MDZ/OS™

Release I

PROGRAMMER Manual

PRELIMINARY

Micro Mike's, Inc.
3015 Plains Blvd.
Amarillo, TX 79102 USA

telephone: 806-372-3633

revisions 01/01
June 22, 1982

making technology uncomplicated ... for People™

TABLE OF CONTENTS

INTRODUCTION

MDZ/OS is a fast, flexible, powerful, and easy to change OEM-type operating system. MDZ/OS is easy to change because of the unique combination of high and low level languages used. All main operating routines which are executed thousands of times a day are written in Z80 assembly language to insure fast and efficient execution. However, a concerted attempt has been made to remove all unnecessary operations so that they are not in the actual operating system and are not executed when they do not need to be.

All programs which set system parameters and most utilities are written in baZic, Micro Mike's, Inc. high-level BASIC interpreter. This BASIC (the MicroDoZ version) is designed so that the interpreter can execute all of the disk operating system commands. All of the programs written in baZic can be modified by a programmer to change the way the system appears to the end user. This flexibility means that a programmer can change completely the way that MDZ/OS appears in a matter of weeks instead of months or years.

MDZ/OS does all of the actual writing of information on the hard disk drive. The servant operating systems (CP/M and MicroDoZ) supply the disk address and MDZ/OS translates that address to find the correct physical location on the disk to write the information. MDZ/OS employs a recursive blocking routine to allow quick access to the entire disk drive. This routine performs read-after-write verification on all disk writes.

If, during a write operation, a sector is encountered that cannot be verified, MDZ/OS discards the sector and "brings in" another sector from a section of the disk that has been reserved for this purpose. This new sector is mapped to appear in the same space as the old sector even though the physical location on the disk is different.

This entire process is invisible to the user of the system. The only time the user should get a hard disk error message from a hard disk is if the drive is physically not working or the very first control sector on the disk has gone bad or all the substitution segments have been used. All other sectors, including the mapping sectors (except the master sector), are substituted automatically when they fail during a write operation.

The MDZ/OS disk now contains several versions of baZic to fit the needs of most everyone. baZics are provided of 8, 10, 12, and 14 digit precisions in

MDZ/OS uses the Z80 Mode 2 interrupts. These interrupts let the processor calculate an interrupt address so that the restart (RST) locations are not required. If your program uses the RST instruction it should continue to work properly under timesharing conditions.

1.1 Differences in CP/M

This section deals with the peculiarities of MDZ/OS CP/M. Since this arrangement is very structured there are certain restrictions which apply to the hard disk that are not normally found on floppy disks.

First, CP/M on the hard disk cannot be SYSGENed or moved using MOVCPM. CP/M is supplied on the hard disk to run a 60K system.

The reason CP/M cannot be SYSGENed is because under MDZ/OS, CP/M is stored as a file under MicroDoZ in the SYSTEM Segment. This special version has been relocated to allow more room for the BIOS since the BIOS has been customized to the hard disk system.

CP/M under MDZ/OS can be run in as many segments as needed by logical drive assignment. The system will install CP/M automatically in the internal memory of the user who is switching to CP/M from MicroDoZ. Since the operating system handles the allocation of CP/M automatically there is no need to do a SYSGEN. CP/M segments may be accessed by different users at the same time but only the first user can write to the segment. All other users are Read Only (R/O).

The reason CP/M cannot be moved is similar. There are certain places where CP/M has to be tied into the executive software and if CP/M were moved, the executive would have no way of determining where CP/M is. For this reason, MDZ/OS does not support the MOVCPM program.

Nothing in CP/M should be changed without consulting Micro Mike's, Inc. for the proper procedure.

CP/M normally boots from the first two tracks of a floppy disk. Under MDZ/OS CP/M boots from a MicroDoZ segment on the hard disk. A machine language program (CPMBOOT) is used to boot CP/M. This program can be executed from the Menu System when the segment is specified to be CP/M.

The boot program loads the CP/M image, the turnkey command and the I/O into memory and moves the constructed CP/M to the proper CP/M location within the slave which requested it.

Turnkey startup of CP/M programs can be accomplished by entering the proper turnkey command from the Menu Editor. The user can specify a cold boot only or that the program is to be booted every time a warm boot is executed by that slave.

EXIT.COM is used to transfer control from the CP/M operating system to MicroDoZ. The program works by calling the reboot location within the slave. The slave is rebooted to the turnkey command for that slave located in the SLTKEY file in the SYSTEM Segment. Any program can be rebooted by calling that location. See the SYSTEM HANDLES section for the exact location to call in the slave executive.

