M Z O S

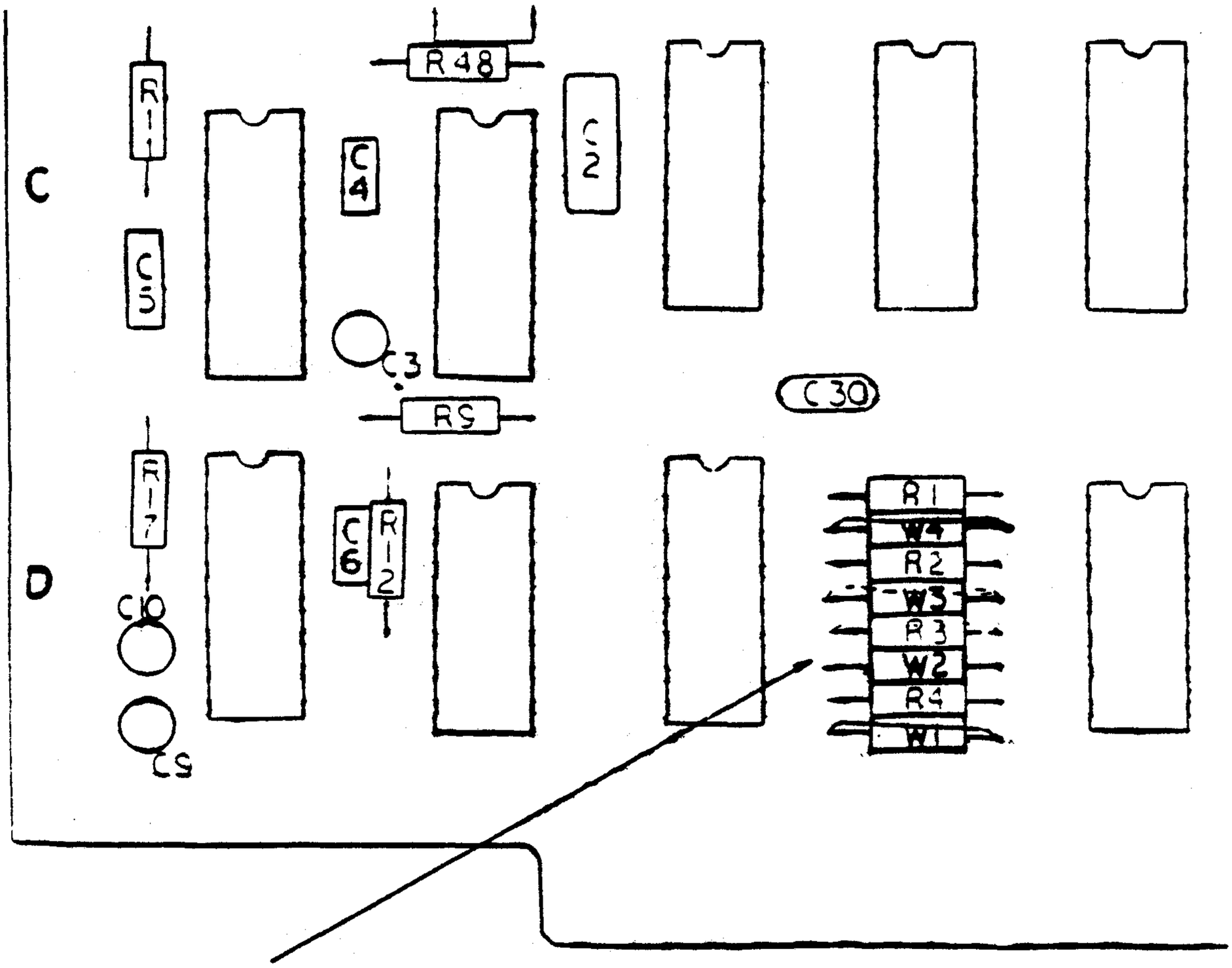
MZ Operating System

by Vector Graphic Inc.

NOTE: Modification on Micropolis Interface Board

for running MZOS

Jumpers in address area should be changed from F4 to D8. (Remove jumper on W3, add jumpers to W1 and W4). The attached diagram shows where jumpers should be added.



Address Jumpers

INTRODUCTION

The Vector MZ operating system, MZOS, was designed specifically for the MZ system. It is a file-oriented disk operating system, allowing you to maintain and use files on the disk. Also provided are subroutines which may be used by the assembly language programmer to interface other software to the disk system, or for programs like BASIC to load and save files on the disk.

This manual reflects version 1.4 of MZOS.

FILE STRUCTURE

First, it is necessary to understand the layout of the disk and the files on it. Thorough understanding of this section will greatly enhance ease of operation of this system.

The disk contains 77 concentric tracks, similar to the grooves in a record. Each track is divided into 16 sectors, and each sector contains one page (256 bytes) of data. Thus the capacity of the entire disk is around 315K bytes. The tracks are numbered 0-76, and the sectors 0-15. For our purposes, however, the disk should be thought of as 1232 sectors, numbered 0-1231, forgetting about tracks altogether.

The first four sectors on the disk are reserved for the directory. The purpose of the directory is to keep track of the files on the disk. There are 64 entries allowed, therefore you may have as many as 64 files on the disk. Each entry in the directory uses 16 bytes. The format of each entry is as follows:

FILENAME	ADDRESS	LENGTH	TYPE	START	SPECIAL
0-7	8-9	10-11	12	13-14	15

The meanings of each field are as follows:

FILENAME
An eight-byte field, this contains the name of the file. It may contain any printing ascii character, except spaces, commas, or lower-case letters.

