=*
45610       BIT     STATFG
45620       BPL     REDTK$
45630       CMP     (IOBUFA),Y      ;YES, COMPARE IT WITH THE ONE
45640       BEQ     REDTK$          ;IN THE I/O BUFFER READ IN BEFORE
45650       LDA     #$40            ;IF NOT, SET ERR FLAG
45660       STA     ERORFG
45670       BNE     R$$TKS
45680       REDTK$=*
45690       STA     (IOBUFA),Y
45700       LDA     #$01
45710       INY
45720       BNE     RE$TKS          ;ONE PAGE DONE?
45730       INC     IOBUFA+1        ;YES, INCREAMENT HIGH BYTE
45740       DEX
45750       BNE     RE$TKS          ;14 PAGE DONE?
45760       LDA     #IOBUF/256
45770       STA     IOBUFA+1        ;RECOVE I/O BUFFER ADDR.
```

```
45780       SEC                 ;SET CARRY INDICATING NO ERR
45790       RTS
45800 ;
45810       WRTTRK=*            ;WRITE A TRACK UTILL NO ERROR
45820 ;
45830       LDA     #$A0        ;SET NO REREAD MODE
45840       STA     STATFG
45850       JSR     TRKPSH      ;POSITION HEAD TO DESIRED TRACK
45860       LDA     #$04        ;SET MAX. WRITE TIMES
45870       STA     ERTMAX+1
45880       W$TTRK=*
45890       JSR     WRTTKS      ;WRITE A TRACK
45900       JMP     RE$TRK      ;GO TO REREAD MODE TO CHECK
45910 ;
45920       REDTRK=*            ;;READ A TRACK UNTIL NO ERROR
45930 ;
45940       LDA     #$40        ;SET NO REREAD MODE
45950       STA     STATFG
45960       JSR     TRKPSH      ;POSITION HEAD TO DESIRED TRACK
45970       LDA     #$04
45980       STA     ERTMAX
45990       RE$TRK=*
46000       LDA     #$06
46010       STA     ERTMAX+2
46020       REDT$K=*
46030       JSR     REDTKS      ;READ A TRACK
46040       BCC     REREAD      ;IF ERR, GO TO ERR HANDLING
46050       BIT     STATFG      ;REREAD MODE/?
46060       BMI     R$DTRK
46070       LDA     #$80        ;IF NOT, SET FLAG TO INDICATE
46080       ORA     STATFG      ;NOW IS IN REREAD MODE
46090       STA     STATFG
46100       BNE     RE$TRK      ;AND GO BACK TO REREAD!
46110       R$DTRK=*
46120       JMP     UNLDHD      ;IF IS REREAD,UNLOAD HEAD, FINISH!
46130       NOP
46140       REREAD=*            ;ERR HANDLING
46150       LDA     #IOBUF/256
46160       STA     IOBUFA+1
46170       DEC     ERTMAX+2
46180       BNE     REDT$K
46190       BIT     ERORFG
46200       BMI     RE$EAD
46210       BIT     STATFG
46220       BVS     R$READ
46230       DEC     ERTMAX+1
46240       BNE     W$TTRK
46250       R$READ=*
46260       LDA     ERORFG
46270       JMP     ERROR
46280       RE$EAD=*
46290       DEC     ERTMAX
46300       BEQ     R$READ
46310       JSR     STPOUT
```

```
46320       JSR     DLYMS1
46330       JSR     STPIN
46340       JSR     DLYMS1
46350       BEQ     RE$TRK
46360       RESV27*=*+5
46370 ;
46380 ; ************************************************************
46390 ; *                                                          *
46400 ; *  NUMBER  14 :  READ/WRITE  B/S  FROM/TO  DISK            *
46410 ; *                                                          *
46420 ; ************************************************************
46430 ;
46440       RWDBS1=*                ;READ/WRITE B/S FROM/TO DISK
46450 ;
46460       LDA     TKNDER          ;TRACK 0?
46470       BEQ     RWDB$$
46480       CMP     TRKDRC          ;DESIRED TRACK = CURRENT TRACK?
46490       BEQ     R$DBS1          ;YES, NO DISK I/O
46500       BIT     IOBFFG          ;'FLUSH' FLAG SET?
46510       BPL     RWDB$$          ;IF NO, NO WRITING BACK
46520       JSR     WRTTRK          ;WRITE BACK TO CURRENT TRACK
46530       RWDB$$=*
46540       LDA     TRKDRC          ;STORE DESIRED NO. TO CURRENT NO.
46550       STA     TKNDER
46560       JSR     REDTRK          ;READ THIS TRACK
46570       LDA     #$00            ;FLUSH 'FLUSH' FLAG
46580       STA     IOBFFG
46590       R$DBS1=*
46600       BIT     FLAG            ;WRITE OR READ?
46610       BVC     RWD$$$
46620       LDA     #$80            ;IF WRITE, SET 'FLUSH' FLSAG
46630       STA     IOBFFG
46640       RWD$$$=*
46650       NOP
46660       NOP
46670       NOP
46680       NOP
46690       NOP
46700       LDA     DMAREG          ;SENT DMA ADDR. TO HLREG1
46710       STA     HLREG1
46720       LDA     DMAREG+1
46730       STA     HLREG1+1
46740       LDY     #$00
46750       STY     TEMP
46760       LDA     #$10            ;SECTOR OPERATION?
46770       BIT     FLAG
46780       BEQ     RE$BS1
46790       LDA     SECNB1          ;YES, READ OR WRITE A SECTOR
46800       LDX     #$07            ;BETWEEN DISK I/O BUFFER AND DMA
46810       JSR     ASLDS1
46820       JSR     ADCIOF
46830       RED$S1=*
46840       BIT     FLAG
46850       BVS     REDBS$
```

```
46860       LDA     (HLREG),Y
46870       STA     (HLREG1),Y
46880       JMP     R$$BS1
46890       REDBS$=*
46900       LDA     (HLREG1),Y
46910       STA     (HLREG),Y
46920       R$$BS1=*
46930       INY
46940       CPY     #$80
46950       BNE     RED$S1
46960       RTS
46970       RE$BS1=*
46980       LDA     BLKNB1          ;IF IS BLOCK OPERATION, READ
46990       LDX     #$09            ;OR WRITE A BLOCK BETWEEN
47000       JSR     ASLDS1          ;DISK I/O BUFFER AND DMA
47010       JSR     ADCIOF
47020       LDX     #$02
47030       REDB$1=*
47040       BIT     FLAG
47050       BVS     RE$$S1
47060       LDA     (HLREG),Y
47070       STA     (HLREG1),Y
47080       JMP     REB$$1
47090       RE$$S1=*
47100       LDA     (HLREG1),Y
47110       STA     (HLREG),Y
47120       REB$$1=*
47130       INY
47140       BNE     REDB$1
47150       INC     HLREG+1
47160       INC     HLREG1+1
47170       DEX
47180       BNE     REDB$1
47190       RTS
47200       RESV28*=*+5
47210 ;
47220       SIOBDK=*                ;SENT I/O BUFFER BACK TO DISK
47230 ;
47240       BIT     IOBFFG          ;'FLUSH' FLAG SET?
47250       BPL     S$OBDK
47260       JSR     WRTTRK          ;IF SET, WRITE BACK TO DISK
47270       LDA     #$00
47280       STA     IOBFFG
47290       S$OBDK=*
47300       RTS
47310 ;
47320       BCDASC=*                ;CONVERT BCD TO TWO ASCII CODES
47330 ;
47340       PHA
47350       AND     #$0F            ;GET LOW NIBBLE
47360       CLC
47370       ADC     #$30            ;CONVERT TO ASCII CODE
47380       TAY
47390       PLA
```