1.2 Back Up Your System

After using a hard disk for one or more years, it becomes very easy to assume the drive will work forever, and many people become lax in their backup procedures. The SYSTEM and SYSTEM1 segments should be backed up regularly to insure that you can retain your data in the event of a failure in the system.

The SYSTEM segment contains the SEGMENT file. This file should be backed up EVERY time that it is changed. This file contains all of the allocations of every segment on your hard disk. If this file is lost, the system has no way to know where your segments are. Effectively your data is lost until this segment is reconstructed.

The MENU files in the SYSTEM1 segment are equally important. These files contain the allocation of segments for each menu item plus the password used to access the option. Each of the menu files ends with an ampersand (&) so they are easy to recognize. Again, these files need to be backed up each and every time they are changed.

You can back up only the files which need backing up or perhaps it is easier to back up the entire SYSTEM and SYSTEM1 segments.

Do not forget to back up any of your data segments from your application programs. Although modern hard disks are generally very reliable, they are produced and used by humans and can fail at any time. Make backing up an integrated part of your use of the system. You should back up data when it takes more time to re-enter the data than it does to make the backup copy.

1.3 System Utilities

This section gives a short explanation for each of the utility programs provided in the MDZ/OS system. A more detailed explanation of each program is provided in the SYSTEM UTILITIES section of this manual.

1.3.1 Setup Utilities

FORMATHD is an assembly language program designed to format the hard disk drive. This process involves writing the sector headers to the drive and the program can erase existing data from the disk. It must be used with caution. This is the first program that is run on the hard disk when setting up the system initially.

MAPHD is an assembly language program which "maps" the sectors into logical sets of sectors. These sets are used by MDZ/OS to organize and structure the disk so that information can be accessed quickly and efficiently. A set equals 256 (512 byte) sectors mapped together. MAPHD writes ASCII 32 (spaces) to the entire disk surface and verifies each write to accomplish an initial check of the quality of the recording surfaces.

Any sectors found to be bad are "thrown away" by the program and only sectors found good are linked together. A small portion of the disk is left not set up (to be used if bad sectors are discovered later). The program, upon completion, reports the number of bad sectors found on the disks.

COPYSYS is used to initiate the copying of the appropriate files from the floppy boot disk to the SYSTEM Segment area of the hard disk.

COPYSYS1 is used to copy the appropriate files (menu system) from the floppy disk to the SYSTEM1 Segment of the hard disk.

SYSFILE is a file located on the floppy disk which has a list of the files which are to be copied to the SYSTEM Segment of the hard disk. These files are:

SEGMENT
JOEMAST
SLVEXEC
SLVMDOZ
BOOTCPM
SLVCPM
SLVBOOT
SLVTKEY
SLVTKEYED
OLAY1
OLAY2
SLVIOX
CONTROL
SYSTEM1
SYSTEM2
BAZIC1Ø
FUNCTION
SEGED

SEGINIT
FORMATFD
M2DØØM
ICOPY
COPYFILE
CLOCK
CPM1
CPMIOX
Z19
SOROC
ADDS
ITUBE
ADM3A
IQ-12Ø
JOETEST
SLVEDIT
CPMIOED

SYS1FILE is a file located on the floppy disk which has a list of the names of files which are to be copied to the SYSTEM1 Segment of the hard disk. These files are:

MENU

MENUCR

MENU1

MENU1ED

MENU1&

CPM

CPMED

CPM&

SUTIL

SUTILED

SUTIL&

1.3.2 SYSTEM Segment Programs and Files

SEGMENT File - This file is perhaps the most important file in the MDZ/OS system since this file stores the allocation of ALL of the Segments defined on the hard disk. **This file should be backed up regularly.** The SEGMENT file is used to convert a logical disk address into a physical disk address. The file contains information on the names of segments, whether it is a CP/M or MicroDoZ segment, the directory size, and the offset on the hard disk where the segment is located. **This file should be backed up every time the segment allocation is changed (SEGED is run).**

JOEMAST is the master executive program which controls the entire system and is actually two executives -- the master executive and the slave executive. The master executive controls all common tasks and communicates with the slaves. The slave executive is responsible for encoding and passing commands to the master. The slave executive resides in the top 4K of memory in each slave while the master executive occupies the entire 64K in the master computer. The executive is written in machine language so as to execute the programs as quickly as possibly. The master executive contains the hard and floppy disk drivers, the DISKTABLE, and the task-queuing routines.

SLVEXEC is a machine language program which resides in the top 4K of memory of each slave. This program contains the slave number, the user's copy of the DISKTABLE and routines for converting operating system calls into MDZ/OS system calls.

SLVMDOZ is the special version of MicroDoZ which resides in each slave. This program is used to pass disk operating system commands to the slave executive.