ADDRESS
This two-byte field contains the address on the disk where the file begins. This address is actually just the sector number of that particular sector. Sector 278 would have a disk address, then, of 278.

LENGTH
The length of the file is the number of sectors it occupies.

TYPE
The type of a file is a special reference to what the file might be used for. A file may be of type 0 through 99. Certain file types have already been defined. They are:

0	Default type unless changed.
1	Executable machine language program.
2	BASIC program.
3	BASIC data file.
4	Ascii text file.
5-7	Reserved.
8	DEX Assembler source file
9	Reserved.

File types 10-99 are not currently used or reserved, so they can be used for special purposes.

START
The start bytes are used to define the loading/starting address for type 1 (machine language) files. In the case of BASIC files, they're used to define the amount of valid data within the file.

SPECIAL
This byte is not used, but is reserved for any special need we might dream up at some future time.

The remaining 1228 sectors on the disk are available for files. There is a restriction that a file may only be 256 sectors long. This is not unreasonable, since it would be rather difficult to load a file longer that 64K into memory anyway. Actually, you may have a file longer then 256 sectors, but you will not be able to load, save, or verify it with MZOS.

A fact to remember is that MZOS does not necessarily access a file when it accesses the directory. Create, delete, and similar operations only modify the directory, not the actual file. Thus if you accidentally delete a file, you can recover it just by creating it with the same addresses as it had before. If you need to create a bigger file, you can just delete the old one and then recreate it with a greater length and the same starting address.

Another thing to remember is that MZOS maintains no copies of the directory in memory, so you may exchange disks at will. (This is as opposed to, say, CPM.)

MZOS MEMORY LAYOUT

It is important to know the places in memory used by MZOS. There are actually three separate areas to know about. The first, symbolically called MZOS, is the actual operating system and buffer areas. This resides in RAM from 2000-29FF. The top of this area is used for system i/o and is explained later. The second area is called MZIO and is a 1K prom at address C400-C7FF. This is the second 2708 on the prom/ram board. It contains all of the disk i/o routines. The third area, MZTMP, is an approximately 32-byte block starting at DF40, in the ram area of the prom/ram board. This is used for the track table and similar important information.

In addition to the regular area, a 2K area immediately following MZOS (actually 2A00-31FF) is used for some mass transfer commands to achieve a higher speed. Execution of the IN, DT, CF, CD, or CO commands will utilize this area, thus overwriting whatever is there.

Last but not least, there are several entry points to the system which you should know about. These, as well as the disk subroutines, are covered in appendix A.

MZOS CONSOLE COMMANDS

Following is an explanation of MZOS commands and their usage. The following conventions are used in this explanation: When something is enclosed in parentheses, e.g. (address), that argument is optional. Something in angle-brackets, e.g. <address>, means that you should substitute a valid argument; in this example, you should type an address. Also, all arguments are in decimal, with the exception of memory addresses, which are in hex.

LI (<unit>)

This will list the directory on the specified unit, with the current disk as the default. Each file will be listed, followed by the address, length, protect status, type, and start address if the file is type 1. A sample listing is shown in appendix C.

FL (<unit>)

This is similar to the LI command, only the directory is listed in a 'fast' format. Only the file is listed, six-wide across the page. This is useful for seeing just if you have a particular file.

CR <file> <length> (<address>)

The create command allows you to create a file entry in the directory. The file will be created with the specified length. If an address is specified, it will be used, otherwise the first free disk address will be defaulted to.

DE <file>

This will delete a file entry. Remember, the file itself will not be harmed, just the entry in the directory. A protected file cannot be deleted.

RN <old-file> <new-file>

This allows you to change the name of (rename) a file. All the other information (type etc.) will remain the same. Do not rename a file to a name that already exists!

TY <file> <type> (<start addr>)

This sets the type of a file. If type 1 is specified, the start address MUST be specified. No accessing of the file is actually done, just the directory entry.

PR <file>

This protects the file specified. When a file is protected, it cannot be deleted. The file may still be written into, however! This is not a perfect protection, but it does keep you from deleting an important file in a moment of frustration. IMPORTANT NOTE! Assembler source (type 8) and BASIC (types 2 and 3) files MAY NOT be protected. If you do, any attempt by DEX or BASIC to look them up will fail.

UP <file>

This will just unprotect a file. No error is caused by unprotecting a file that wasn't protected anyway.

LF <file> <mem addr>
SF <file> <mem addr>
VF <file> <mem addr>

These commands load, save, or verify a file. The load and save commands do so with respect to the memory address specified. Thus 'SF TEST 3000' would save memory into a file called TEST, starting at memory address 3000. Data will be written into the file sequentially until it is full. The LF command works in a similar manner, only data is read from the disk into memory. The VF command does no ram access; it verifies the file by doing an internal check on it. The memory address is necessary, though, so just use 0 as a dummy address,

CF <old-file> <new-file>

This will copy the data from one file to another. In addition, the type and protect information will be copied too. Make sure that the destination file is at least as large as the source file, or you will receive an error.

GO <file>

This will load a type 1 file into memory at its start address and jump to it. Note that this is equivalent to using the LF command followed by a JP command to the start address of the file. Obviously, this only is for type 1 files.