```
47400       LSR     A
47410       LSR     A
47420       LSR     A
47430       LSR     A
47440       AND     #$0F          ;GET HIGH NIBBLE
47450       CLC
47460       ADC     #$30          ;CONVERT TO ASCII CODE
47470       RTS
47480 ;
47490 ; ************************************************************
47500 ; *                                                          *
47510 ; *  NUMBER  15 :  ERROR HANDLING                            *
47520 ; *                                                          *
47530 ; ************************************************************
47540 ; *                                                          *
47550 ; *  ENTRY PARAMETERS :                                      *
47560 ; *      REGISTER A :  ERROR  NUMBER                         *
47570 ; *                                                          *
47580 ; ************************************************************
47590 ;
47600       ERROR=*
47610 ;
47620       JSR     BCDASC        ;CONVERT ERR NO. TO 2 ASCII
47630       LDX     #$0B          ;AND INSERT THEM TO THE ERR
47640       STA     EROR,X        ;MESSAGE
47650       INX
47660       TYA
47670       STA     EROR,X
47680       LDA     CRTDSK        ;CONVERT CURRENT DISK NO.TO
47690       CLC                   ;ASCII AND INSERT IT TO THE
47700       ADC     #$41          ;ERR MESSAGE
47710       LDX     #$16
47720       STA     EROR,X
47730       LDA     #EROR         ;OUTPUT ERR MESSAGE
47740       STA     DEREG
47750       LDA     #EROR/256
47760       STA     DEREG+1
47770       JSR     OUTBFF
47780       JMP     CCPST1
47790       EROR    .BYTE '   OPST ERR   ON DISK  $'
47800 ;
47810       DFBUFA*=*+512         ;DEFAULT BUF
47820       DIRBUF*=*+512         ;DIRECTORY BUF
47830       MAPAR0*=*+64          ;MAP AREA OF DISK A
47840       MAPAR1*=*+64          ;MAP AREA OF DISK B
47850       CHKAR0*=*+14          ;CHECK AREA OF DISK A
47860       CHKAR1*=*+14          ;CHECK AREA OF DISK B
47870       FLCTBK*=*+35          ;FILE CONTROL BLOCK
```

Appendix B5 : Listing of DSKUTY

```
 5 ; DISKRTTE UTILITY
 10       *=$0200
 30       JMP     DIKULT
 40       BDOS=$05
 50       HLREG=$70
 60       HLREG1=$72
 70       FLAG=$74
 80       TEMP=$75
 90       TKNDER=$76        ;DESIRED TRACK
100       TRKCRN=$77        ;HEAD POSITION
110       TRKCR1=$78        ;HEAD RESERVATION
120       TKTMAX=$79        ;MAX TIMES FOR MATCH TRAC#
130       ERORFG=$7A        ;ERROR FLAG
140       IOBUFA=$7B        ;FOR I/O BUF ADDRESS
150       PAGNUB=$7D        ;PAGE COUNTER
160       PAGENM=$7E        ;DESIRED PAGE NO.
170       ERTMAX=$7F        ;MAX R/W TIMES
180       NOCMP1=$80        ;CHECK PRAMETER
190       NOCMP2=$82
200       SATTRK=$83        ;STARTING TRACK NO.
210       ENDTRK=$84        ;END TRACK NO.
220       SORCDK=$85        ;SOURCE DISK NO.
230       DESTDK=$86        ;DEST. DISK NO.
240       LCTREG=$87        ;ADDRESS FOR DESIRED MOVEING
250       HEADST=$89        ;LOADER VECTOR
260       STPTST=$8B        ;STACK POINTER RESERVATION
270       STATFG=$8C        ;STATE FLAGE
280       FRYCST=$E41B
290       DVACIA=$C010      ;DISK PORT
300       DVPIA1=$C000      ;DISK CONTROL
310       DVPIA2=$C002      ;DISK CONTROL
320       IOBUF=$B000       ;I/O BUF
330       DIKCRT  .BYTE $00  ;CURRENT DISK
340       DKCONT  .BYTE  $40,$00
350       INBUF .BYTE $05  ;INPUT BUFFER
360       INBUF1*=*+5
370       DPLBF1 .BYTE '-- DISKETTE UTILITY --',$0D,$0A
380              .BYTE $0D,$0A,'SELECT ONE',$0D
390              .BYTE $0A,$0D,$0A,'1) COPY ALL TRACKS'
400             .BYTE $0D,$0A,$0D,$0A,'2) COPY OPTION '
410              .BYTE 'TRACKS',$0D,$0A,$0D,$0A,'3) IN'
420              .BYTE 'ITIALIZE OPTION TRACKS (TRACK'
430              .BYTE ' 0 IS NOT ALLOWED)'
440              .BYTE $0D,$0A,$0D,$0A,'4) READ '
450              .BYTE 'OPTION TRACK',$0D,$0A,$0D,$0A
460              .BYTE '5) WRITE OPTION TRACK',$0D,$0A
470              .BYTE $0D,$0A,'6) QUIT TO OUP/M',$0D
480              .BYTE $0A,$0D,$0A,'-- TRACK NUMBER:'
490              .BYTE ' A 2-DIGIT DECIMAL VALUE',$0D
500              .BYTE $0A,$0D,$0A,'-- ADDRESS & '
510              .BYTE 'VECTOR: A 4-DIGIT HEXIDECIMAL '
520              .BYTE 'VALUE',$0D,$0A,$0D,$0A,'-- '
```

