SECTION 5

USING AIM 65 DOS VERSION 1.0

The AIM 65 Disk Operating System (DOS) Version 1.0 (A65-090) integrates the RM 65 FDC primitive subroutines with fundamental file management functions for use on the AIM 65 Microcomputer. These ROM-based functions, contained in a 4K-byte R2332 ROM, are accessible from the operator through the Debug Monitor/Text Editor ROMs as well as language ROMs and from an application program. Being ROM-based, DOS operation may proceed immediately upon power turn-on without waiting for separate loading of the DOS into RAM.

Text and program source code may be written to, and read from, floppy disk with the Editor List and Read commands, respectively. Similarly, binary data and program object code may be written to, and loaded from, disk using the Monitor Dump and Load commands, respectively. Files containing source and object code for application programs written in AIM 65 Assembler, BASIC, FORTH, PL/65 and Pascal languages are, therefore, supported.

DOS primary commands are selected by the operator at the Monitor command level. These commands invoke the following functions:

Format a Disk
List the Directory
Backup a Disk
List a File
Delete a File
Recover a File

Files are created automatically upon writing a file to disk. A file name and the disk drive number are operator entered in response to system prompts.

Disk read or write errors, both at the DOS and FDC device level, are reported upon detection. User-alterable variables allow changing of default values to application unique values.

5.1 INITIALIZATION

After installing the AIM 65 DOS ROM on the RM 65 FDC module, the DOS is ready for use. The DOS must first be initialized, however, by executing the startup routine, e.g.:

```
<*>=8000
<G>/.
```

The FDC module primitive subroutine variables are initialized, the DOS variables are set to default values (see Section 5.7), the IRQ vectors are initialized, the F1 and F2 user function keys are linked to DOS, the user-defined I/O vectors are loaded for floppy disk I/O, and other temporary variables are initialized. Control then returns to the I/O ROM and subsequently to the Monitor command level. DOS commands may be entered when the Monitor prompt is displayed. (The DOS may not be operated without the Monitor/Editor ROMs installed.)

AIM 65 DOS 1.0 is designed primarily to support AIM 65 Monitor/Editor and language functions in response to commands entered from the keyboard with response directed to the display/printer. When used in this manner, the Monitor/Editor ROMs must be installed. The DOS may also be used to write and read a data file under program control (see Section 5.5). In this case, the Monitor/Editor ROMs are not required unless a Monitor subroutine is used.

> **NOTE**
>
> The DOS uses two contiguous 256-byte buffers, the output buffer and the input buffer, to transfer data between RAM and a floppy disk (as well as for other intermediate data storage). The high byte of the output (or source) buffer is specified by the variable BUFFER (see Section 5.7). The input (or destination) buffer is located immediately above the output buffer. BUFFER is initialized by the startup routine ($8000) to $06 which locates the buffers in low RAM (i.e., $600-7FF).

> **NOTE (Cont'd)**
>
> Application programs must be located above the FDC module primitive subroutine variables ($040A-$4FF--see Section 4.6), DOS 1.0 variables ($0500-$0564--see Section 5.7) and above the DOS buffers (if left at their default locations, i.e., $600-$7FF).

5.2 DISK FORMAT

A floppy disk must be formatted in the AIM 65 DOS 1.0 format before it can be used (see Section 5.3.3). The DOS disk format (illustrated in Figure 5-1) reserves track 0 for future use, uses track 1 and one-half of track 2 for the directory, and the rest of the tracks for sector header and data storage. The sectors are formatted sequentially on the disk.

Trk No.	Sect No.	Function	Single-Density			Double-Density		
			Byte No.	File No. 5"	File No. 8"	Byte No.	File No. 5"	File No. 8"
0	All	Reserved	1-128	-	-	1-256	-	-
1	1	File Entry	1- 16	1	1	1- 16	1	1
		File Status	1			1		
		00=Directory End 01=Active File 02=Deleted File						
		File Name	2- 11			2- 11		
		Start Track	12			12		
		Start Sector	13			13		
		Length-Low Byte	14			14		
		Length-High Byte	15			15		
		Null (00)	16			16		
		File Entries	17-128	2- 8	2- 8	17-256	2- 16	2- 16
	2	File Entries	1-128	9-16	9- 16	1-256	17- 32	17- 32
	3- 8	File Entries	1-128	17-64	17- 64	1-256	33-128	33-128
	9-13	File Entries	1-128	-	65-104	1-256	-	129-208
	14-16	Free	1-128	-	-	1-256	-	-
2	All	Free	1-128	-	-	1-256	-	-

Figure 5-1. AIM 65 DOS 1.0 Disk Format

5.3 PRIMARY OPERATOR COMMANDS

The primary operator commands are selected by using one of two keys: F1 and F2. The F1 key directly commands the List Directory Function, (see Section 5.3.1) while the F2 key displays a UTILITY = prompt. One of five utility functions may then be commanded (see Sections 5.3.2 through 5.3.3). If an invalid key is pressed for a utility function, the utility menu is displayed, e.g.:

```
<]> UTILITY=<RETURN>
(F)ORMAT, (B)ACKUP, (L)IST, (D)EL, (R)EC
```

The <ESC> key is vectored through variable FESCIV (see Section 5.7) to jump to the Monitor command level. The <ESC> key vector is initialized upon cold reset and may be user-altered. Note that this vector is also used to return control to the Monitor after reporting an error condition (see Section 5.6).