BOOTCPM is used to boot the CP/M operating system into any slave. This program is called by the Menu system when an option is selected which requires CP/M.

SLVTKEY is a file located in the SYSTEM Segment which contains the turnkey command for each of the 16 slaves allowed in the system. This file is read when each slave is booted or rebooted to determine the proper programs to load in the slave.

SLTKEYED is a baZic program which writes the turnkey commands into the SLVTKEY file. Each slave may have a different turnkey command.

OLAY1 is the machine language program which is responsible for overlaying the appropriate drivers for your CRT. A typical call of the overlay program would be made by placing the following in the turnkey command for a slave:

```
     1>OLAY1,5\Z19,5
```

where "Z19" is the name of the file containing the CRT configuration information for the Zenith Z19 CRT.

OLAY2 is similar to OLAY1 except this overlay program is used to overlay the I/O file for each slave in the system. A sample execution of this program would appear as follows:

```
     1>OLAY2,5\SLVIOX,5
```

Z19 is the CRT overlay file for the Z19 CRT.

SOROC is the CRT overlay file for the SOROC CRT.

ADDS is the CRT overlay file for the ADDS CRT.

ITUBE is the CRT overlay file for the Intertube CRT.

ADM3A is the CRT overlay file for the ADM-3A CRT.

SLVIOX is the default interrupt I/O for each slave in the system.

BREAK is a baZic program which FILLs a jump vector with a jump to "reboot me" when a break is set from a CRT to the CRT serial port on a slave computer. If this program is run upon boot-up, each user will be able to reboot his slave by pressing the BREAK key.

CONTROL is a baZic program which is used to "turn on" the SYSTEM and SYSTEM1 Segments and then branch to the Control Menu (Menu1).

SYSTEM1 is a baZic program which is used to enable the appropriate segments and then branch to the Control Menu.

SYSTEM2 is a baZic program which is used to enable the appropriate segments and then branch to the System Utility Menu.

Control MENU (Menu1) - The Control MENU is the first menu the user sees upon bootup. This program lets the user branch to other menus for specific purposes or lets the user enter into baZic. The Control Menu also is referred to as the MDZ/OS SYSTEM MENU, MENU ONE.

The FUNCTION program is a set of user-defined functions which performs the actual assignment of segments to drive numbers. As many as seven calls of Function A are allowed. The function call is given the segment name, the logical drive assignment, and the physical drive number of where to find the segment. The function returns a value that informs the user whether or not the drive assignments are legal. If all segment assignments are legal then Function B is called. This function FILLs the offset table with the appropriate information and the new logical drive assignments are made. A CAT after executing Functions A and B would yield different results than a CAT before the function was called, assuming that a different assignment of drives was made.

1.3.3 SYSTEM1 Segment Programs and Files

MENU1ED allows the user to edit the Control Menu (Menu1).

MENU1& is the file which contains the allocations of the Control Menu.

CPM is the menu which allows access to CP/M segments.

CPMED is used to edit the CPM menu.

CPM& contains the allocations as defined by the CPM editor.

System MENU (SUTIL) - This menu contains all of the system utilities needed to assign, initialize, test, and determine the characteristics of the segments. In addition, you may create new Menu programs from this menu.

SUTILED is the editor program for the system utility menu.

SUTIL& contains the allocation of menu items for the system utility menu.

1.3.4 BOOT Disk Programs

TKEYEDIT is a baZic program which allows editing the TKEYF or turnkey file. This program is used to set MicroDoZ to boot the hard disk operating system (HDISK), to set whether messages are to be printed to the CRT from the master, and to set the turnkey command for MicroDoZ at the initial boot from the floppy disk.

HDISK is the hard disk operating system which is appended to MicroDoZ if the hard disk boot flag has been set by the TKEYEDIT program.

SETHD is a baZic program which sets the size of hard disk which is to appear as each physical hard disk in the system. The hard disks can be Physical Drives 5, 6, 7 and 8. Each drive may be 1Ø, 14, 2Ø, or 26 megabytes.

MONITOR is a machine language program which is the standard monitor delivered with MicroDoZ. This program allows the viewing, changing, and testing of memory locations.

COMPACT is a machine language program designed to reorganize a floppy disk or hard disk segment to recover space lost because of deletion of files. Compact will compact segments only and not the entire hard disk.

COPYFILE is a copy file routine which allows the user to specify a file-of-file-names file a program will use to make a the back-up. Manual entry of file names is supported or entry can be via a DOSCoMmanD string from a baZic program. When this program has completed a copy, baZic and any program can be chained to automatically.