```
530             .BYTE 'PAGE NUMBER: A SINGLE HEXI'
540             .BYTE 'DECIMAL VALUE',$0D,$0A,$0D,$0A
550             .BYTE '?$'
560     DPLBF2  .BYTE 'FROM WHICH DISK (A/B) ?$'
570     DPLBF3  .BYTE 'TO WHICH DISK (A/B) ?$'
580     DPLBF4  .BYTE 'FROM WHICH TRACK ?$'
590     DPLBF5  .BYTE 'TO WHICH TRACK ?$'
600     DPLBF6  .BYTE '-- DISKETTE COPIER --',$0D,$0A
610             .BYTE $0D,$0A,$0D,$0A,'THIS ROUTINE '
620             .BYTE 'COPIES ALL TRACKS FROM A TO B'
630             .BYTE $0D,$0A,$0D,$0A,'BE SURE PUT'
640             .BYTE 'MASTER DISK INTO A & YOUR DISK'
650             .BYTE $0D,$0A,$0D,$0A,'INTO B$'
660     DPLBF7  .BYTE '-- DISKETTE OPTION COPIER --'
670             .BYTE $0D,$0A,$0D,$0A,$0D,$0A,'THIS '
680             .BYTE 'ROUTINE OPTIONLY COPIES DESIRE'
690             .BYTE ' TRACKS',$0D,$0A,$0D,$0A,'FROM'
700             .BYTE ' SOURCE DISK TO DESTINATION '
710             .BYTE 'DISK$'
720     DPLBF8  .BYTE '-- DISKETTE INITIALIZER --'
730             .BYTE $0D,$0A,$0D,$0A,$0D,$0A,'THIS '
740             .BYTE 'ROUTINE OPTIONLY INITIALIZES '
750             .BYTE 'TRACK HEADER ',$0D,$0A,$0D,$0A
760             .BYTE 'OF DESIRED DISK EXPECT TRACK '
770             .BYTE '0 $'
780     DPLBF9  .BYTE '-- DISKETTE READER --',$0D,$0A
790             .BYTE $0D,$0A,$0A,$0A,'THIS ROUTINE '
800             .BYTE 'READS ANY TRACK FROM DESIRED '
810             .BYTE $0D,$0A,$0D,$0A,'DISK TO THE '
820             .BYTE 'LOCATION YOU WANT$'
830     DPLBFA  .BYTE '-- DISKETTE WRITER --',$0D,$0A
840             .BYTE $0D,$0A,$0D,$0A,'THIS ROUTINE '
850             .BYTE 'WRITES ANY TRACK FROM THE LO-'
860             .BYTE $0D,$0A,$0D,$0A,'CATION TO THE '
870             .BYTE 'DESIRED DISK$'
880     DPLBFB  .BYTE 'TO WHICH LOCATION ?$'
890     DPLBFC  .BYTE 'FROM WHICH LOCATION ?$'
900     DPLBFD  .BYTE 'GIVE THE LOADER VECTOR !$'
910     DPLBFE  .BYTE 'HOW MANY PAGES ?$'
920     DPLBFF  .BYTE 'ALL DONE ! ANOTHER (Y/N) ?$'
930     DPLBFG  .BYTE 'ARE YOU SURE (Y/N) ?$'
940     DPLBFH  .BYTE 'ERROR !! INDEX HOLE FOUND'
950             .BYTE ' WHILE WRITING$'
960     DPLBFI  .BYTE 'ERROR !! INDEX HOLE FOUND '
970             .BYTE 'WHILE READING$'
980     DPLBFJ  .BYTE 'ERROR !! DRIVE NOT REDAY$'
990     DPLBFK  .BYTE 'ERROR !! TRACK NOT MATCH$'
1000    DPLBFL  .BYTE 'ERROR !! DISK WRITING '
1010            .BYTE 'PROTECTION$'
1020    DPLBFM  .BYTE 'ERROR !! BUFFER & DISK NOT'
1030            .BYTE ' MATCH$'
1040    DPLBFN  .BYTE 'ERROR !! PARITY ERROR$'
1050    DPLBFO  .BYTE 'ERROR !! NO SUCH COMMAND$'
1060    DPLBFP  .BYTE 'TRY AGAIN (Y/N) ?$'
```

```
1070        DPLTAB .WORD DPLBF1,DPLBF2,DPLBF3,DPLBF4
1080               .WORD DPLBF5,DPLBF6,DPLBF7,DPLBF8
1090               .WORD DPLBF9,DPLBFA,DPLBFB,DPLBFC
1100               .WORD DPLBFD,DPLBFE,DPLBFF,DPLBFG
1110               .WORD DPLBFH,DPLBFI,DPLBFJ,DPLBFK
1120               .WORD DPLBFL,DPLBFM,DPLBFN,DPLBFO
1130               .WORD DPLBFP
1140 ;
1150        DELAYB=*        ;;DELAY ROUTINE
1160 ;
1170        INC     DELYMR
1180        INC     DELYMR
1190        INC     DELYMR
1200        INC     DELYMR
1210        RTS
1220        DELYMR*=*+1
1230 ;
1240        DELYMS=*        ;DELAY 1 MS
1250 ;
1260        JSR     DELAYB
1270        DEY
1280        BNE     DELYMS
1290        RTS
1300 ;
1310        DLYMS1=*        ;DELAY (X) MS
1320 ;
1330        LDX     #$0C
1340        D$YMS1=*
1350        LDY     FRYCST
1360        JSR     DELYMS
1370        DEX
1380        BNE     D$YMS1
1390        DL$MS1=*
1400        RTS
1410 ;
1420        DELYUS=*        ;DELAY (X)*100 US
1430 ;
1440        LDA     FRYCST
1450        D$LYUS=*
1460        BIT     $00
1470        SEC
1480        SBC     #$05
1490        BCS     D$LYUS
1500        DEX
1510        BNE     DELYUS
1520        RTS
1530 ;
1540 ;
1550        HOMEHD=*        ;HOME HEAD TO ZERO
1560 ;
1570        JSR     STPIN
1580        JSR     DLYMS1
1590        STY     TRKCRN
1600        STY     TKNDER
```

```
1610        H$MEHD=*
1620        LDA     #$02
1630        BIT     DVPIA1
1640        BEQ     DLYMS1
1650        JSR     STPOUT
1660        BEQ     H$MEHD
1670 ;
1680        ADCOF1=*
1690 ;
1700        CLC
1710        ADC     HLREG
1720        STA     HLREG
1730        LDA     TEMP
1740        ADC     HLREG+1
1750        STA     HLREG+1
1760        RTS
1770 ;
1780        DIKSEL=*
1790 ;
1800        LDX     DIKCRT
1810        LDA     DKCONT,X
1820        STA     DVPIA1
1830        LDA     #$FF
1840        STA     DVPIA2
1850        D$KSEL=*
1860        LDA     #$10
1870        CPX     #$01
1880        BEQ     DI$SEL
1890        LDA     #$01
1900        DI$SEL=*
1910        BIT     DVPIA1
1920        RTS
1930 ;
1940 ;
1950        SEKIHL=*        ;SEEK INDEX HOLE
1960 ;
1970        LDA     DVPIA1
1980        BMI     SEKIHL
1990        S$KIHL=*
2000        LDA     DVPIA1
2010        BPL     S$KIHL
2020        RTS
2030 ;
2040        STPOUT=*        ;;STEP OUT ONE TRACK
2050 ;
2060        LDA     DVPIA2
2070        ORA     #$04
2080        BNE     STEP
2090        STPIN=*         ;STEP IN ONE TRACK
2100        LDA     #$FB
2110        AND     DVPIA2
2120        STEP=*
2130        STA     DVPIA2
2140        JSR     DL$MS1
```