NOTE

Upon cold reset, the density selection corresponding to each disk drive is initially set to double. If a disk cannot be read at the selected density, DOS automatically switches to the other density, and attempts to read the disk again. If the read is successful, the selected density is saved for future reading from that disk drive (until a cold reset is performed). This process is essentially transparent to the operator except for a slight delay.

If DOS cannot read the disk at either density, the operator will be prompted to enter the disk density (e.g., DENSITY=) upon the next disk command. The DOS will then attempt to read the disk again at the commanded density (and, if necessary, the other density).

5.3.1 List the Directory

The List Directory function displays the file name of all files recorded on the disk, the sector length of the files and the active/deleted status of each file. The number of free sectors, i.e., the amount of space left on the disk, is also reported.

To list the directory, press the F1 key, then enter the disk drive number in response to the prompt.

Example:

To list the contents of the directory on disk drive No. 1:

```
<[>
DISK=1
FILE NAME     LEN S
LOGGER1.OBJ   269 A
TEXT1         193 A
TEXT2          26 A
SEC LEFT= 178
```

5.3.2 Backup a Disk

The Backup Disk function copies all the active files from one disk to another disk. Since deleted files on the input disk are not copied to the output disk, the newly created disk is essentially compressed (i.e., deleted files removed) to provide additional free space.

To backup a disk, press the F2 key to request the utilities prompt, then press the B key. Enter the disk drive number of the disk to copy from, then the disk drive number of the disk to copy to, in response to prompts.

Example:

To copy a disk on disk drive No. 1 to a disk on disk drive No. 2:

```
<]>
UTILITY=B DISK=1
DISK=2
```

5.3.3 Format a Disk

The Format Disk function initializes a disk to prepare it for subsequent use (see Section 5.2). A new disk, i.e., an unformatted disk, must be formatted before a file can be written upon it. It may also be desired to initialize a previously formatted disk to remove all existing files before re-use.

The Format Disk function also verifies proper operation of the disk media before file storage use. The disk is first initialized by writing sector headers, gaps and CRC data, then a $E5 byte pattern to all sectors. The directory is next initialized to all $00 bytes. If the NOVRFY flag (see Section 5.7) is non-zero (default value), the sector headers are verified, otherwise control returns to the Monitor command level. If the NOVRFY flag is zero, the sector headers are not checked, allowing a faster formatting of the disk.

To format a disk, press the F2 key to request the utilities prompt, then press the F key. Enter the desired disk number and density character when requested. The disk number may vary from 1 to 4 for single-sided disks and from 1 to 8 for double-sided disks (i.e., each side is treated as a separate drive number with odd numbers being the front side and even numbers being the back side). The density character may be either S (for single-density) or D (for double-density).

Since the initialization completely overwrites the disk, a command verification prompt, "ARE YOU SURE?", is displayed before continuing, to prevent inadvertent reformatting. Press Y to continue, or any other key to terminate the command.

The WAIT message is displayed during disk initialization, followed by the Monitor command prompt upon completion. The number of free sectors and bytes is dependent upon disk size and density while the initialization time is dependent upon the state of the NOVRFY flag:

Disk Size	Density Code	Approx. Format Time (Min:Sec)		Free Sectors	Free Bytes
		Verify NOVRFY≠0	No Verify NOVRFY=0		
5"	S	1:45	0:20	535	68,480
5"	D	1:45	0:20	535	136,960
8"	S	5:00	0:30	1962	251,136
8"	D	5:00	0:30	1962	502,272

Example:

Initialize a 5" disk in disk drive No. 1 to single-density then list the directory to check it.

```
<]> UTILITY=F
DISK=1 DENSITY=S
ARE YOU SURE?Y
WAIT

<[>
DISK=1
FILE NAME  LEN S
SEC LEFT= 535
```

NOTES

1. An RM 65 DMA module (RM65-5104) must be used to write and read double-density on 8" disks.
2. If an RM 65 DMA module is used (with either 5" or 8" disk drives), refer to Section A.3 for operating instructions.

5.3.4 List a File

The List File function lists the contents of a file from a disk to another peripheral device. If the peripheral is another disk, a file may be copied from one disk to another.

To list a file, press the F2 key to request the utilities prompt, then press the L key. Enter the file name, disk drive number, and output device code as requested. Respond to output device code subprompts.

Example 1:

To list a text file to the display/printer:

```
<]> UTILITY=L F=TEXT3
DISK=1
OUT=<RETURN>
```

Example 2:

To list an object code file from disk drive No. 1 to disk drive No. 2:

```
<]>
UTILITY=L F=PROG1.OBJ
DISK=1
OUT=U F=PROG1.OBJ
DISK=2
```

The listing may be suspended by pressing the <SPACE> bar, then resumed by pressing the <SPACE> bar again (or one of the other keys).

5.3.5 Delete a File

The Delete File Function changes a file from the active (A) state to the deleted (D) state. In the deleted state, the file cannot be accessed by file name by any function except Recover File (see Section 5.3.5).

To delete a file, press the F2 key to request the utilities prompt, then press the D key. Enter the file name and disk drive number in response to prompts. The Monitor prompt will be displayed upon completion. List the directory to verify the file deletion.

Example:

To delete file TEXT3 (currently active) from disk drive No. 1:

```
<]>
UTILITY=D F=TEXT3
DISK=1
{[}
DISK=1
FILE NAME  LEN S
TEXT1      193 A
TEXT2       94 A
TEXT3       63 D
SEC LEFT= 238
```

A deleted file may be returned to the active state by using the Recover File function.