RD <addr> <mem addr> <sectors>
WR <addr> <mem addr> <sectors>
VR <addr> <mem addr> <sectors>

These commands will read, write, or verify data directly on the disk, with no regard for files. These commands are used infrequently, as normally all work is done with files. Using the RD command as an example, take the command RD 4 3000 10. This will read 10 sectors, starting with address 4, and load them into memory beginning at address 3000 (hex). The WR and VR commands work similarly. The VR command, like the VF command, does internal checking only. You may use 0 as a dummy ram address for this command.

IN (<unit>)

This will initialize a fresh diskette. It completely fills the diskette with spaces (ascii 20 hex). This MUST be done to ALL diskettes before use, as they contain garbage before initialization. This command also writes the sector id's on each of the sectors (These id's are used for error checking). You shouldn't initialize a diskette that contains valid data, of course.

DT (<unit>)

This will test a diskette by writing and then verifying a constantly changing pattern over the entirety of the diskette. This destroys any data that was on the diskette, so use with caution, and NEVER on a diskette after it has valid data written on it.

CD <source> <dest>

This will copy the entirety of the source diskette onto the destination diskette. No file-oriented access is used, just direct read and write. The resulting diskette is an exact duplicate of the source diskette.

CO (<unit>)

This compacts the data on the disk by moving all files toward the beginning of the disk. This can become necessary when you have several files, and you delete one or more out of the middle. See also a note in the usage section of this manual referring to this command.

JP <mem adr>

This just transfers control to the specified address in memory. It is used, for example, to start a program that is already in memory.

DD (<unit>)

This sets the default unit to the one specified. If no unit is specified, then unit 1 is used. All disk accesses, when no unit is specified, use the default unit.

; <text>

Typing a semicolon (;) as the first character of a line will cause the line to be ignored. Only thirty characters may be typed, though, or you will receive an error.

<file>

Typing a file name alone is exactly like typing GO followed by the file name. Thus BASIC and GO BASIC are identical commands.

USAGE NOTES ON MZOS

The following is a collection of things to know while using MZOS.

When typing a command, it may be aborted at any time by typing control-C. To erase the last character typed, hit Backspace (if this doesn't suit you, see appendix A).

During execution of a command like DT or IN, you may abort execution by typing control-C.

Typing control-X at ANY time input is expected (as a command, during a DT or IN, etc.) will return you to the prom monitor.

The DT command will run until it finds an error or you stop it.

The IN command takes around a minute and a half.

NEVER NEVER NEVER NEVER interrupt the execution of the CO command, unless you don't mind your diskette being totally destroyed. Compaction normally takes well under five minutes, but depending on the degree of disaster your diskette is in, it could take ten or twenty minutes to compact it. DON'T PANIC! A relatively messy disk could require over a million read/writes, and interrupting it in midstream is guaranteed to leave the disk in a totally unknown state. So unless the disk starts smoking or making really terrible sounds, just leave it alone.

In a multiple-unit system, the particular drive which you wish to reference is indicated by adding a ,1 ,2 ,3 or ,4 to either a filename or disk address. Thus referencing the file TEST on unit 2 would look like TEST,2; a reference to sector 24 on unit 3 would be 24,3. This is true of ANY file reference, in any command; or of the disk address in the RD, WR and VR commands.

MZOS will digest lower case letters as well as upper case, since it just translates everything to upper case anyway.

To repeat something said elsewhere, all disk accesses, where an explicit unit isn't specified, reference the disk specified in the last DD command. If no such command has been issued, then unit 1 is used.

If you have a printer interfaced with your system, you should know that error messages will only print on the console, not the printer, even if the printer is enabled. This prevents getting error messages in the middle of a good printout.

Needless to say, this operating system is completely compatable with the North Star DOS. It is specifically designed so that any software that runs on the North Star system will run on MZOS.

ERRORS AND MISTRAKES

A reasonably comprehensive set of error messages are provided with MZOS. Following is a list of these errors and why you might get them.

Huh?

This means that you typed something that MZOS can't possibly understand. Usually this is from misspelling a command.

% Syntax error

This indicates that you typed a valid command, but invalid arguments. Examples: TY FILE 1 without a start address, or LF FILE QWERT where QWERT is obviously not a hex address.

% File error

This is caused by an invalid file reference. It can be caused by creating a file that already exists, referencing one that doesn't, trying to GO a non-type-1 file, and so forth.

% Disk overflow

You tried to create a file that would extend beyond the boundries of the disk (that is, past sector 1231).

% Write protected

You either tried to write on a protected disk, or tried to delete a file that is protected.

% Disk offline

This can be caused by trying to access the disk when either there is no diskette in the drive, the drive is not up to speed, or the controller just isn't installed in the computer.

% Illegal argument

An attempt was made to read, write, or verify beyond the boundries of the disk. This is similar to the overflow error, only this refers to access attempts, rather that file creation.

% CRC error at sector xxxx,n

Bad data was encountered on the disk in unit n, at sector xxxx. This is usually caused by a faulty diskette, or access to an uninitialized diskette.

% Sector id error at sector xxxx,n

This means that the sector read was not the one wanted. The most common cause of this is a glitch in the stepper motor, or possibly a bad diskette.

Some software that does disk access may want to have control returned to it after an error occurs. Such software should put the address they want control sent to (on an error) into memory at address DF5C. That address is the last two bytes of a jump instruction, which is used after any error. When it is finished, though, it should replace 2028 in that location.

FAMOUS LAST WORDS

MZOS has been thoroughly tested, and should hopefully serve you well on your system. In the event you have any trouble, though, or need help in figuring out how it works, feel free to contact us for assistance. We would also appreciate any comments, suggestions, or improvements to this manual, or additions to or ideas for MZOS, that you might think up.