```
2150        AND     #$F7
2160        STA     DVPIA2
2170        JSR     DL$MS1
2180        JSR     DL$MS1
2190        ORA     #$08
2200        STA     DVPIA2
2210        JSR     DL$MS1
2220        LDX     #$08
2230        JMP     D$YMS1
2240 ;
2250        SIHSDK=*        ;SEEK HOLE ,SET ACIA
2260 ;
2270        JSR     SEKIHL
2280        LDA     #$03
2290        STA     DVACIA
2300        LDA     #$58
2310        STA     DVACIA
2320        RTS
2330 ;
2340        LDHEAD=*        ;LOAD HEAD
2350 ;
2360        LDA     #$7F
2370        AND     DVPIA2
2380        L$HEAD=*
2390        STA     DVPIA2
2400        LDX     #$28
2410        JMP     D$YMS1
2420 ;
2430        UNLDHD=*        ;UNLOAD HEAD
2440        LDA     #$80
2450        ORA     DVPIA2
2460        BNE     L$HEAD
2470 ;
2480        WRTDIK=*        ;WRITE A BYTE TO DISK
2490 ;
2500        LDA     DVPIA1
2510        BPL     W$TDIK
2520        LDA     DVACIA
2530        LSR     A
2540        LSR     A
2550        BCC     WRTDIK
2560        STX     DVACIA+1
2570        RTS
2580        W$TDIK=*
2590        LDA     #$11
2600        JMP     ERROR
2610        RESV25*=*+5
2620 ;
2630        REDDIK=*        ;READ ABYTE FROM DISK
2640 ;
2650        LDA     DVPIA1
2660        BPL     R$DDIK
2670        LDA     DVACIA
2680        LSR     A
```

```
2690        BCC     REDDIK
2700        LDA     DVACIA+1
2710        RTS
2720        R$DDIK=*
2730        LDA     #$12
2740        JMP     ERROR
2750 ;
2760        TRKPSA=*        ;PUT HEAD TO DESIRED TRACK
2770 ;
2780        LDX     DIKCRT
2790        JSR     D$KSEL
2800        BEQ     T$KPSA
2810        LDA     #$13
2820        JMP     ERROR
2830        T$KPSA=*
2840        LDA     TKNDER
2850        CMP     TRKCRN
2860        BEQ     TR$PSA
2870        BCS     TRK$SA
2880        JSR     STPOUT
2890        LDA     #$99
2900        BCC     TRKP$A
2910        TRK$SA=*
2920        JSR     STPIN
2930        TXA
2940        TRKP$A=*
2950        SED
2960        ADC     TRKCRN
2970        STA     TRKCRN
2980        CLD
2990        JMP     T$KPSA
3000        TR$PSA=*
3010        CMP     #$43
3020        LDA     DVPIA2
3030        AND     #$BF
3040        LDY     #$00
3050        NOP
3060        BCS     TRKPS$
3070        LDA     #$40
3080        ORA     DVPIA2
3090        TRKPS$=*
3100        STA     DVPIA2
3110        RTS
3120 ;
3130        TRKPSH=*
3140 ;
3150        JSR     TRKPSA
3160        LDA     #IOBUF
3170        STA     IOBUFA
3180        LDA     #IOBUF/256
3190        STA     IOBUFA+1
3200        JMP     LDHEAD
3210 ;
3220        TKNMTH=*        ;MATCH TRACK#
```

```
3230 ;
3240        LDA     #$05
3250        STA     TKTMAX
3260        TKNM$H=*
3270        JSR     SIHSDK
3280        T$NMTH=*
3290        JSR     REDDIK
3300        TK$MTH=*
3310        LDX     TKNDER          ;IF TRACK 0,
3320        BEQ     TKNMT1          ;NO CHECKING
3330        CMP     #$43
3340        BNE     T$NMTH
3350        TKNMT1=*
3360        STA     HEADST+1        ;STORE TRACK 0 LDVECTOR
3370        JSR     REDDIK
3380        LDX     TKNDER
3390        BEQ     TKNMT2
3400        CMP     #$57
3410        BNE     TK$MTH
3420        TKNMT2=*
3430        STA     HEADST
3440        JSR     REDDIK
3450        LDX     TKNDER
3460        BEQ     TKNMT3
3470        CMP     TKNDER
3480        BNE     TKN$TH
3490        TKNMT3=*
3500        STA     PAGENM          ;STORE TRACK 0 PGNO.
3510        RTS
3520        TKN$TH=*
3530        DEC     TKTMAX
3540        BEQ     TKNMT$
3550        LDA     TKNDER
3560        PHA
3570        JSR     HOMEHD
3580        PLA
3590        STA     TKNDER
3600        JSR     T$KPSA
3610        JMP     TKNM$H
3620        TKNMT$=*
3630        LDA     #$14
3640        JMP     ERROR
3650        RESV26*=*+5
3660 ;
3670        WRTTKS=*        ;WRITE A TRACK TO DISK
3680 ;
3690        LDA     #$20
3700        BIT     DVPIA1
3710        BNE     W$TTKS
3720        LDA     #$15
3730        JMP     ERROR
3740        W$TTKS=*
3750        LDX     TKNDER
3760        BNE     WRTT$S          ;CHECK TRACK 0
```

```
3770        JMP     WRTTK0          ;IF IS,TO WRITE TRACK 0
3780        WRTT$S=*
3790        BIT     FLAG            ;CHECK INIT
3800        BPL     WRT$KS          ;GO WRITING NON 0 TK
3810        JSR     SIHSDK          ;SEARCH HOLE,SET ACIA
3820        LDA     #$FC            ;ENABLE WRITE & ERASE
3830        AND     DVPIA2
3840        STA     DVPIA2
3850        LDX     #$0A            ;DELAY 1MS
3860        JSR     DELYUS
3870        LDX     #$43          ;WT REGUIAR HEADER
3880        JSR     WRTDIK
3890        LDX     #$57
3900        JSR     WRTDIK
3910        LDX     TKNDER
3920        JSR     WRTDIK
3930        JMP     WRTT$$
3940        WRT$KS=*        ;WRITING NON 0 TRACK
3950        JSR     TKNMTH          ;MATCH TRACK NO.
3960        LDX     #$04            ;DELAY 0.4 MS
3970        JSR     DELYUS
3980        LDA     #$FE            ;ENABLE WRTITING
3990        AND     DVPIA2
4000        STA     DVPIA2
4010        LDX     #$02            ;DELAY 0.2 MS
4020        JSR     DELYUS
4030        LDA     #$FD            ;ENABLE ERASING
4040        AND     DVPIA2
4050        STA     DVPIA2
4060        LDX     #$08            ;DELAY 0.8 MS
4070        JSR     DELYUS
4080        LDX     #$76            ;WT START FLSG
4090        JSR     WRTDIK
4100        LDX     #$0E            ;SET REGULAR PGNO.
4110        STX     PAGNUB
4120        WRT$$$=*
4130        LDY     #$00
4140        WR$TKS=*
4150        LDA     (IOBUFA),Y
4160        TAX
4170        JSR     WRTDIK
4180        INY
4190        BNE     WR$TKS
4200        INC     IOBUFA+1
4210        DEC     PAGNUB
4220        BNE     WR$TKS
4230        LDX     #$47
4240        JSR     WRTDIK
4250        LDX     #$53
4260        JSR     WRTDIK
4270        JMP     WRTTK$
4280        WRTT$$=*
4290        LDA     DVPIA1          ;WAIT,TILL NEXT HOLE
4300        BMI     WRTT$$
```