5.3.6 Recover a File

The Recover File function changes a file from the deleted (D) state to the active (A) state. In the active state, the file can be accessed by file name by other DOS functions.

To recover a file, press F2 key to request the utilities prompt, then press the R key. Enter the file name and disk drive number in response to prompts. The Monitor prompt will be displayed upon completion. List the directory to verify file recovery.

Example:

To recover file TEXT3 (currently deleted) from a disk on disk drive No. 1:

```
<]>
UTILITY=R F=TEXT3
DISK=1
<[>
DISK=1
FILE NAME  LEN S
TEXT1      193 A
TEXT2       44 A
TEXT3       63 A
SEC LEFT= 238
```

5.3.7 Extension of UTILITY Functions

The DOS links to the UTILITY functions through two indirect jump vectors. These two vectors, MENUVC and MENULS, are user-alterable and may be changed to provide alternative linkage and functions. Upon depression of the] key, the DOS jumps through the before menu vector, MENUVC, to check for the five default command characters. If the command character is an F, B, L, D or R, the appropriate processing is performed (see Section 5.3.2 through 5.3.6). If the command character is not one of these letters, the DOS jumps through the after menu vector, MENULS, to display the command menu then redisplay the "UTILITY=" prompt.

Either one or both of these vectors may be altered to meet application dependent requirements.

5.4 FILE HANDLING UNDER OPERATOR CONTROL

A file may be written to, or read from, a disk under operator control by pressing U in response to the OUT=, or IN=, prompt displayed by other major functions, e.g., Monitor, Editor or language. Enter the file name and disk number in response to subprompts.

5.4.1 Using the AIM 65 Monitor

The Monitor Dump (D) function can be used to output machine code from memory to a disk file in ASCII format, while the Monitor Load (L) function can be used to read the file from a disk into RAM (see Sections 4.8.2 and 4.8.1, respectively, in the AIM 65 User's Guide).

a. Writing a File

Example:

Dump machine code, in ASCII, from addresses 0800 through 08FF to file PGM1*A on disk drive No. 1:

```
<D>
FROM=0800 TO=08FF
OUT=U F=PGM1*A
DISK=1
MORE?N
```

b. Reading a File

Example:

Load file PGM1*A from disk drive No. 1 into RAM:

```
<L>IN=U F=PGM1*A
DISK=1
```

5.4.2 Using the AIM 65 Editor

The Editor List (L) function can be used to output ASCII text from the Editor Text Buffer in RAM to a disk file while the Read (R) function can be used to read the file into the Text Buffer (see Sections 5.4.9 and 5.4.1, respectively, in the AIM 65 User's Guide).

Be sure that the Text Buffer is located above the DOS variables and input/output buffers, e.g., $800. If the buffers are relocated, the Text Buffer may start at $600.

a. Writing a File

Example:

Write all the text in the Text Buffer to file PGM1 on disk drive No. 1:

```
=<T>
=<L>
/.
OUT=U F=PGM1
DISK=1
```

b. Reading a File

Example:

Read file PGM1 from disk drive no. 1 into the Text Buffer:

```
=<R>
IN=U F=PGM1
DISK=1

END
```

5.4.3 Using the AIM 65 Assembler

Source code may be read from a disk file into the AIM 65 assembler and/or object code may be written to a disk file from the assembler during the assembly process.

Be sure that the symbol table and, if located in RAM, the source and/or object code are located above the DOS variables and buffers, e.g., $800. If the DOS buffers are relocated, RAM can be used starting at $600.

To assemble a program from source code on the disk, the program shown in Figure 5-2 must first be loaded. Entry to the assembler is now made using the <F3> rather than the usual <N> command. Only use this <F3> entry when assembly is from source code on disk. When using this program, the motors must be turned off after the assembly is complete by depressing the RESET switch. Use of this program is demonstrated in Examples 2 and 3.

> **NOTE**
> When assembling from disk using the driver in Figure 5-2, the first character of the source file is ignored. Begin the first line with a space to avoid an error from occurring.

LINKAGE TO ASSEMBLE SOURCE FROM DISK ON AIM 65

```
ADDR OBJECT   SOURCE

                FOPIN=$8110
                NOPENA=$8155
                ASMVEC=$53F
                ASMFLG=$536
                      *=$112          ;SET UP F3 KEY FOR ENTRY
0112 4C 41 05         JMP START
                      *=ASMVEC+2      ;FREE MEMORY
0541 A9 01    START   LDA #01         ;FLAG FOR SOURCE ON DISK
0543 8D 36 05         STA ASMFLG
0546 A9 53            LDA #<PASSCK    ;CHECK ON EVERY BYTE OF SOURCE
0548 A0 05            LDY #>PASSCK    ;THAT IT IS STILL PASS 1
054A 8D 3F 05         STA ASMVEC
054D 8C 40 05         STY ASMVEC+1
0550 4C 00 D0         JMP $D000       ;ENTER THE ASSEMBLER

0553 A5 23    PASSCK  LDA $23         ;CHECK FOR PASS 2
0555 2D 36 05         AND ASMFLG
0558 D0 03            BNE PASS2       ;STILL PASS 1 -- CONTINUE
055A 4C 55 81         JMP NOPENA

055D A9 00    PASS2   LDA #00         ;ZERO THE ASSEMBLER FLAG
055F 8D 36 05         STA ASMFLG      ;TO SKIP PASSCK
0562 20 10 81         JSR FOPIN       ;REOPEN FILE
0565 4C 55 81         JMP NOPENA      ;GET THE CHAR
                      .END
```