APPENDIX A: MZOS ENTRY POINTS

Assuming you have some familiarity with assembly language, this should help you understand interfacing to MZOS. Although MZOS is actually a Z-80 program, these explanations will use 8080 code whenever possible as it seems to be more popular.

MZOS 2028

The MZOS entry point is where you jump to start MZOS. The monitor 'J' command will jump here very nicely. At the end of some other program, putting a JMP 2028H will return control to MZOS.

MBOOT C400

This is where to jump to in order to boot up MZOS. The monitor 'E' command can be used to initially do so.

DOOM C402 or 2022

This is the nitty-gritty disk i/o routine. All disk access can be done here. Basically, what you do is set up the registers for what you want to do, then call here (either address will do, since 2022 is just a JMP 0C402H). The registers should be set up as follows:

A	number of sectors to access
B	command - see below
C	unit - 1, 2, 3, or 4
DE	ram address to read into or write from
HL	disk address (sector) to begin access at

Commands are:		
	0	write from memory to disk
	1	read from disk to memory
	2	verify CRC internally on disk
	3	seek (go to) sector specified only
	4	recalibrate drive (go to sector 0)

The last two, seek and recalibrate, are mainly for test purposes. The recalibrate is used to insure proper positioning of the head, in case (for example) the track table is accidentally destroyed, or the stepper motor doesn't.

DLOOK 201C

This is used to lookup a file in the directory. Looking up a blank entry will locate a free space, for creating a new file. When this is called, A should contain the unit number, and HL should point to (contain the address of) a string of characters representing the file name, followed by either a return (0D hex) or a blank (20 hex).

When this routine returns, if the CY flag is set, then the file you looked up does not exist. In this case, HL contains the first free address on the disk. If CY isn't set, then the file was successfully found. In this case, HL points to the eighth byte of a copy of the entry. This copy is actually within the directory buffer of MZOS. See the file structure section of this manual for the structure of the entry.

DWRIT 201F

This will write a directory entry back to the disk. It is important that NO DISK ACCESS OCCUR between DLOOK and DWRIT. Now, the procedure for reading a file would be to use DLOOK to lookup the file; assuming it is really there, incrementing HL will make it point to the starting disk address of the file; then use DCOM to actually read the file. To write a new file to disk, you would first lookup the file, to make sure that it doesn't already exist. This will fail, assuming the name isn't there yet. Next, lookup a blank name. In the event this fails (it shouldn't), the disk you're using doesn't have a directory on it, or is full. Anyway, now HL points to the eighth byte of a blank entry. Now you should copy in the file name, disk address (from your first lookup), length, type, and start address (if needed). This done, call DWRIT to update the directory.

DLIST 2025

This will print the directory of the unit specified in A. The list is exactly the same as the LI command.

RWCHK 202B

This is not an entry point, but rather a flag. If this byte is 1, then a verify will be done after a write. This will slow down write operations considerably, but may be desired if you don't trust your diskette.

BSCHR 25F4

This location contains the character that will be used for a backspace. Normally, this is an 08 (ascii backspace), but you may wish to change it. The most common other characters used are underline (5F) or DEL (7F).

PRMPT 2071

This location contains the character used as a prompt. Currently it is set to a number sign (#, 23 hex), but you may change it if you like.

APPENDIX B: CONSOLE I/O

Here we will discuss the console i/o provided with your system, and how you may change it if necessary.

First, there are four entry points in MZOS which reference console i/o. They are as follows.

INCH 2010

This will input a single character from the console.

OUTCH 200D

This will output a single character.

OUTPR 200A

This will output a character to the printer.

CHKCH 2016

This does two things. First, it checks whether a character has been typed. If not, it returns immediately with the Z flag cleared. If a character has been typed, it sees if it was a control-C. If this is the case, then it returns, with Z set. Finally, if a character was typed, and it was a space, then another character is waited for. This allows you to momentarily suspend output by hitting the space bar. After another character is typed, this routine returns, with the Z flag set if that character was a control-C.

TINIT 2013

This is used to initialize the i/o system, if needed. It is currently set up to initialize the Bitstreamer board, but may be able to be replaced with RET instruction.

So. Those are the entry points. The actual routines, though, are located in a 64-byte block from 2900-29FF. A listing is provided of the routines as provided, in case your system is different.