1.4 Definition of Terms

This section is included to define those terms that may be unique to the product described in this manual. No attempt is made to describe all computer-related terms, as it is assumed that the user of this manual already has a basic understanding of computer systems and the terminology associated with the use of computers.

Block - A block, as used in this manual, is equivalent to a sector. All hard and floppy disks supported by this manual have 512 bytes per sector. A block is 512 bytes.

DISKTABLE - The DISKTABLE is the segment allocation table which is contained in RAM in the executive software. It is the table which allows different operating systems to co-exist on the hard disk. A version of the table is located in the slave executive for each user on the system as well as a master version located in the SysteMaster RAM.

Handle - A handle refers to assembly language TAGS. A handle is either a routine which may be needed by an end-user or the location of a jump that points to the location of the routine. These routines can be accessed, in most cases, by the user to provide additional versatility for specific installations of the software.

Physical Drives - Physical drives refer to the actual disk drives attached to the computer. The physical drive assignment as defined by Micro Mike's, Inc. are: 1 to 4 are floppy disk drives and 5 to 8 are hard disk drives.

Logical Drives - MDZ/OS allows the user to assign the drive number to all segments created on the hard disk and to all other drives on the system (eight-inch). The drive number assignment may be any number from 1 to 7 and does not have to be related to the physical drive number. The drive numbers assigned by the user through the MENU system are called the logical drive assignments.

Set - A set is 256 blocks which are mapped together on the hard disk. The user of this software will not encounter a "set" but the term is used to describe blocks that are "tied together" logically on the hard disk. The master mapping sector is located at Track Ø, Head Ø, Sector Ø. This sector contains a list of 16 bit addresses of all of the set sectors on the hard disk. The set sectors map most of the remaining sectors. A partial set of sectors is reserved for sector substitution in the case where data written to the disk cannot be verified. All sectors (except the master) can be substituted if they fail.

SEGCR is a baZic program which CREATEs the System Segment File. This file controls the allocation of all MicroDoZ and CP/M segments on the disk. This program should be run only once, since running this program after the segment has been created has the effect of erasing the "directory" of all segment allocations.

INPUT/OUTPUT

The input and output (I/O) routines of MDZ/OS are stored in disk files which are overlayed when a slave computer is booted or from the menu system when a CP/M segment allocation is made. Sample listings of each type of I/O can be found in the Appendix. The I/O files can be modified from the monitor, assembled with an assembler or edited with the baZic programs provided.

IOEDIT is used to set the baud rate and other factors associated with the two serial ports and the one parallel port on the SysteMaster. This program also allows the user to set the cursor addressing sequence and clear screen codes for the version of MicroDoZ which boots on the SysteMaster. The CRT must be listed in the menu of CRTs and have an associated CRT file located on the SYSTEM Segment of the hard disk.

SLVEDIT is used to establish the baud rate and other associated parameters of the serial ports on each slave computer. In addition, you may edit the input and output device assignments and save the edited file for use as an overlay file for any or all slaves.

Two overlay programs are provided. OLAY1 is used to overlay the CRT characteristics while OLAY2 is used to overlay the I/O files.

To edit the CP/M I/O use the CPMIOED program. This program will allow you to edit the console and reader input devices and the console, punch, and list output devices.

2.1 IOEDIT

The IOEDIT program is used to edit the parameters associated with the SysteMaster. The serial ports can be edited to change the clock factor, number of stop bits, parity, number of data bits, auto enables (hand shaking), and baud rate. The program also allows the user to define the input and output devices and save the changes under a file name of the user's choice.

The IOEDIT program is not an option on any of the menus and must be loaded and run from baZic. Because the program modifies the TKEY program which is on the boot disk, you must select the NORMAL option (Option 6) from the System Utilities Menu to "turn on" the Floppy Disk Drive 1.

The IOEDIT program does not use cursor addressing or clear screen codes because this is the program which initially sets these codes.

Select the NORMAL Option from the System Utilities Menu and load and run the IOEDIT program. The sequence will appear as follows:

```
READY
LOAD IOEDIT
READY
RUN

SYSTEMASTER CRT AND IO PROT EDITOR
WAIT FOR TKEY PROGRAM TO BE READ
```

There will be a slight pause while a portion of the TKEY program is read into a baZic variable. Once the proper number of bytes has been read, the program will display:

```
CHOOSE OPTION TO EDIT
1.  CRT SERIAL PORT
2.  SERIAL PRINTER PORT
3.  PARALLEL PORT PIO A
4.  CRT CONFIGURATION
RETURN TO SAVE ON DISK
```

Option 1 is used to establish the characteristics of the CRT serial port (SIO A). If you select Option 1, you will see the following display:

```
CRT SERIAL
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

To change any of the previous factors, select the number of the option you want to change. If you are unsure of the meaning or function of any of the options, turn to the subsection on the SIO serial interface chip located in this section. Because all of the serial ports for the master and slave computers are edited in the same manner, the detail of each option is also located in the subsection on the SIO device.

Once you have changed the factors to reflect your situation, press Return to go back to the previous menu.

The characteristics of the printer serial port can be changed by selecting Option 2. The display will be the same as above except the title will be changed to reflect the desired port:

```
SERIAL PRINTER
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

Again, make the changes to fit your situation and press Return to branch to the first menu.