Figure 5-2. Driver to Assemble from Source Code on Disk

Example 1:

Assemble source code from memory (i.e., the Text Buffer) and write the generated object code in ASCII to file PGM1*A on disk drive No. 2:

```
<N>
ASSEMBLER
FROM=1800 TO=1FFF
IN=M
LIST?N
LIST-OUT=<RETURN>

OBJ?Y
OBJ-OUT=U F=PGM1*A
DISK1

PASS 1

PASS 2

ERRORS= 0000
```

Example 2:

Assemble source code from file PGM1 on disk drive No. 1 and do not generate object code.

```
<F3>
ASSEMBLER
FROM=1800 TO=1FFF
IN=U F=PGM1
DISK=1
LIST ?N
LIST-OUT=<RETURN>

OBJ?Y
OBJ-OUT=X

PASS 1

PASS 2

ERRORS= 0000
```

Example 3:

Assemble source code from file PGM1 on disk drive No. 1 and write the generated object code to file PGM1* on disk drive No. 1:

```
<F3>
ASSEMBLER
FROM=1800 TO=1FFF
IN=U F=PGM1
DISK=1

LIST?N
LIST-OUT=<RETURN>

OBJ?Y
OBJ-OUT=U F=PGM1*
DISK=1

PASS 1

PASS 2

ERRORS= 0000
```


5.4.4 Using AIM 65 BASIC

The starting address of RAM used by BASIC to store the application program and variables must be changed from its default value of $211 to a value above the DOS 1.0 variables and input/output buffers, e.g., $0800.

a. BASIC Program Relocation

The following can be used to change the start of the BASIC program:

1. Enter BASIC from the Monitor and enter the memory size and terminal width in response to prompts.

```
<5>
MEMORY SIZE?
WIDTH?
 11758 BYTES FREE
  AIM 65 BASIC V1.1
```

2. Press <ESC> to return to the Monitor.

3. Move the BASIC program and variables start address from $211 to greater than $7FF, e.g., $800. Also, load zeros into three bytes at the start of the BASIC program.

```
<M>=0073 12 02 14 02
</> 0073 01 08 03 08
<M>=0800 AA AA AA AA
</> 0800 00 00 00    <RETURN>
```

4. Initialize DOS 1.0:

```
<*>=8000
<G>/.
```

5. Re-enter BASIC.

```
<6>
```

6. Enter BASIC commands.

The AIM 65 BASIC SAVE and LOAD functions may be used to write and read a BASIC application program, to and from a disk file, respectively (see Section 303 in the AIM 65 BASIC Language Reference Manual).

b. Saving a File

Example:

Save the BASIC program in RAM to file PGM3 on disk drive No. 1:

```
SAVE
OUT=U F=PGM3
DISK=1
```

c. Loading a File

Example:

Load a BASIC program into RAM file PGM3 on disk drive No. 1:

```
LOAD
IN=U F=PGM3
DISK=1
```

5.4.5 Using AIM 65 FORTH

The source code for an application program written in AIM 65 FORTH may be formatted to edit in the AM 65 Text Editor or to input in the FORTH screen format. Refer to Sections 5.4.9 and 5.4.1 in the AIM 65 User's Guide to write source code from the Text Editor to disk or to read source code from disk into the Text Editor. Refer to Section 12 in the AIM 65 FORTH User's Manual for FORTH screen format usage.

The start-up address of RAM used by AIM 65 FORTH to store the application words must be changed from its default value of $300 to a value above the DOS variables and input/output buffers, e.g., $0800.

a. FORTH Application Dictionary Relocation

The following procedure can be used to change the start of the application words:

1. Enter FORTH from the Monitor with the 5 key.

```
<5>
 AIM 65 FORTH V1.2
```

2. Move the FORTH starting address from $300 to greater than $7FF, e.g., $800.

```
FORGET TASK
1280 ALLOT
:TASK ;
```

> **NOTE**
>
> Any time FORTH is entered with the 5 key, the above steps must be repeated.

3. Press <ESC> to return to the Monitor.

4. Initialize DOS:

```
<*>=8000
<G>/.
```

5. Re-enter FORTH with the 6 key.

```
<6>
```

6. Enter FORTH commands.

5.4.6 Using AIM 65 PL/65

The source code for an application program written in AIM 65 PL/65 may be read from disk into the PL/65 compiler and/or the generated object code (i.e., in 6502 assembly language) may be written to disk (see Section 2.6 in the AIM 65 PL/65 User's Manual).

a. Writing a File

Compile source code from memory (i.e., the Text Buffer) and write the output assembler code to file PGM2 or disk drive No. 2:

```
<5>
AIM 65 PL/65 V1.0
IN=M OUT=U F=PGM2
DISK=1
PASS(1 OR 2)?2

ERRORS=00
```

b. Reading a File

Compile source code from file PGM2L on disk drive No. 1 and output the generated assembly language to memory:

```
<5>
AIM 65 PL/65 V1.0
IN=U F=PGM2L
OUT=M
PASS(1 OR 2)?2

ERRORS=00
```

5.4.7 Using AIM 65 Instant Pascal

An application program written in Pascal can be compiled (i.e., to an internal language code for interpretation) into RAM, from a disk file, by AIM 65 Instant Pascal. The source code for an application program in RAM may also be translated from internal format to structured Pascal statements and written to a disk file. Refer to Sections 3.1 and 3.2.3, respectively, in the AIM 65 Instant Pascal User's Manual for user instructions.