Checking with the listing, notice that there are addresses assigned to each routine. They are spaced far enough apart that you should be able to fit in any of your own routines without moving any other routine. This means that you shouldn't have to patch the jump table (entry points) at all.

```
2900                    0001 * MZOS STANDARD I/O SYSTEM
2900                    0002 * NEALE BRASSELL [28-JUN-78]
2900                    0003 *
2900                    0004 IOLOC  EQU   2900H
2900                    0005 MZOS   EQU   2028H
2900                    0006 *
2900                    0007 *
2900                    0008 * THIS IS JUST A JUMP TO MZOS, AS A REENTRY POINT
2900                    0009 *
2900                    0010        ORG   IOLOC
2900                    0011 *
2900 C3 28 20           0012 REENT  JMP   MZOS
2903                    0013 *
2903                    0014 *
2903                    0015 * THIS IS THE TINIT (TERMINAL INITIALIZATION) ROUTINE
2903                    0016 *
2903                    0017        ORG   IOLOC+16
2910                    0018 *
2910 AF                 0019 INI8   XRA   A
2911 D3 03              0020        OUT   3
2913 D3 03              0021        OUT   3
2915 D3 03              0022        OUT   3
2917 3E 40              0023        MVI   A,40H
2919 D3 03              0024        OUT   3
291B 3E CE              0025        MVI   A,0CEH
291D D3 03              0026        OUT   3
291F 3E 27              0027        MVI   A,27H
2921 D3 03              0028        OUT   3
2923 C9                 0029        RET
2924                    0030 *
2924                    0031 *
2924                    0032 * THIS CHECKS FOR <^C>, AND SUSPENDS OUTPUT ON <SPACE>
2924                    0033 *
2924                    0034        ORG   IOLOC+64
2940                    0035 *
2940 CD DC C0           0036 CHK8   CALL  0C0DCH
2943 FE 03              0037        CPI   3
2945 C8                 0038        RZ
2946 FE 20              0039        CPI   32
2948 C0                 0040        RNZ
2949 CD 60 29           0041        CALL  IN8
294C FE 03              0042        CPI   3
294E C9                 0043        RET
294F                    0044 *
294F                    0045 *
294F                    0046 * THIS IS THE CONSOLE INPUT ROUTINE
294F                    0047 *
294F                    0048        ORG   IOLOC+96
2960                    0049 *
2960 CD DC C0           0050 IN8    CALL  0C0DCH
2963 CA 60 29           0051        JZ    IN8
2966 E6 7F              0052        ANI   127
2968 C9                 0053        RET
2969                    0054 *
```

```
2969                    0055 *
2969                    0056 * CONSOLE OUTPUT - IF A=1 THEN THE PRINTER IS USED
2969                    0057 *
2969                    0058        ORG    IOLOC+128
2980                    0059 *
2980 FE 01              0060 OUT8   CPI    1
2982 CA C0 29           0061        JZ     OUTPR
2985 78                 0062        MOV    A,B
2986 C3 98 C0           0063        JMP    0C098H
2989                    0064 *
2989                    0065 *
2989                    0066 * PRINTER OUTPUT - JUST USES CONSOLE
2989                    0067 *
2989                    0068        ORG    IOLOC+192
29C0                    0069 *
29C0 78                 0070 OUTPR  MOV    A,B
29C1 C3 98 C0           0071        JMP    0C098H
29C4                    0072 *
```

SYMBOL TABLE

CHK8	2940	IN8	2960	INI8	2910	IOLOC	2900	MZOS	2028
OUT8	2980	OUTPR	29C0	REENT	2900				

APPENDIX C: SAMPLE DIRECTORY LISTING

Here is a sample listing, as produced by the LI command.

```
MZOS         4   10 P 00           BASIC       14   45   01 2A00
DEX         59   23 P 01 2A00      TESTFL      82    4   02
IOSYS       86   90   08           DATAFILE   176  250   03
INXFILE    426   50   03
```

Here is an analysis of the above listing.

MZOS is a file, starting at address (sector) 4 and taking 10 sectors. It is protected, so it cannot be deleted.

BASIC is a 45-sector machine code file, starting at address 14. It's ram starting address is 2A00. It could be executed by typing 'GO BASIC', or just 'BASIC'.

DEX is a machine code file, similar to BASIC. It is protected, though. Its starting ram address is also 2A00.

TESTFL is a BASIC program (type 2), which can be loaded and executed with BASIC.

IOSYS is a DEX source file (type 08).

DATAFILE is a rather large BASIC data file (type 3). It is accessed with READ and WRITE statements in BASIC.

INXFILE is a BASIC data file, like DATAFILE.

MZOS Utilities

by Vector Graphic Inc.

There are three utility programs provided on your system disk as received from Vector Graphic. They are DIAB, CENT, and NS2MZ. Here we will provide you with source listings and instructions for their use.

The first, DIAB, is a driver routine for a Diablo printer. It assumes that you are using ports 2 and 3 for the printer, and that the printer is the version that runs at 1200 baud, with handshaking logic.

The second, CENT, is for a Centronics printer, such as 781, 702, etc. with parallel handshaking logic. Port 1 is used for this printer.

For both routines, they may be invoked by simply typing the name of the one you want, either DIAB or CENT. If you want to save MZOS with one of these routines incorporated into it, you would just type SF MZOS 2000 after loading the driver. Example:

```
#DIAB
#SF MZOS 2000
```

The system on the diskette now has the driver incorporated in it.

Notice (in the source listings) that the various routines are spaced over the entire 2900-29FF block allocated to them. This is so that you can change the drivers without changing the jump table in MZOS. The current assignments are as follows:

REENTRY	2900	jump back to MZOS
TINIT	2910	initialize
CCCHK	2940	control-C check
INCH	2960	input character
OUTCH	2980	output character
OUTPR	29C0	output character to printer

The reentry spot is so that this can be executed as a program, allowing you to type 'DIAB' instead of 'LF DIAB 2900'.

There are a couple of features included in the input and output routines to complement the printer. First, when you type a control-P, the input routines toggles the printer flag, then discards the character. This way, even if your program doesn't allow control characters, typing control-P will still work. What toggling the printer flag actually does is to allow output to go to both the console and the printer. Typing control-P a second time will turn the printer off, etc. Second, typing control-L will send a formfeed to the printer, and discard the character. This should be done before you print something the first time, as it also sets the line counter. The output is paged; every 56 lines, it skips to the top of the next page. Checking the listings provided should help you understand how the system works.