```
4310        WRTTK$=*
4320        LDA     DVPIA2
4330        ORA     #$01
4340        STA     DVPIA2
4350        LDX     #$05
4360        JSR     DELYUS
4370        LDA     DVPIA2
4380        ORA     #$02
4390        STA     DVPIA2
4400        LDA     #IOBUF/256
4410        STA     IOBUFA+1
4420        RTS
4430        WRTTK0=*        ;WRITE TRACK 0
4440        LDA     DVPIA1
4450        BPL     WRTTK0
4460        W$TTK0=*
4470        LDA     DVPIA1
4480        BMI     W$TTK0
4490        LDA     #$FC        ;ENABLING WRITE & ERASE
4500        AND     DVPIA2
4510        STA     DVPIA2
4520        WR$TK0=*
4530        LDA     DVPIA1
4540        BPL     WR$TK0
4550        LDX     #$0A        ;DELAY 1 MS
4560        JSR     DELYUS
4570        LDX     HEADST+1    ;WT TRACK 0 VECTOR
4580        JSR     WRTDIK
4590        LDX     HEADST
4600        JSR     WRTDIK
4610        LDX     PAGENM      ;SET DESIRED PGNO.
4620        STX     PAGNUB
4630        JSR     WRTDIK      ;WRITE IT
4640        JMP     WRT$$$      ;GO WRITING REST
4650 ;
4660        REDTKS=*        ;READ A TRACK
4670 ;
4680 ;
4690        JSR     TKNMTH
4700        LDX     #$0E
4710        LDA     TKNDER
4720        BNE     R$DTKS      ;IF NOT 0 SET PGNO.
4730        LDX     PAGENM
4740        JMP     REDT$S      ;NOT CHECK START FLAG
4750        R$DTKS=*
4760        JSR     REDDIK
4770        CMP     #$76
4780        BNE     R$DTKS
4790        REDT$S=*
4800        LDY     #$00
4810        STY     ERORFG
4820        LDA     #$01
4830        RE$TKS=*
4840        BIT     DVACIA
```

```
4850        BEQ     RE$TKS
4860        LDA     DVACIA+1
4870        BVC     RED$KS
4880        LDA     #$80
4890        STA     ERORFG
4900        R$$TKS=*
4910        CLC
4920        RTS
4930        RED$KS=*
4940        BIT     STATFG
4950        BPL     REDTK$
4960        CMP     (IOBUFA),Y
4970        BEQ     REDTK$
4980        LDA     #$40
4990        STA     ERORFG
5000        BNE     R$$TKS
5010        REDTK$=*
5020        STA     (IOBUFA),Y
5030        LDA     #$01
5040        INY
5050        BNE     RE$TKS
5060        INC     IOBUFA+1
5070        DEX
5080        BNE     RE$TKS
5090        LDA     #IOBUF/256
5100        STA     IOBUFA+1
5110        SEC
5120        RTS
5130 ;
5140        WRTTRK=*        ;WRITE A TRACK UTILL NO ERROR
5150 ;
5160        LDA     #$A0
5170        STA     STATFG
5180        JSR     TRKPSH
5190        LDA     #$04
5200        STA     ERTMAX+1
5210        W$TTRK=*
5220        JSR     WRTTKS
5222        BIT     FLAG
5224        BPL     W$$TRK
5226        RTS
5228        W$$TRK=*
5230        JMP     RE$TRK
5240 ;
5250        REDTRK=*        ;READ A TRACK UNTIL NO ERROR
5260 ;
5270        LDA     #$40
5280        STA     STATFG
5290        JSR     TRKPSH
5300        LDA     #$04
5310        STA     ERTMAX
5320        RE$TRK=*
5330        LDA     #$06
5340        STA     ERTMAX+2
```

```
5350        REDT$K=*
5360        JSR     REDTKS
5370        BCC     REREAD
5380        BIT     STATFG
5390        BMI     R$DTRK
5400        LDA     #$80
5410        ORA     STATFG
5420        STA     STATFG
5430        BNE     RE$TRK
5440        R$DTRK=*
5450        JMP     UNLDHD
5470        REREAD=*
5480        LDA     #IOBUF/256
5490        STA     IOBUFA+1
5500        DEC     ERTMAX+2
5510        BNE     REDT$K
5520        BIT     ERORFG
5530        BMI     RE$EAD
5540        BIT     STATFG
5550        BVS     R$READ
5560        DEC     ERTMAX+1
5570        BNE     W$TTRK
5580        R$READ=*
5590        LDA     ERORFG
5600        JMP     ERROR
5610        RE$EAD=*
5620        DEC     ERTMAX
5630        BEQ     R$READ
5640        JSR     STPOUT
5650        JSR     DLYMS1
5660        JSR     STPIN
5670        JSR     DLYMS1
5680        BEQ     RE$TRK
5690        RESV27*=*+5
5700 ;
5710        LFTSHT=*
5720 ;
5730        ASL     A
5740        ASL     A
5750        ASL     A
5760        ASL     A
5770        RTS
5780 ;
5790        RITSHT=*
5800 ;
5810        LSR     A
5820        LSR     A
5830        LSR     A
5840        LSR     A
5850        RTS
5860 ;
5870        LUCVSG=*        ;COVERT A CHAR LOWER TO UPPER
5880 ;
5890        CMP     #$61
```

```
5900        BCC     L$CVSG
5910        CMP     #$7C
5920        BCS     L$CVSG
5930        AND     #$5F
5940        L$CVSG=*
5950        RTS
5960 ;
5970        LUCVBF=*        ;COVERT WHOLE CHARS TO UPPER
5980 ;
5990        LDY     #$01
6000        LDA     INBUF,Y
6010        TAX
6020        L$CVBF=*
6030        INY
6040        LDA     INBUF,Y
6050        JSR     LUCVSG      ;COVERT CHAR TO UPPER
6060        STA     INBUF,Y
6070        DEX
6080        BNE     L$CVBF
6090        RTS
6100 ;
6110        CSLINP=*        ;CONSOLE INPUT
6120 ;
6130        LDA     #INBUF
6140        LDY     #INBUF/256
6150        LDX     #$06
6160        JSR     BDOS
6170        JMP     LUCVBF      ;COVERT LO TO UP
6180 ;
6190        CSLIN1=*        ;INPUT CONSOLE,GET A NO.
6200 ;
6210        JSR     CSLINP
6220        LDY     #$02
6230        LDA     INBUF,Y
6240        RTS
6250 ;
6260        DIGCHK=*        ;CKECK DIGIT WITHIN THE RANGE
6270 ;
6280        STX     NOCMP1
6290        STY     NOCMP2
6300        SEC
6310        SBC     NOCMP2
6320        BCC     D$GCHK
6330        CMP     NOCMP1
6340        BCS     D$GCHK
6350        RTS
6360        D$GCHK=*
6370        JMP     ERROR1
6380 ;
6390        CRLF=*
6400 ;
6410        LDA     #$0D
6420        LDX     #$02
6430        JSR     BDOS
```