The start-up address of RAM used by AIM 65 Instant Pascal to start the application program internal code must be changed from it's default value of $02FC to a value above the DOS 1.0 variables and input/output buffers, e.g., $0800.

a. Instant Pascal Program Relocation

The following procedure can be used to move the start-up address of the internal code:

1. Enter Instant Pascal from the Monitor and enter the memory size and terminal width in response to prompts.

```
<5>MEMORY WIDTH?
WIDTH?
```

2. Press <ESC> to return to the Monitor.

3. Change the program start-up address to a value greater than $7FF, e.g., $800:

```
<M>=001A FC 02 XX XX
</> 001A FC 07 XX XX
```

4. Initialize AIM 65 DOS 1.0:

```
<*>=8000
<G>/.
```

5. Re-enter Instant Pascal:

```
<5>
```

6. Enter Pascal statements.

> **NOTE**
>
> Any time a Cold Reset (Pascal Z command) is performed, location $1A-$1B will be reset to $02FC.

b. Writing a File

Example:

Write the Pascal statements for a program from AIM 65 Instant Pascal to file PGM4P on disk drive No. 1:

```
+<L>/.
OUT=U F=PGM4P
DISK=1
```

c. Reading a File

Example:

Read the Pascal statements into AIM 65 Instant Pascal from file PGM4P on disk drive No. 1:

```
+<R>IN=U F=PGM4P
DISK=1
```

c. Read a File

Compile source code from file PGM2L on disk drive No. 1 and output the generated assembly language to memory:

```
<5>AIM 65 PL/65 V1.Ø
 IN=F F=PGM2L
 DISK=1
 OUT=M
 PASS(1 OR 2)?2
 ERRORS=ØØ
```

5.5 FILE HANDLING UNDER PROGRAM CONTROL

Often it is desirable to store and/or retrieve data from a floppy disk entirely under program control, e.g., when collecting data in an unattended installation. While this can be easily done by writing a program which calls the primitive routines (see Section 4), use of the primitive routines alone do not support the directory format used by AIM 65 DOS 1.Ø. DOS compatible data files written under program control allow easy examination and handling under operator control using DOS

utility functions. This section describes a method to write data to and read data from a floppy disk in a format that is compatible with the AIM 65 DOS 1.Ø file structure.

5.5.1 Writing and Reading Data Files

The AIM 65 DOS 1.Ø file structure allows the writing and reading of data to and from disk independent of the data meaning (e.g., program source code, program object code, or symbol table) or data format (e.g., ASCII coding or machine code). The assembly listing in Figure 5-3 details a DOS compatible file handler. This handler contains both an output driver (to write a file to disk) and an input driver (to read a file from disk). The DOS subroutines and variables used by these drivers are listed in Tables 5-1 and 5-2, respectively.

To write a file, first set up the file name, disk number and side number, then call the write open subroutine (OPENWR). Next, pass data a byte at a time to the disk file by loading each byte into the Accumulator and calling the Monitor OUTALL subroutine. When all data bytes have been output, output an End of File (EOF) character ($1A) and call the write close subroutine (CLOSEW) to close the file. The CLOSEW subroutine automatically outputs the additional fill bytes (value=$ØØ) to complete the last sector.

To read a file, first set up the file name, disk number and side number, then call the read open subroutine (OPENRD). Next, read data from the disk file, a byte at a time, by calling the Monitor INALL subroutine and storing the read byte in the Accumulator into memory. When an EOF is received, turn off the disk drives, call the read close file (CLOSER) to close the file.

> NOTE
>
> This input driver uses $1A as the EOF indicator. When $1A is detected by the INALL subroutine, the disk drive motors are turned off. The user program must close the file however.

```
ADDR OBJECT   SOURCE

                ;
                ; FDC ADDRESSES ( 5-26-82 )
                DISK2=$885B
                DISK1=$884A
                TRKSK=$84E6
                CLOPEN=$81A7
                FOPIN=$8110
                CLSOUT=$82AD
                CLSIN=$8195
                LISTRT=$8353
                DOSSET=$8006
                WRTIT=$8248
                ; FDC VARIABLES
                FINOUT=$522
                DSTBUF=$D9
                SRCBUF=$D7
                BUFFER=$531
                CLOUTV=$52E
                EOF=$537
                ;
                ; AIM 65 VARIABLES
                OUTFLG=$A413
                INFLG=$A412
                ;
                      *=$F00
                ;
                ; AFTER TURNING ON THE SYSTEM , THE FDC FIRMWARE MUST BE
                ; INITIALIZED ( TYPICALLY AS A RESET PROCEDURE ).
                ; THIS CAN BE UNDER OPERATOR CONTROL BY EXECUTING $8000
                ; OR UNDER PROGRAM CONTROL BY :
                ;   INIT  JSR  DOSSET     ;INITIALIZE FDC DOS
                ;
                ; TO OPEN A WRITE FILE , SET UP DISK,SIDE AND
                ; NAME BEFORE CALLING THIS OPEN ROUTINE:
                ;  DISK = $501 - NUMBER 0-7
                ;  SIDE = $503 - 0 FOR FIRST SIDE
                ;  FILENAME = $518 - 10 CHARACTERS PADDED WITH $20
                ;  DENSITY = $505 - 0 FOR DOUBLE (OPTIONAL)
                ;
0F00 A2 01      OPENW  LDX #1             ;USE WRITE BUFFER
0F02 8E 22 05          STX FINOUT
0F05 A9 53             LDA #<LISTRT       ;DISABLE CHARACTER CHECKS
0F07 A0 83             LDY #>LISTRT
0F09 8D 2E 05          STA CLOUTV
0F0C 8C 2F 05          STY CLOUTV+1
0F0F 20 5B 88          JSR DISK2          ;TURN ON WRITE DRIVE
0F12 A9 00             LDA #0             ;SET UP DESTINATION BUFFER
0F14 85 D9             STA DSTBUF
0F16 AC 31 05          LDY BUFFER
0F19 C8                INY                ;WRITES USE BUFFER+1
0F1A 84 DA             STY DSTBUF+1
0F1C 20 E6 84          JSR TRKSK          ;RESTORE WITH DENSITY CHECK
0F1F 20 A7 81          JSR CLOPEN         ;OPEN THE FILE
0F22 A9 55             LDA #'U            ;POINT OUTALL TO FLOPPY
0F24 8D 13 A4          STA OUTFLG
0F27 60                RTS
```