The last utility provided is the NS2MZ program, which, as its name implies, transfers files from North Star disk to MZOS. It is a simple program, as you can see from the listing. To use it, first initialize a disk with MZOS. Then load the NS2MZ program into memory, anywhere EXCEPT from 2000 to 3400. The program is relocatable, so it doesn't matter; we recommend that you load it at 4000. Next, boot your North Star DOS. Insert the disk you want to copy into drive 1 (North Star), and the disk you just initialized into drive 1 (Micropolis). Now JP to whatever address you loaded the program at. It will copy the entire disk, exactly as it is, onto the Micropolis disk. Now insert a system disk into the Miropolis drive and boot MZOS. Type LF MZOS 3000 (load MZOS into memory), then insert the new diskette; type SF DOS 3000 (which saves it onto the disk) followed by RN DOS MZOS (which renames the DOS to MZOS). You now have your North Star diskette on Micropolis disk. This procedure may be repeated for each disk you wish to copy. Note, though, that the disk is copied onto another disk exactly, so you may only copy disks one-to-one. Since the Micropolis disks hold more, you may want to merge several disks together manually after you've copied them.

```
2900                    0001 * MZOS DIABLO I/O SYSTEM
2900                    0002 * NEALE BRASSELL [28-JUN-78]
2900                    0003 *
2900                    0004 IOLOC  EQU   2900H
2900                    0005 MZOS   EQU   2028H
2900                    0006 *
2900                    0007 *
2900                    0008 * THIS IS JUST A JUMP TO MZOS, AS A REENTRY POINT
2900                    0009 *
2900                    0010        ORG   IOLOC
2900                    0011 *
2900 C3 28 20           0012 REENT  JMP   MZOS
2903                    0013 *
2903                    0014 *
2903                    0015 * THIS IS THE TINIT (TERMINAL INITIALIZATION) ROUTINE
2903                    0016 *
2903                    0017        ORG   IOLOC+16
2910                    0018 *
2910 AF                 0019 INI8   XRA   A
2911 D3 03              0020        OUT   3
2913 D3 03              0021        OUT   3
2915 D3 03              0022        OUT   3
2917 3E 40              0023        MVI   A,40H
2919 D3 03              0024        OUT   3
291B 3E CE              0025        MVI   A,0CEH
291D D3 03              0026        OUT   3
291F 3E 27              0027        MVI   A,27H
2921 D3 03              0028        OUT   3
2923 C9                 0029        RET
2924                    0030 *
2924                    0031 *
2924                    0032 * THIS CHECKS FOR <^C>, AND SUSPENDS OUTPUT ON <SPACE>
2924                    0033 *
2924                    0034        ORG   IOLOC+64
2940                    0035 *
2940 CD DC C0           0036 CHK8   CALL  0C0DCH
2943 FE 03              0037        CPI   3
2945 C8                 0038        RZ
2946 FE 20              0039        CPI   32
2948 C0                 0040        RNZ
2949 CD 60 29           0041        CALL  IN8
294C FE 03              0042        CPI   3
294E C9                 0043        RET
294F                    0044 *
294F                    0045 *
294F                    0046 * CONSOLE INPUT - <^P> & <^L> ARE HANDLED SPECIALLY
294F                    0047 *
294F                    0048        ORG   IOLOC+96
2960                    0049 *
2960 CD DC C0           0050 IN8    CALL  0C0DCH        ;GET CHARACTER
2963 CA 60 29           0051        JZ    IN8
2966 FE 10              0052        CPI   16            ;SEE IF ^P
2968 CA 75 29           0053        JZ    CTLP
296B FE 0C              0054        CPI   12            ;SEE IF ^L
```

```
296D C0                 0055            RNZ                     ;RETURN IF NOT
296E 47                 0056            MOV     B,A
296F CD C0 29           0057            CALL    OUTPR           ;SEND FORMFEED
2972 C3 60 29           0058            JMP     IN8             ;DISCARD CHARACTER
2975 3A FB 29           0059 CTLP       LDA     OUTFL
2978 2F                 0060            CMA
2979 32 FB 29           0061            STA     OUTFL           ;^P COMPLIMENTS PRINTER FLAG
297C C3 60 29           0062            JMP     IN8             ;AND DISCARD CHARACTER
297F                    0063 *
297F                    0064 *
297F                    0065 * OUTPUT CHARACTER - IF A=1, THEN USE DIABLO
297F                    0066 *
297F                    0067            ORG     IOLOC+128
2980                    0068 *
2980 FE 01              0069 OUT8       CPI     1
2982 CA C0 29           0070            JZ      OUTPR           ;IF A=1 THEN USE PRINTER
2985 78                 0071            MOV     A,B
2986 CD 98 C0           0072            CALL    0C098H          ;SEND CHARACTER TO CONSOLE
2989 3A FB 29           0073            LDA     OUTFL
298C B7                 0074            ORA     A               ;CHECK PRINTER FLAG
298D 78                 0075            MOV     A,B
298E C8                 0076            RZ                      ;RETURN IF NOT SET
298F C3 C0 29           0077            JMP     OUTPR           ;SEND TO PRINTER IF SET
2992                    0078 *
2992                    0079 *
2992                    0080 * DIABLO PRINTER OUTPUT ROUTINE
2992                    0081 *
2992                    0082            ORG     IOLOC+192
29C0                    0083 *
29C0 DB 03              0084 OUTPR      IN      3
29C2 E6 01              0085            ANI     1
29C4 CA C0 29           0086            JZ      OUTPR           ;WAIT FOR READY FROM USART
29C7 78                 0087            MOV     A,B
29C8 D3 02              0088            OUT     2               ;SEND CHARACTER
29CA FE 0A              0089            CPI     10              ;SEE IF LINEFEED
29CC CA D9 29           0090            JZ      ITALF
29CF FE 0C              0091            CPI     12              ;SEE IF FORMFEED
29D1 C0                 0092            RNZ                     ;RETURN IF NOT
29D2 3E 38              0093            MVI     A,56
29D4 32 FC 29           0094            STA     LNCNT           ;RESET PAGE COUNTER ON FORMFEED
29D7 78                 0095            MOV     A,B
29D8 C9                 0096            RET
29D9 C5                 0097 ITALF      PUSH    B
29DA 06 03              0098            MVI     B,3             ;SEND ETX CHARACTER AFTER LINEFEED
29DC CD C0 29           0099            CALL    OUTPR
29DF DB 03              0100 WTACK      IN      3
29E1 E6 02              0101            ANI     2
29E3 CA DF 29           0102            JZ      WTACK           ;WAIT FOR ACKNOWLEDGE
29E6 DB 02              0103            IN      2               ;GOBBLE CHARACTER
29E8 C1                 0104            POP     B
29E9 3A FC 29           0105            LDA     LNCNT
29EC 3D                 0106            DCR     A               ;ADJUST LINE COUNTER
29ED 32 FC 29           0107            STA     LNCNT
29F0 78                 0108            MOV     A,B
29F1 C0                 0109            RNZ                     ;RETURN IF NOT FULL PAGE
29F2 C5                 0110            PUSH    B
```

```
29F3 06 0C              0111          MVI   B,12          ;WHEN FULL PAGE, SEND FORMFEED
29F5 CD C0 29           0112          CALL  OUTPR
29F8 C1                 0113          POP   B
29F9 78                 0114          MOV   A,B
29FA C9                 0115          RET                 ;AND RETURN
29FB                    0116 *
29FB 00                 0117 OUTFL    DB    0             ;PRINTER FLAG
29FC 38                 0118 LNCNT    DB    56            ;LINE COUNTER
29FD                    0119 *
```