Option 3 is not currently implemented since the parallel port is supported in an output mode only. If Option 3 is selected, you will see:

```
PARALLEL PROT IS OUTPUT MODE ONLY FOR NOW
PRESS ANY KEY TO CONTINUE
```

Once you press any key, you will branch to the first menu.

To establish the cursor addressing sequence and clear screen codes, select Option 4, CRT CONFIGURATION. The screen will appear as follows:

```
CHOOSE TERMINAL CONFIGURATION
1.  ZENITH Z19
2.  SOROC
3.  ADDS
4.  ITUBE
5.  ADM3A
6.  IQ-12Ø
7.  OTHER NOTE FILE MUST BE PRESENT ON DISK
RETURN TO LEAVE UNCHANGED
```

If you do not want to change the terminal configuration, press Return and you will branch to the first menu. If your terminal is listed, you are in luck! Enter the number which corresponds to your CRT and the program will print the following message:

```
READING CRT FILE
CRT FILE HAS BEEN READ
```

```
 CHOOSE OPTION TO EDIT
1.  CRT SERIAL PORT
2.  SERIAL PRINTER PORT
3.  PARALLEL PORT PIO A
4.  CRT CONFIGURATION
RETURN TO SAVE ON DISK
```

Once all your selections have been made, press Return. The following message will be printed on the CRT:

```
WRITING DATA TO TKEY PROGRAM
TKEY PROGRAM UPDATE COMPLETE
READY
```

The TKEY program had now been updated, but you must reboot for the changes to be enabled.

If your CRT is not among those listed, you must advance to the Creating a New CRT File subsection. You will need to use the monitor to modify an existing CRT file to match the characteristics of your CRT and then save the modified file in the SYSTEM Segment. You will then need to add your CRT to the IOEDIT program and run the program, selecting your CRT. Save the edit session by pressing Return and then reboot your system to enable the changes.

2.2 SLVEDIT

The SLVEDIT program is used to set the parameters for the serial ports on each slave computer. The clock factor, number of stop bits, parity, number of data bits, auto enable, and baud rate can all be set by this program. In addition, this program can edit the assignment of all supported input and output devices.

This program uses the SLVIOX or any other user-defined I/O file as a pattern to establish an I/O file with the parameters set to your specifications. When the program is run, the pattern I/O file is read and the parameters displayed. By selecting options you can modify any of the parameters listed and store the modified I/O file back in the SYSTEM Segment of the hard disk.

Once a new I/O file is made, you can enable the I/O for any slave by using the OLAY2 program to overlay the I/O file. All overlays as edited by the SLVEDIT program are used for MicroDoZ I/Os only. All CP/M I/Os are modified either through the CP/M STAT command or through the CPMIOED program.

SLVEDIT is a baZic program and is not listed on any menu so it must be loaded and run. The sequence will appear as follows:

```
SLVEDIT IOFILE EDITOR
ENTER FILENAME,DRIVE OF PATTERN FILE
OR RETURN TO USE SLVIOX,5
?
```

Normally you will enter a Return to use the SLVIOX,5 file as your pattern. Any file which has been created by this program can be used as your pattern file. Enter Return and see:

```
     READING DATA FROM FILE
```

There will be a slight pause while the data is read from the pattern file into a baZic variable. Once the data are read, the program will display the first prompt:

```
     CHOOSE OPTION
     1.  EDIT SLAVE SERIAL IO DEVICES
     2.  EDIT INPUT DEVICE ASSIGNMENTS
     3.  EDIT OUTPUT DEVICE ASSIGNMENTS
     4.  SAVE EDITED FILE
```

If you want to change the baud rate or any other parameters associated with the serial ports on the slave computer, select Option 1. The display will be:

```
     CHOOSE SERIAL DEVICE TO EDIT OR RETURN UNCHANGED
     1.  SERIAL CRT PORT
     2.  SERIAL PRINTER PORT
```