Figure 5-3. Program Control File Handler


```
ADDR OBJECT   SOURCE

                ;
                ; TO WRITE DATA , CALL OUTALL WITH BYTE THE IN A .
                ; THE USER DRIVER WILL WRITE 128/256 BYTES AT A TIME .
                ; PASS AN EOF AS THE LAST CHARACTER .
                ;
                ; TO CLOSE THE WRITE FILE
0F28 AD 37 05   CLOSEW LDA EOF            ;CLOSE OUT LAST SECTOR
0F2B 20 48 82          JSR WRTIT
0F2E A9 20             LDA #$20           ;RESTORE ADD TO AIM 65
0F30 8D 13 A4          STA OUTFLG
0F33 4C AD 82          JMP CLSOUT         ;CLOSE DISK FILE
                ;
                ; TO OPEN A READ FILE , SET UP DISK,SIDE AND
                ; NAME BEFORE CALLING THIS OPEN ROUTINE:
                ;  DISK = $500 - DRIVE NO.  0-7
                ;  SIDE = $502 - 0 FOR FIRST SIDE
                ;  FILENAME = $50E - 10 CHARACTERS PADDED WITH $20
                ;  DENSITY = $504 - 0 FOR DOUBLE (OPTIONAL)
                ;
0F36 A2 00      OPENR  LDX #0             ;USE READ BUFFER
0F38 8E 22 05          STX FINOUT
0F3B 20 4A 88          JSR DISK1          ;TURN ON READ DRIVE
0F3E A9 00             LDA #0             ;SET UP SOURCE BUFFER
0F40 85 D7             STA SRCBUF
0F42 AC 31 05          LDY BUFFER
0F45 84 D8             STY SRCBUF+1
0F47 20 E6 84          JSR TRKSK          ;RESTORE WITH DENSITY CHECK
0F4A 20 10 81          JSR FOPIN          ;OPEN THE READ FILE
0F4D 60                RTS
                ;
                ; TO READ DATA , CALL INALL - BYTE RETURNED IN A .
                ; CHECK FOR AN EOF CHARACTER IF NEEDED .
                ;
                ; TO CLOSE THE READ FILE
0F4E 4C 95 81   CLOSER JMP CLSIN          ;CLOSE DISK READ FILE
                       END

 ERRORS=0000
```

Figure 5-3. Program Control File Handler (Cont'd)

Table 5-1. Commonly Used AIM 65 DOS 1.Ø Subroutines

Label	Address (Hex)	Function
CLOPEN	81A7	Open the Write Disk File.
CLSIN	8195	Close the Read Disk File and Turn Off the Drives.
CLSOUT	82AD	Close the Write Disk File and Turn Off the Drives.
DISK1	884A	Set Up and Turn On the Read Drive.
DISK2	885B	Set Up and Turn On the Write Drive.
DOSSET	8ØØ6	Iintialize FDC Primitive and DOS Variables.
ERRLCL	84D6	Error Handler for the AIM 65 DOS (includes FESC).
FESC	84DA	Turn Off Motors and Jump Through FESCIV.
FOPIN	811Ø	Open the Read Disk File.
LISTRT	8353	Return Instructions.
TRKSK	84E6	Seek to the Track in UCYL with Verify and Density Check.
WRTIT	8248	Write out one Byte from A (called through OUTALL).

Table 5-2. Commonly Used AIM 65 DOS 1.Ø Variables

Label	Address (Hex)	Function
BUFFER	531	MSB of Pointer to DOS Buffer.
CLOUTV	52E	Pointer to the close check for write files.
CURCMD	4AE	Last Command Executed by FDC device.
EOF	537	Character used as an End-of-File marker.
DSTBUF	ØD9	Pointer to DOS write buffer.
ERSTAT	524	DOS error status.
ERRVEC	532	Pointer to the DOS error processing routine
FBYTE	5ØA	Buffer 1 current byte (Read).
	5ØB	Buffer 2 current byte (Write).
FESCIV	534	Pointer to the DOS escape processing routine
FDISK	5ØØ	Buffer 1 disk number (Read).
	5Ø1	Buffer 2 disk number (Write).
FSIDE	5Ø2	Buffer 1 side number (Read).
	5Ø3	Buffer 2 side number (Write).
FDEN	5Ø4	Buffer 1 density (Read).
	5Ø5	Buffer 2 density (Write).
FNAMES	5ØE-517	Buffer 1 file name (Read - 1Ø characters.
	518-521	Buffer 2 file name (Write) - 1Ø characters.
FINOUT	522	Read/Write indicator (Ø=Read).
LEN	53D	Number of remaining sectors (Read).
STFLG	4AD	Image of FDC device status register.
SRCBUF	ØD7	Pointer to DOS Read buffer.