```
6440        LDA     #$0A
6450        LDX     #$02
6460        JMP     BDOS
6470 ;
6480        PROMPT=*        ;GIVE DISPLAYING
6490 ;
6500        SEC
6510        SBC     #$01
6520        PHA
6525        JSR     CRLF
6530        JSR     CRLF
6540        PLA
6550        ASL     A       ;(A)*2
6560        LDX     #DPLTAB         ;GET DISPLAY TABLE
6570        STX     HLREG           ;ADDRESS
6580        LDX     #DPLTAB/256
6590        STX     HLREG+1
6600        LDY     #$00        ;ADD OFFSET
6610        STY     TEMP
6620        JSR     ADCOF1
6630        LDA     (HLREG),Y   ;GET DESIRED DISPLAY
6640        PHA                 ;ADDRESS
6650        INY
6660        LDA     (HLREG),Y
6670        TAY
6680        PLA
6690        LDX     #$05        ;DISPLAY THEM
6700        JMP     BDOS
6710 ;
6720        CMNOHD=*        ;HANDLE COMMAND NO.
6730 ;
6740        JSR     PROMPT          ;PROMPT FOR COMMAND NO.
6750        JSR     CSLIN1          ;GET IT
6760        LDX     #$07            ;CHECK ITS RANGE
6770        LDY     #$31
6780        JMP     DIGCHK
6790 ;
6800        DKNOHD=*                ;DISK NO. HANDLE
6810 ;
6820        JSR     PROMPT          ;PROMPT FOR DISK NO.
6830        JSR     CSLIN1          ;GET IT
6840        LDX     #$02
6850        LDY     #$41        ;CHECK IT
6860        JMP     DIGCHK
6870 ;
6880        DIGCH1=*
6890 ;
6900        LDX     #$0A
6910        LDY     #$30
6920        JMP     DIGCHK
6930 ;
6940        TRNOHD=*            ;TRACK NO. HANDLE
6950 ;
6960        JSR     PROMPT          ;PROMPT FOR TRACK NO.
```

```
6970        JSR     CSLINP          ;GET IT
6980        LDY     #$02
6990        LDA     INBUF,Y          ;GET FIRST DIGIT
7000        JSR     DIGCH1          ;CHECK IT
7010        JSR     LFTSHT          ;LEFT SHIFT FOUR BITS
7020        STA     TEMP
7030        LDY     #$03            ;GET NEXT DIGIT
7040        LDA     INBUF,Y
7050        JSR     DIGCH1
7060        ORA     TEMP            ;FORM TRACK NO.
7070        CMP     #$77            ;CKECK IT
7080        BCS     T$NOHD
7090        RTS
7100        T$NOHD=*
7110        JMP     ERROR1
7120 ;
7130        TKNOHD=*        ;START AND END TRACK NOHANDL
7140 ;
7150        LDA     #$04        ;START NO. HANDLE
7160        JSR     TRNOHD
7170        STA     SATTRK
7180        LDA     #$05        ;END NO. HANDLE
7190        JSR     TRNOHD
7200        STA     ENDTRK
7210        CMP     SATTRK
7220        BCC     T$NOHD
7230        RTS
7240        T$NOHD=*
7250        JMP     ERROR1
7260 ;
7270        DIGCH2=*        ;CHECK DIGIT '0-9' & 'A-F'
7280 ;
7290        SEC
7300        SBC     #$30        ;CHECK '0-9'
7310        BCC     D$GCH2
7320        CMP     #$0A
7330        BCC     DI$CH2
7340        SBC     #$11        ;CHECK 'A-F'
7350        BCC     D$GCH2
7360        CMP     #$06
7370        BCS     D$GCH2
7380        ADC     #$0A
7390        DI$CH2=*
7400        RTS
7410        D$GCH2=*
7420        JMP     ERROR1
7430 ;
7440        ADNOCK=*        ;CHECK ADDRESS BYTE
7450 ;
7460        LDA     INBUF,Y         ;GET FIRST DIGIT
7470        JSR     DIGCH2
7480        JSR     LFTSHT          ;LEFT SHIFT FOUR BITS
7490        STA     TEMP
7500        INY
```

```
7510        LDA     INBUF,Y         ;GET NEXT DIGIT
7520        JSR     DIGCH2          ;CHECK IT
7530        ORA     TEMP            ;FORM A BYTE
7540        RTS
7550 ;
7560        ADNOH1=*        ;ADDRESS NO. HANDLE
7570 ;
7580        JSR     PROMPT          ;PROMPT FOR ADDRESSS
7590        JSR     CSLINP          ;GET THEM
7592        LDY     #$01            ;CHECK NUMBERS
7594        LDA     INBUF,Y
7596        CMP     #$04
7598        BEQ     A$NOH1
7600        JMP     ERROR1
7602        A$NOH1=*
7604        INY                     ;HANDLE HIGH BYTE
7610        JSR     ADNOCK          ;CHECK & CONVERT IT
7620        PHA
7630        INY
7640        JSR     ADNOCK          ;HANDLE LOWER BYTE
7650        TAX
7660        PLA
7670        RTS
7680 ;
7690        ADNOHD=*
7700 ;
7710        JSR     ADNOH1
7720        STA     LCTREG+1
7730        STX     LCTREG
7740        RTS
7750 ;
7760        PGNOHD=*        ;PAGE NO. HANDLE
7770 ;
7780        JSR     PROMPT          ;PROMPT FOR PAGE NO.
7790        JSR     CSLINP          ;GET IT
7800        LDY     #$01            ;CKECK IT
7810        LDA     INBUF,Y
7820        CMP     #$02
7830        BCS     P$NOHD
7840        INY
7850        LDA     INBUF,Y
7860        JSR     DIGCH2
7870        CMP     #$0F            ;F PAGE IS NOT ALLOWED
7880        BEQ     P$NOHD
7882        CMP     #$00            ;0 PAGE IS NOT ALLOWED
7884        BEQ     P$NOHD
7890        STA     PAGENM
7900        RTS
7910        P$NOHD=*
7920        JMP     ERROR1
7930 ;
7940        INCTRN=*        ;INCREMENT TRACK NO.
7950 ;                       BY DECIMAL MOOD
7960        SED
```