SYMBOL TABLE

CHK8	2940	CTLP	2975	IN8	2960	INI8	2910	IOLOC 2900
ITALF	29D9	LNCNT	29FC	MZOS	2028	OUT8	2980	OUTFL 29FB
OUTPR	29C0	REENT	2900	WTACK	29DF			

```
2900                    0001 * MZOS CENTRONICS I/O SYSTEM
2900                    0002 * NEALE BRASSELL [28-JUN-78]
2900                    0003 *
2900                    0004 IOLOC  EQU   2900H
2900                    0005 MZOS   EQU   2028H
2900                    0006 *
2900                    0007 *
2900                    0008 * THIS IS JUST A JUMP TO MZOS, AS A REENTRY POINT
2900                    0009 *
2900                    0010        ORG   IOLOC
2900                    0011 *
2900 C3 28 20           0012 REENT  JMP   MZOS
2903                    0013 *
2903                    0014 *
2903                    0015 * THIS IS THE TINIT (TERMINAL INITIALIZATION) ROUTINE
2903                    0016 *
2903                    0017        ORG   IOLOC+16
2910                    0018 *
2910 AF                 0019 INI8   XRA   A
2911 D3 03              0020        OUT   3
2913 D3 03              0021        OUT   3
2915 D3 03              0022        OUT   3
2917 3E 40              0023        MVI   A,40H
2919 D3 03              0024        OUT   3
291B 3E CE              0025        MVI   A,0CEH
291D D3 03              0026        OUT   3
291F 3E 27              0027        MVI   A,27H
2921 D3 03              0028        OUT   3
2923 C9                 0029        RET
2924                    0030 *
2924                    0031 *
2924                    0032 * THIS CHECKS FOR <^C>, AND SUSPENDS OUTPUT ON <SPACE>
2924                    0033 *
2924                    0034        ORG   IOLOC+64
2940                    0035 *
2940 CD DC C0           0036 CHK8   CALL  0C0DCH
2943 FE 03              0037        CPI   3
2945 C8                 0038        RZ
2946 FE 20              0039        CPI   32
2948 C0                 0040        RNZ
2949 CD 60 29           0041        CALL  IN8
294C FE 03              0042        CPI   3
294E C9                 0043        RET
294F                    0044 *
294F                    0045 *
294F                    0046 * CONSOLE INPUT - <^P> & <^L> ARE HANDLED SPECIALLY
294F                    0047 *
294F                    0048        ORG   IOLOC+96
2960                    0049 *
2960 CD DC C0           0050 IN8    CALL  0C0DCH        ;GET CHARACTER
2963 CA 60 29           0051        JZ    IN8
2966 FE 10              0052        CPI   16            ;SEE IF ^P
2968 CA 75 29           0053        JZ    CTLP
296B FE 0C              0054        CPI   12            ;SEE IF ^L
```

```
29F3 C1                  0111          POP    B
29F4 78                  0112          MOV    A,B
29F5 C9                  0113          RET                    ;AND RETURN
29F6                     0114 *
29F6 00                  0115 OUTFL    DB     0               ;PRINTER FLAG
29F7 38                  0116 LNCNT    DB     56              ;LINE COUNTER
29F8                     0117 *
```