If an error occurs during writing or reading using this handler, the IRQ interrupts are reenabled, the disk drive motors are turned off and control is returned to the AIM 65 Monitor command level. The source of the error can be determined by examining the command type (WRCMD), the FDC Status Register (STFLG), and the DOS error status (ERSTAT).

Note that the address of the error handler is placed in ERRVEC and the address to jump to after the error has been serviced is placed in FESCIV. If errors are to be handled under program control, these vectors must be loaded appropriately. The error handler must isolate and service the error as required. If the disk operation is to be terminated, the IRQ interrupts must be reenabled (JSR ION) the disk drive motors turned off (JMP FESC) and I/O restored to the system terminal.

5.5.2 DOS Compatibility Considerations

The user is responsible for the meaning and coding of all data in files created under program control, including the definition and inclusion of control bytes. Sometimes the files are handled by knowing exactly how many bytes are written to a file, and therefore how many to expect upon reading it (which can be detected by a logical zero result in LEN, LEN+1 and FBYTE). Any $00 pad bytes used to fill the last sector can be ignored.

Another approach is to include one or more special characters to indicate the end of a file. The end of file character(s) must not conflict with valid data values using their approach, however.

In order for a data file to be listed using the DOS list utility function (UTILITY=L, see Section 5.3.4), the data must be coded in ASCII and terminated with an ASCII End of File (EOF) character (byte value = $1A).

An example program using the handler shown in Figure 5-3 is listed in Figure 5-4. This example writes a series of ASCII characters to disk then reads the file. Since an $1A EOF terminator is used, the written file can be examined using the UTILITY=L DOS function. Constants are used for the file name, disk number and side number in this example.

5.6 DOS ERROR REPORTING

The DOS reports an error condition and value indicating the cause of the error whenever improper disk operation is detected. These errors may occur due to reasons such as improper disk drive operation, incorrect disk drive/FDC module connection, bad disk media, and operator error (e.g., wrong disk installed, wrong file name specified, wrong disk drive specified, attempted use of unformatted disk, etc.).

The error reporting format is:

```
DISK ERROR=
```

where:

XX = DOS detected errors plus seek error from the FDC device on the FDC module.

YY = The contents of the FDC device Status Register (less SEEK error) on the FDC module (see Section 4.5).

The error definitions are described in Table 5-3.

The error reporting linkage is through two vectors, ERRVEC (vector before error handling) and FESCIV (vector after error handling)--see Section 5.7. These vectors are initialized upon cold reset and may be altered to link to user-provided additional or alternative error handling.

Note that the default error handler return is to the Monitor (to COMIN at $A32E) after reporting an error code.

```
ADDR OBJECT   SOURCE

                OUTALL=$E9BC
                INALL=$E993
                OUTPUT=$E97A
                OPENW=$F00
                OPENR=$F36
                CLOSEW=$F28
                CLOSER=$F4E
                EOF=$537
                CHAR   =0
                       *=$F80
                ;
                ; THIS TEST PROGRAM WRITES OUT TWO SECTORS OF NUMBERS TO
                ; TO THE DISK .  THE FILE IS THEN READ AND DISPLAYED .  ALL
                ; UNDER PROGRAM CONTROL .  THE FILE ( NAME IS FROM FNAME )
                ; IS SUITABLE TO BE LISTED FROM DOS
                ;
                ; SET UP THE READ AND WRITE DISK PARAMETERS FIRST
                ;
0F80 A9 00      TEST   LDA #0
                ; SET UP THE DRIVES
0F82 8D 00 05          STA $500          ;READ DRIVE
0F85 8D 01 05          STA $501          ;WRITE DRIVE
                ; SET UP THE SIDES
0F88 8D 03 05          STA $503          ;READ SIDE
0F8B 8D 04 05          STA $504          ;WRITE SIDE
                ; DOWNLOAD FILENAME FROM FNAME
0F8E A2 0A      NAME   LDX #10
0F90 BD DD 0F   NAME1  LDA FNAME,X
0F93 9D 0E 05          STA $50E,X        ;READ FILE NAME
0F96 9D 18 05          STA $518,X        ;WRITE FILE NAME
0F99 CA                DEX
0F9A 10 F4             BPL NAME1
                ;
                ; NEXT WRITE OUT A TEST FILE
                ;
0F9C 20 00 0F   TESTWR JSR OPENW         ;OPEN FILE
0F9F A2 07      ISSUE  LDX #7
0FA1 A0 3C      NEWLIN LDY #60
0FA3 A9 30      ZERO   LDA #'0
0FA5 85 00             STA CHAR
0FA7 A5 00      LOOP   LDA CHAR
0FA9 C9 3A             CMP #':
0FAB F0 F6             BEQ ZERO
0FAD 20 BC E9   WRITE  JSR OUTALL
0FB0 E6 00             INC CHAR
0FB2 88                DEY
0FB3 D0 F2             BNE LOOP
0FB5 A9 0D             LDA #$0D          ;CR
0FB7 20 BC E9          JSR OUTALL
0FBA CA                DEX
0FBB D0 E4             BNE NEWLIN
0FBD AD 37 05   END    LDA EOF           ;SHOULD BE $1A
0FC0 20 BC E9          JSR OUTALL
0FC3 20 28 0F   DONE   JSR CLOSEW
```