```
7970        CLC
7980        LDA     #$01
7990        ADC     TKNDER
8000        STA     TKNDER
8010        CLD
8020        RTS
8030 ;
8040        SLSODK=*        ;SELECT SOURCE DISK
8050 ;
8060        LDA     SORCDK
8070        STA     DIKCRT
8080        JMP     DIKSEL
8090 ;
8100        SLDNDK=*        ;SELECT DEST. DISK
8110 ;
8120        LDA     DESTDK
8130        STA     DIKCRT
8140        JMP     DIKSEL
8150 ;
8160        DIGOUT=*        ;DIGIT OUTPUT
8170 ;
8180        CLC
8190        ADC     #$30
8200        LDX     #$02
8210        JMP     BDOS
8220 ;
8230        DPTKNO=*        ;DISPLAY TRACK NO.
8240 ;
8250        JSR     CRLF
8260        LDA     TKNDER      ;HIGHER BYTE OUTPUT
8270        JSR     RITSHT
8280        JSR     DIGOUT
8290        LDA     #$0F        ;LOWER BYTE OUTPUT
8300        AND     TKNDER
8310        JMP     DIGOUT
8320 ;
8330        SURERT=*
8340 ;
8350        LDA     #$10        ;PROMPT 'ARE YOU SURE'
8360        JSR     PROMPT
8370        NOP
8380        NOP
8390        JSR     CSLIN1      ;GET THE ANSWER
8400        CMP     #'Y
8410        BEQ     S$RERT
8420        JMP     D$KULT
8430        S$RERT=*
8440        RTS
8450 ;
8460        NTRDER=*
8470 ;
8480        LDA     #$13
8490        BNE     E$ROR       ;SHOW 'DISK NO READY'
8500        ERROR1=*
```

```
8510 ;
8520        LDA     #$18
8530        BNE     E$ROR           ;SHOW 'COMMAND ERROR'
8540 ;
8550        ERROR=*
8560 ;
8570        STA     TEMP
8580        BIT     TEMP
8590        BMI     ER$OR
8600        BVC     E$ROR
8610        LDA     #$16
8620        BNE     E$ROR
8630        ER$OR=*
8640        LDA     #$17
8650        E$ROR=*
8660        JSR     PROMPT
8665        JSR     UNLDHD
8670        JSR     CRLF
8680        LDA     #$19            ;PROMPT FOR 'TRY AGAIN'
8690        JSR     PROMPT
8700        JMP     EN$
8710 ;
8720        OPCPY1=*                ;MAIN COPY ROUTINE
8730 ;
8740        JSR     SURERT
8750        JSR     SLSODK          ;SELECT SOURCE DISK
8760        BNE     O$CPY1          ;NOT READG,ERROR
8770        JSR     HOMEHD
8780        JSR     SLDNDK          ;SELECT DEST. DISK
8790        BEQ     OPCPY$
8800        O$CPY1=*
8810        JMP     NTRDER
8820        OPCPY$=*
8830        JSR     HOMEHD
8840        LDA     TRKCRN
8850        STA     TRKCR1          ;RESERVE DEST. HEAD
8860        LDA     SATTRK          ;SET START TRACK
8865        STA     TKNDER
8870        OP$PY1=*
8880        JSR     SLSODK          ;SELECT SOURCE DISK
8890        JSR     DPTKNO          ;DISPLAY TRACK NO.
8900        JSR     REDTRK          ;READ TRACK TO I/O BUF
8910        LDA     TRKCR1          ;RECOVE DEST. HEAD
8920        STA     TRKCRN
8930        JSR     SLDNDK          ;SELECT DEST. DISK
8931        LDX     TKNDER
8932        BEQ     OPCP$1          ;IF TRACK 0,NO INIT
8933        LDX     #$80            ;INIT HEADER
8934        STX     FLAG
8935        JSR     WRTTRK
8936        LDX     #$00            ;WRITE A TRACK FROM I/O
8937        STX     FLAG
8938        OPCP$1=*
8940        JSR     WRTTRK
```

```
8950        LDA     TRKCRN          ;RESERVE DEST. HEAD
8960        STA     TRKCR1
8970        LDA     TKNDER          ;ALL DONE ?
8980        CMP     ENDTRK
8990        BEQ     OPC$Y1
9000        JSR     INCTRN          ;;INCREAMENT TRACK NO.
9010        BNE     OP$PY1
9020        OPC$Y1=*
9030        RTS
9040 ;
9050 ;      DISKETTE COPIER
9060 ;      ***************
9070 ;
9080        ALCOPY=*                ;COPY DISK A TO B
9090 ;
9100        LDA     #$06
9110        JSR     PROMPT          ;PROMPT COPIER
9120        LDA     #$00
9130        STA     SORCDK          ;SET SOURCE DISK TO A
9140        STA     SATTRK          ;SET START TRACK TO 0
9150        LDA     #$01            ;SET DEST. DISK TO B
9160        STA     DESTDK
9170        LDA     #$76            ;SET END TRACK TO 76
9180        STA     ENDTRK
9190        JSR     OPCPY1          ;GO COPYING
9200        JMP     END
9210 ;
9220 ;      DISKETTE OPTION COPIER
9230 ;      **********************
9240 ;
9250        OPCOPY=*            ;COPY OPTION TRACK ROUTNE
9260 ;
9270        LDA     #$07          ;PROMPT OPTION COPIER
9280        JSR     PROMPT
9290        LDA     #$02
9300        JSR     DKNOHD          ;PROMPT FOR SOURCE NO.
9310        STA     SORCDK          ;AND HANDLE IT
9320        LDA     #$03            ;PROMPT FOR DEST. NO.
9330        JSR     DKNOHD          ;AND HANDLE IT
9340        STA     DESTDK
9350        JSR     TKNOHD          ;PROMPT FOR START AND
9360                                ;END TRACK,HANDLE THEM
9370        JSR     OPCPY1          ;GO COPYING
9380        JMP      END
9390 ;
9400        INHDRT=*                ;MAIN INIT ROUTINE
9410 ;
9420        LDA     SATTRK          ;SET START TRACK
9430        STA     TKNDER
9440        LDA     #$80            ;SET INIT FLAG
9450        STA     FLAG
9460        I$HDRT=*
9470        JSR     DPTKNO          ;DISPLAY TRACK NO.
9480        JSR     WRTTRK          ;INIT ONE TRACK
```

```
 9490       LDA     TKNDER
 9500       CMP     ENDTRK        ;ALL DONE ?
 9510       BEQ     IN$DRT
 9520       JSR     INCTRN        ;IF NOT,DO NEXT
 9530       BNE     I$HDRT
 9540       IN$DRT=*
 9545       JSR     UNLDHD
 9550       JMP     END
 9560 ;
 9570 ;     DISKETTE INITIALIFER
 9580 ;     ********************
 9590 ;
 9600       INITRT=*
 9610 ;
 9620       LDA     #$08        ;PROMPT INITIALIZER
 9630       JSR     PROMPT
 9640       LDA     #$03        ;PROMPT FOR DISK NO.
 9650       JSR     DKNOHD        ;HANDLE DISK NO.
 9660       STA     SORCDK
 9670       JSR     TKNOHD        ;HANDLE TRACK NO,S
 9680       LDA     SATTRK        ;TRACK 0 IS NOT ALLOWED
 9690       BNE     I$ITHD
 9700       JMP     ERROR1
 9710       I$ITHD=*
 9720       JSR     SURERT
 9730       JSR     SLSODK        ;SELECT DESURED DISK
 9740       BEQ     IN$THD
 9750       JMP     NTRDER        IF NOT READY,ERROR
 9760       IN$THD=*
 9770       JSR     HOMEHD
 9780       JMP     INHDRT        ;GO INIIALIZING
 9790 ;
 9800       RWIODK=*          ;R/W I/O BUF TO/FROM
 9810                         ;DESIRED LOCATION
 9820 ;
 9830       LDY     #$00
 9835       LDX     PAGENM
 9840       R$IODK=*
 9850       LDA     (HLREG),Y
 9860       STA     (HLREG1),Y
 9870       INY
 9880       BNE     R$IODK
 9890       INC     HLREG+1
 9900       INC     HLREG1+1
 9910       DEX
 9920       BNE     R$IODK
 9930       RTS
 9940 ;
 9950       RDOPT1=*          ;MAIN READER ROUTINE
 9960 ;
 9970       JSR     SLSODK        ;SELECT DESIRED DISK
 9980       BEQ     R$OPT1
 9990       JMP     NTRDER        ;IF NOT READY,ERROR
10000       R$OPT1=*
```