Select 1 or 2 to edit the CRT or the printer port. In either case you will see:

```
     SERIAL CRT (OR PRINTER) PORT
     PRESENT CLOCK FACTOR = 16X
     PRESENT STOP BITS = 2
     PRESENT PARITY OPTION IS OFF
     PRESENT DATA BITS = 8
     PRESENTLY AUTO ENABLES IS OFF
     BAUD RATE = 96ØØ

     SIO EDITOR
     CHOOSE OPTION TO EDIT
     1.  CLOCK FACTOR
     2.  STOP BITS
     3.  PARITY
     4.  DATA BITS
     5.  AUTO ENABLE
     6.  BAUD RATE
     RETURN FOR THE PREVIOUS MENU
```

Any of the listed parameters may now be modified. See Section 2.6 (SIOs) for more information on each of the parameters. Once you have made your selections, enter a Return and the program will branch to the previous menu and you will see:

```
     CHOOSE SERIAL DEVICE TO EDIT OR RETURN UNCHANGED
     1.  SERIAL CRT PORT
     2.  SERIAL PRINTER PORT
```

When both ports have been edited to your specifications, press Return to return to the first menu:

```
CHOOSE OPTION
1.  EDIT SLAVE SERIAL IO DEVICES
2.  EDIT INPUT DEVICE ASSIGNMENTS
3.  EDIT OUTPUT DEVICE ASSIGNMENTS
4.  SAVE EDITED FILE
```

Under MicroDoZ as many as eight input and eight output devices can be defined at any one time. These devices are assigned a number from Ø to 7. Device Ø is normally the "console" CRT and Device 1 is normally the printer. By selecting Options 2 and 3, you can modify the assignment of these I/O devices so that input or output goes to any device supported in the system.

Select Option 2 and you will see:

```
INPUT DEVICE ASSIGNMENT
DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 1  LOCAL SERIAL CRT PORT
DEVICE 2  LOCAL SERIAL CRT PORT
DEVICE 3  LOCAL SERIAL CRT PORT
DEVICE 4  LOCAL SERIAL CRT PORT
DEVICE 5  SYSTEMASTER SERIAL CRT PORT
DEVICE 6  SYSTEMASTER SERIAL PRINTER PORT
DEVICE 7  LOCAL SERIAL CRT PORT
CHOOSE DEVICE Ø..7 OR RETURN TO PREVIOUS MENU
```

If your assignments have been made or you are satisfied with the way they are, press Return and you will return to the first menu. If you select any device (Ø to 7), you will see:

```
CHOOSE DEVICE TO ASSIGN OR RETURN UNCHANGED
1.  LOCAL SLAVE SERIAL CRT PORT
2.  LOCAL SLAVE SERIAL PRINTER PORT
3.  SYSTEMASTER SERIAL CRT PORT
4.  SYSTEMASTER SERIAL PRINTER PORT
```

At the present time, these are the only input devices supported. Select the number of the port you want. It will be assigned to the device number you have specified. As an example, if you select Device 4 to edit and then 3 to assign the SysteMaster serial CRT port to this device, the display will be as follows:

```
INPUT DEVICE ASSIGNMENT
DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 1  LOCAL SERIAL CRT PORT
DEVICE 2  LOCAL SERIAL CRT PORT
DEVICE 3  LOCAL SERIAL CRT PORT
DEVICE 4  SYSTEMASTER SERIAL CRT PORT   <--(note change)
DEVICE 5  SYSTEMASTER SERIAL CRT PORT
DEVICE 6  SYSTEMASTER SERIAL PRINTER PORT
DEVICE 7  LOCAL SERIAL CRT PORT
CHOOSE DEVICE Ø..7 OR RETURN TO PREVIOUS MENU
```

Once all of the input assignments have been made, press Return to return to the first menu:

```
CHOOSE OPTION
1.  EDIT SLAVE SERIAL IO DEVICES
2.  EDIT INPUT DEVICE ASSIGNMENTS
3.  EDIT OUTPUT DEVICE ASSIGNMENTS
4.  SAVE EDITED FILE
```

Select Option 3 to edit the output device assignment. The output assignment is similar to the input assignment except two devices are supported for each output device. This feature allows the user to print to the CRT and the printer at the same time or any two other devices. Once you enter 3, you will see:

```
OUTPUT DEVICE ASSIGNMENTS
DEVICE Ø  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE Ø  SUB DEVICE 1  NULL DEVICE
DEVICE 1  SUB DEVICE Ø  LOCAL SERIAL PRINTER PORT
DEVICE 1  SUB DEVICE 1  NULL DEVICE
DEVICE 2  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 2  SUB DEVICE 1  LOCAL SERIAL PRINTER PORT
DEVICE 3  SUB DEVICE Ø  LOCAL PARALLEL PORT
DEVICE 3  SUB DEVICE 1  NULL DEVICE
DEVICE 4  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 4  SUB DEVICE 1  LOCAL PARALLEL PORT
DEVICE 5  SUB DEVICE Ø  SYSTEMASTER SERIAL CRT PORT
DEVICE 5  SUB DEVICE 1  NULL DEVICE
DEVICE 6  SUB DEVICE Ø  SYSTEMASTER SERIAL PRINTER PORT
DEVICE 6  SUB DEVICE 1  NULL DEVICE
DEVICE 7  SUB DEVICE Ø  SYSTEMASTER PARALLEL PORT
DEVICE 7  SUB DEVICE 1  NULL DEVICE

CHOOSE DEVICE Ø..7 OR RETURN TO PREVIOUS MENU
```