Figure 5-4. Example Program Using File Handler


```
ADDR OBJECT   SOURCE

                ;
                ; NOW READ IN THE TEST FILE
                ;
0FC6 20 36 0F   TESTRD JSR OPENR         ;OPEN FILE
0FC9 20 93 E9   READ   JSR INALL         ;READ IN A CHARACTER
0FCC CD 37 05          CMP EOF           ;SAME AS MARK WRITTEN
0FCF F0 06             BEQ EOFOUT
0FD1 20 7A E9   OUTWIT JSR OUTPUT
0FD4 4C C9 0F          JMP READ
0FD7 20 4E 0F   EOFOUT JSR CLOSER
0FDA 4C A1 E1          JMP $E1A1         ;COMIN1 ==>
                ;
                ; THE NAME OF THE FILE IS  ...
0FDD 54 45      FNAME  .BYT 'TESTFILE11'
                       .END

 ERRORS=0000
```

Figure 5-4. Example Program Using File Handler (Cont'd)

Table 5-3. AIM 65 DOS 1.0 Error Definitions

Code	Error	Definition
ØlYY	SEEK ERROR	The FDC device did not locate the data on the disk.
Ø2YY	Undefined	
Ø4YY	DISK FULL	The disk is full, thus, no more data may be written on it.
Ø8YY	DIRECTORY FULL	The directory is full, thus, no more file names may be entered.
1ØYY	FILE NAME NOT FOUND	The specified file name is not in the directory.
2ØYY	FILE NAME EXISTS	The specified file name already exists for an active file.
8ØYY	DIRECTORY ERROR	The directory terminator was not found within the directory limits.
XXØ1	BUSY	The FDC device is busy.
XXØ2	Undefined	
XXØ4	LOST DATA	The CPU did not respond to DRQ in one byte time.
XXØ8	CRC ERROR	If RECORD NOT FOUND error exists, an error was found in one or more ID fields, otherwise a data field error exists.
XX1Ø	RECORD NOT FOUND	The desired track, sector, or side were not found.
XX2Ø	RECORD TYPE/WRITE FAULT	On read record, this bit indicates the record-type code from the data field address mark (1=Deleted Data Mark, Ø=Data Mark). On a write, this bit indicates a write fault.
XX4Ø	WRITE PROTECT ERROR	Writing was attempted to a write protected disk.
XX8Ø	NOT READY	The drive is not ready.

5.7 DOS VARIABLES

The DOS variables, located on page zero and five are listed in Table 5-4. Some of these variables are for internal use only (to store temporary values) while others are user-alterable and constant after initialization. The default values for the user-alterable variables, initialized by the sub-routine DOSSET, are also shown in Table 5-4.

Table 5-4. AIM 65 DOS 1.0 Variables

Addr (Hex)	Label	No. Bytes	Init Value	Definition
0D7	SRCBUF	2	-	DOS Source Buffer Vector
0D9	DSTBUF	2	-	DOS Destination Buffer Vector
500	FDISK	1	00	Buffer 1 Disk No.
501		1	00	Buffer 2 Disk No.
502	FSIDE	1	00	Buffer 1 Side No.
503		1	00	Buffer 2 Side No.
504	FDEN	1	00	Buffer 1 Density
505		1	00	Buffer 2 Density
506	FTRACK	1	00	Buffer 1 Track No.
507		1	00	Buffer 2 Track No.
508	FSECTR	1	00	Buffer 1 Sector No.
509		1	00	Buffer 2 Sector No.
50A	FBYTE	1	00	Buffer 1 Byte Count
50B		1	00	Buffer 2 Byte Count
50C	SECLEN	1	00	Buffer 1 Bytes/Sector
50D		1	00	Buffer 2 Bytes/Sector
50E	FNAMES	10	00 -> 00	Buffer 1 File Name
518		10	00 -> 00	Buffer 2 File Name
522	FINOUT	1	00	Buffer to use
523	CHAR	1	00	Character (Temporary) Storage
524	ERSTAT	1	00	Error Status
525	FILNM1	2	00 00	Temporary Storage
527	FILNM2	2	00 00	Temporary Storage
529	FILNM3	2	00 00	Temporary Storage
52B	RETCNT	1	00	Retry Count
52C	READSV	1	00	Temporary Storage
52D	LSTCHR	1	00	Last Character
52E	CLOUTV	2	00 00	Close Output Vector
530	INITFL	1	00	Density Ask Flag
531	BUFFER	1	06	Source Buffer Addr. High Byte*
532	ERRVEC	2	EB 84	Before Error Handler Vector*
534	FESCIV	2	56 E9	After Error Handler Vector*
536	ASMFLG	1	00	Assembly Flag
537	EOF	1	1A	End of File Marker

Table 5-4. AIM 65 DOS 1.0 Variables (Cont'd)

Addr (Hex)	Label	No. Bytes	Init Value	Definition
538	MENUVC	2	64 80	Before UTILITY Decode Vector*
53A	MENULS	2	8A 80	After UTILITY Decode Vector*
53C	NOVRFY	1	01	Format Disk No Verify Flag*
53D	LEN	2	-	Length Count for Read File
53F	ASMVEC	2	-	Assembler Check Vector

NOTE

*User-alterable

1. The DOS 1.0 variables are initialized by calling the DOS routine DOSSET ($8006).