```
10010       JSR     HOMEHD
10020       LDX     SATTRK        ;SET DESIRED TRACK
10030       STX     TKNDER
10040       JSR     REDTRK        ;READ DISK TO I/O BUF
10050       LDA     LCTREG        ;SET DESIRED ADDRESS
10060       STA     HLREG1
10070       LDA     LCTREG+1
10080       STA     HLREG1+1
10090       LDA     #IOBUF        ;SET I/O BUF ADDRESSS
10100       STA     HLREG
10110       LDA     #IOBUF/256
10120       STA     HLREG+1
10130       LDX     SATTRK        ;CHECK TRACK 0
10140       BEQ     RD$PT1
10150       LDX     #$0E          ;IF NOT 0,CHANGE PAGE
10160       STX     PAGENM        ;NO. TO REGUIAR VALUE
10170       RD$PT1=*
10180       JSR     RWIODK        ;READ TO I/O BUF
10190       JMP     END
10200 ;
10210 ;     DISKETTE TRACK READER
10220 ;     **********************
10230 ;
10240       RDOPTK=*          ;READ OPTION TRACK INTO
10250                         ;DESIRED LOCATION
10260 ;
10270       LDA     #$09          ;PROMPT READER
10280       JSR     PROMPT
10290       LDA     #$02          ;PROMPT FOR DISK NO.
10300       JSR     DKNOHD
10310       STA     SORCDK
10320       LDA     #$04          ;PROMPT FOR TRACK NO.
10330       JSR     TRNOHD        ;HANDLE IT
10340       STA     SATTRK
10350       LDA     #$0B          ;PROMPT FOR LOCATION
10360       JSR     ADNOHD
10370       JSR     SURERT
10380       JMP     RDOPT1        ;GO READING
10390 ;
10400       WTOPT1=*          ;MAIN WRITER ROUTINE
10410 ;
10420       JSR     SLDNDK        ;SELECT DESIRED DISK
10430       BEQ     W$OPT1
10440       JMP     NTRDER
10450       W$OPT1=*
10460       JSR     HOMEHD
10470       LDA     ENDTRK        ;SET TRACK NO.
10480       STA     TKNDER
10490       BEQ     WT$PT1
10500       LDX     #$0E        ;IF NOT TRACK 0,CHANG PG
10510       STX     PAGENM        ;NO. TO REGULAR VALUE
10520       WT$PT1=*
10530       LDA     LCTREG        ;MOVE FROM DESIR
10540       STA     HLREG         ;LOCATION TO I/O BUF
```

```
10550       LDA     LCTREG+1
10560       STA     HLREG+1
10570       LDA     #IOBUF
10580       STA     HLREG1
10590       LDA     #IOBUF/256
10600       STA     HLREG1+1
10610       JSR     RWIODK
10620       JSR     WRTTRK          ;WRITE I/O TO DISK
10630 ;
10640       END=*
10650 ;
10660       LDA     #$0F            ;PROMPT 'ALL DONE
10670       JSR     PROMPT          ;ANOTHER (Y/N) ? '
10680       NOP
10690       NOP
10700       EN$=*
10710       JSR     CSLIN1          ;GET THE ANSWER
10720       CMP     #'Y
10730       BNE     E$D
10740       JMP     D$KULT          ;IF YES,GO REPERTING
10750       E$D=*
10760       JMP     QUIT
10770 ;
10780 ;     DISKETTE TRACK WRITER
10790 ;     **********************
10800 ;
10810       WTOPTK=*        ;WRITE FROM DESIRED LOCATION
10820                       ;TO DESIRED TRACK
10830 ;
10840       LDA     #$0A            ;PROMPT WRITER
10850       JSR     PROMPT
10860       LDA     #$03            ;PROMPT FOR DISK NO.
10870       JSR     DKNOHD
10880       STA     DESTDK
10890       LDA     #$05        ;PROMPT FOR TRACK NO.
10900       JSR     TRNOHD
10910       STA     ENDTRK
10920       LDA     #$0C            ;PROMPT FOR LOCATION
10930       JSR     ADNOHD
10940       LDA     ENDTRK
10950       BNE     W$OPTY
10960       LDA     #$0D            ;IF TRACK 0,PROMPT FOR
10970       JSR     ADNOH1          ;LOADER VECTOR
10980       STA     HEADST+1
10990       STX     HEADST
11000       LDA     #$0E            ;PROMPT FOR PAGE NO.
11010       JSR     PGNOHD
11020       W$OPTY=*
11030       JSR     SURERT
11040       JMP     WTOPT1          ;GO WRITING
11050 ;
11060 ;     DISKETTE UTILITY
11070 ;     ****************
11080 ;
```

```
11090       DIKULT=*
11100       CLD
11110       TSX
11120       STX     STPTST        ;RESERVE STACK POINTER
11130       D$KULT=*
11140       LDX     #$80          ;SET NEW STACK POINTER
11150       TXS
11160       LDX     #$00          ;RESET INIT FLAG
11170       STX     FLAG
11180       LDA     #$01          ;PROMPT FORCCOMMAND
11190       JSR     CMNOHD        ;AND HANDLE IT
11200       STA     TEMP
11210       ASL     A
11220       CLC
11230       ADC     TEMP
11240       LDX     #SWTBLE        ;DECIDE OPTION
11250       STX     HLREG          ;COMMAND ADDRESS
11260       LDX     #SWTBLE/256
11270       STX     HLREG+1
11280       ADC     HLREG
11290       STA     HLREG
11300       BCC     DI$ULT
11310       INC     HLREG+1
11320       DI$ULT=*
11330       JMP     (HLREG)        ;BRANCH TO DIFFRENT
11340                              ;COMMAND ROUTINE
11350 ;
11360       SWTBLE=*       ;SWTICH TABLE
11370 ;
11380       JMP     ALCOPY
11390       JMP     OPCOPY
11400       JMP     INITRT
11410       JMP     RDOPTK
11420       JMP     WTOPTK
11430       QUIT=*
11440       LDX     STPTST        ;RECOVE STACK POINTER
11450       TXS
11460       RTS
```

.