To leave the device assignments as they are, press Return and you will return to the first menu. If any of the devices are edited, the sequence will be similar. The program will ask you to define each of the two sub-devices supported for each device. If you select one of the device numbers, the display will appear as follows:

```
SUB DEVICE Ø
CHOOSE DEVICE TO ASSIGN OR RETURN UNCHANGED
1.  LOCAL SLAVE SERIAL CRT PORT
2.  LOCAL SLAVE SERIAL PRINTER PORT
3.  LOCAL SLAVE PARALLEL PORT
4.  SYSTEMASTER SERIAL CRT PORT
5.  SYSTEMASTER SERIAL PRINTER PORT
6.  SYSTEMASTER PARALLEL PORT
7.  NULL DEVICE
```

You will be assigning the first sub-device. Select the device you want assigned. If you do not want any device assigned to this sub-device, select 7 (NULL DEVICE). Press Return if you do not want the present assignment changed. Once you have made your entry, you will see:

```
SUB DEVICE 1
CHOOSE DEVICE TO ASSIGN OR RETURN UNCHANGED
1.  LOCAL SLAVE SERIAL CRT PORT
2.  LOCAL SLAVE SERIAL PRINTER PORT
3.  LOCAL SLAVE PARALLEL PORT
4.  SYSTEMASTER SERIAL CRT PORT
5.  SYSTEMASTER SERIAL PRINTER PORT
6.  SYSTEMASTER PARALLEL PORT
7.  NULL DEVICE
```

You can now edit the second sub device. Select the port you want, the null device, or press Return to leave the assignment unchanged. At this time you will see:

```
OUTPUT DEVICE ASSIGNMENTS
DEVICE Ø  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE Ø  SUB DEVICE 1  NULL DEVICE
DEVICE 1  SUB DEVICE Ø  LOCAL SERIAL PRINTER PORT
DEVICE 1  SUB DEVICE 1  NULL DEVICE
DEVICE 2  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 2  SUB DEVICE 1  LOCAL SERIAL PRINTER PORT
DEVICE 3  SUB DEVICE Ø  LOCAL PARALLEL PORT
DEVICE 3  SUB DEVICE 1  NULL DEVICE
DEVICE 4  SUB DEVICE Ø  LOCAL SERIAL CRT PORT
DEVICE 4  SUB DEVICE 1  LOCAL PARALLEL PORT
DEVICE 5  SUB DEVICE Ø  SYSTEMASTER SERIAL CRT PORT
DEVICE 5  SUB DEVICE 1  NULL DEVICE
DEVICE 6  SUB DEVICE Ø  SYSTEMASTER SERIAL PRINTER PORT
DEVICE 6  SUB DEVICE 1  NULL DEVICE
DEVICE 7  SUB DEVICE Ø  SYSTEMASTER PARALLEL PORT
DEVICE 7  SUB DEVICE 1  NULL DEVICE

CHOOSE DEVICE Ø..7 OR RETURN TO PREVIOUS MENU
```

You may now edit other output devices. Once all of the assignments have been made, press Return and you will see the first menu:

```
CHOOSE OPTION
1.  EDIT SLAVE SERIAL IO DEVICES
2.  EDIT INPUT DEVICE ASSIGNMENTS
3.  EDIT OUTPUT DEVICE ASSIGNMENTS
4.  SAVE EDITED FILE
```

At this time, you should have the I/O file edited to your satisfaction. If not, continue selecting options until you do have the I/O like you want it. Once everything is in order, select Option 4 to save the edited file. The display will appear as follows:

```
ENTER FILENAME,DRIVE OF DESTINATION FILE
FILE MUST BE ON THE SYSTEM SEGMENT TO USE AS AN OVERLAY
?
```

As an example, you may want to call your I/O file TESTIO,5. The I/O file must be in the SYSTEM Segment which is normally referenced as Drive 5. The file name should have a ",5" to make sure the file is saved in the SYSTEM Segment. You may use any valid file name. To enable the edited file, it must be overlayed in the slave turnkey sequence (SLTKEYED,5) by the OLAY2 program. A sample overlay would appear as follows:

```
1>OLAY2,5\TESTIO,5
```

2.3 CPMIOED

The CPMIOED program is used to define the console and reader input devices, and the console, punch, and list output devices. Of course, these devices can be assigned under CP/M by use of the STAT command. The CPMIOED program allows the user to predefine these I/O devices and to have them overlayed by the menu system when a CP/M Segment selection is made.