SYMBOL TABLE

```
CHK8   2940    CTLP   2975    IN8    2960    INI8   2910    IOLOC 2900
ITALF  29E4    LNCNT  29F7    MZOS   2028    OUT8   2980    OUTFL 29F6
OUTPR  29C0    REENT  2900
```

```
0000                    0001 *
0000                    0002 * NORTH STAR TO MZOS DISK COPY PROGRAM
0000                    0003 * NEALE BRASSELL [26-JUN-78]
0000                    0004 *
0000                    0005 * THIS PROGRAM IS 100% RELOCATABLE.
0000                    0006 *
0000                    0007 MZ      EQU   0C402H        ;MZOS DCOM ROUTINE
0000                    0008 NS      EQU   02022H        ;NORTH STAR DCOM ROUTINE
0000                    0009 *
0000                    0010 DJNZ    EQU   10H
0000                    0011 *
0000 3E 0A              0012 NS2MZ   MVI   A,10          ;10 SECTORS AT A TIME
0002 06 23              0013         MVI   B,35          ;35 SUCH TRANSFERS
0004 0E 01              0014         MVI   C,1           ;WE'LL USE DISK 1
0006 11 00 2A           0015         LXI   D,2A00H       ;TYPICAL BUFFER SPOT
0009 21 00 00           0016         LXI   H,0           ;START FROM THE BEGINNING
000C C5                 0017 N2M     PUSH  B             ;SAVE COUNTER
000D 06 01              0018         MVI   B,1           ;READ FROM NS
000F F5                 0019         PUSH  PSW
0010 D5                 0020         PUSH  D
0011 E5                 0021         PUSH  H
0012 CD 22 20           0022         CALL  NS            ;<< DO IT >>
0015 E1                 0023         POP   H
0016 D1                 0024         POP   D
0017 F1                 0025         POP   PSW
0018 06 00              0026         MVI   B,0           ;WRITE TO MZ
001A CD 02 C4           0027         CALL  MZ            ;<< DO IT >>
001D D5                 0028         PUSH  D             ;SAVE DE
001E 11 0A 00           0029         LXI   D,10          ;TEN SECTORS DONE
0021 19                 0030         DAD   D             ;REFLECT THIS
0022 D1                 0031         POP   D             ;RESTORE DE
0023 C1                 0032         POP   B             ;GET COUNTER
0024 10                 0033         DB    DJNZ          ;USE NEAT Z-80 CODE
0025 E6                 0034         DB    N2M-$-1       ;TO EFFECT A LOOP
0026 C9                 0035         RET                 ;HOPEFULLY RETURN TO WHOEVER
0027                    0036 *

SYMBOL TABLE

DJNZ  0010    MZ    C402    N2M   000C    NS    2022    NS2MZ 0000
```

```
2A00                    0001 * LITTLE PROGRAM TO TITLE A DISK
2A00                    0002 * NEALE BRASSELL [12-JUL-78]
2A00                    0003 *
2A00 AF                 0004 TITLE  XRA   A
2A01 06 3E              0005        MVI   B,'>'
2A03 CD 0D 20           0006        CALL  0200DH        ;OUTPUT NOTE
2A06 3E 01              0007        MVI   A,1
2A08 01 01 01           0008        LXI   B,101H
2A0B 11 00 2B           0009        LXI   D,2B00H
2A0E 21 04 00           0010        LXI   H,4
2A11 CD 22 20           0011        CALL  02022H        ;READ IN MZOS
2A14 AF                 0012        XRA   A
2A15 06 3A              0013        MVI   B,':'
2A17 CD 0D 20           0014        CALL  0200DH        ;PROMPT
2A1A 21 00 2B           0015        LXI   H,2B00H
2A1D 06 09              0016        MVI   B,9
2A1F CD 10 20           0017 LOOP   CALL  2010H         ;GET CHARACTER
2A22 FE 0D              0018        CPI   13
2A24 28                 0019        DB    28H           ;IF <CR>, END
2A25 09                 0020        DB    CR-$-1
2A26 47                 0021        MOV   B,A
2A27 AF                 0022        XRA   A
2A28 CD 0D 20           0023        CALL  200DH
2A2B 70                 0024        MOV   M,B
2A2C 23                 0025        INX   H
2A2D 10                 0026        DB    16            ;CONTINUE
2A2E F0                 0027        DB    LOOP-$-1
2A2F 2B                 0028 CR     DCX   H
2A30 3E 80              0029        MVI   A,128
2A32 B6                 0030        ORA   M
2A33 77                 0031        MOV   M,A
2A34 06 3C              0032        MVI   B,'<'
2A36 AF                 0033        XRA   A
2A37 CD 0D 20           0034        CALL  0200DH
2A3A 3E 01              0035        MVI   A,1
2A3C 01 01 00           0036        LXI   B,1
2A3F 11 00 2B           0037        LXI   D,2B00H
2A42 21 04 00           0038        LXI   H,4
2A45 CD 22 20           0039        CALL  2022H         ;WRITE MZOS BACK
2A48 C3 28 20           0040        JMP   2028H

SYMBOL TABLE

CR    2A2F    LOOP  2A1F    TITLE 2A00
```

This is a program which will title a disk for you. The title will be printed whenever you boot the disk. To run, put a system disk into drive 1, copy this program onto it, then type TITLE. The program will print >, indicating that it is reading in MZOS. It will then print : and wait for you to type the title. Maximum is 9 characters, and terminate with CR. Make no errors - no provision is made for backspacing. After CR or nine characters, it will print <, then write MZOS back onto disk. Reboot and observe the result. This program is set up to run at 2A00, and use a buffer from 2B00-2C00. The program itself is relocatable, so you can put it elsewhere by changing the starting address (for example. TY TITLE 1 0 to run at 0000). You can change the program (lines 9 and 15) to use a different buffer area, also. Note that in order to title a disk, it MUST have MZOS as the first file.