The program is not on the menu system and must be loaded and run from baZic. The sequence would appear as follows:

```
READY
LOAD CPMIOED,5
READY
RUN

CPMEDIT  IO FILE EDITOR
ENTER FILENAME,DRIVE OF PATTERN FILE
OR RETURN TO USE CPMIOX,5
?
```

Normally you will use the CPMIOX,5 file. However, any previously defined CPM I/O file can be used. Press Return and you will see:

```
READING FILE
```

The program will pause for a short period while the I/O file is being read. Once the file is read the display will show:

```
CHOOSE OPTION
1.  EDIT CONSOLE INPUT DEVICES
2.  EDIT CONSOLE OUTPUT DEVICES
3.  EDIT READER INPUT DEVICES
4.  EDIT PUNCH OUTPUT DEVICES
5.  EDIT LIST OUTPUT DEVICES
6.  SAVE EDITED FILE
```

When editing any of the CP/M I/O devices, the program will refer to Devices Ø, 1, 2, and 3. This is because any of the four devices (CON:, RDR:, PUN:, OR LST:) can be set to reference any of four devices. The following table shows the relationship between the devices of CP/M.

CPMIOED DEVICES

	Ø	1	2	3
CON: =	TTY:	CRT:	BAT:	UC1:
RDR: =	TTY:	PTR:	UR1:	UR2:
PUN: =	TTY:	PTP:	UP1:	UP2:
LST: =	TTY:	CRT:	LPT:	UL1:

As an example, the List Device (Option 5) will be explained first. For this example, we will setup the LIST Device so that printing to the List Device will print to the SysteMaster Serial Printer Port. Select Option 5 from the menu and you will see:

```
OUTPUT DEVICE ASSIGNMENTS
DEVICE Ø        LOCAL SERIAL CRT PORT
DEVICE 1        LOCAL SERIAL CRT PORT
DEVICE 2        LOCAL SERIAL CRT PORT
DEVICE 3        LOCAL SERIAL CRT PORT

CHOOSE DEVICE Ø..3 OR RETURN TO PREVIOUS MENU
```

Since we are going to edit Device Ø (TTY:) of the LST: device, we will select Device Ø. (Refer to the CPMIOED DEVICE chart above if you are unsure why we selected Device Ø). The display will show:

```
CHOOSE DEVICE TO ASSIGN OR RETURN UNCHANGED
1.  LOCAL SLAVE SERIAL CRT PORT
2.  LOCAL SLAVE SERIAL PRINTER PORT
3.  LOCAL SLAVE PARALLEL PORT
4.  SYSTEMASTER SERIAL CRT PORT
5.  SYSTEMASTER SERIAL PRINTER PORT
6.  SYSTEMASTER PARALLEL PORT
7.  NULL DEVICE
```

Select Option 5 to assign the SysteMaster serial printer port to be the TTY: portion (Device Ø) of the LST: device. If the file resulting from this edit program is overlayed from the menu system, and a print is directed to the LIST device, the printout will appear on the printer connected to the printer serial port on the SysteMaster. Also if the STAT DEV: command is executed, the results will be LST: = TTY:.

Once you have made your selection, or pressed Return to leave the assignment unchanged, the device assignment menu will show on the screen again:

```
OUTPUT DEVICE ASSIGNMENTS
DEVICE Ø        LOCAL SERIAL CRT PORT
DEVICE 1        LOCAL SERIAL CRT PORT
DEVICE 2        LOCAL SERIAL CRT PORT
DEVICE 3        LOCAL SERIAL CRT PORT

CHOOSE DEVICE Ø..3 OR RETURN TO PREVIOUS MENU
```

Edit each device until they meet your approval. Press Return to return to the first menu:

```
CHOOSE OPTION
1.  EDIT CONSOLE INPUT DEVICES
2.  EDIT CONSOLE OUTPUT DEVICES
3.  EDIT READER INPUT DEVICES
4.  EDIT PUNCH OUTPUT DEVICES
5.  EDIT LIST OUTPUT DEVICES
6.  SAVE EDITED FILE
```

All of the output device options can be edited the same as the previous example. Any of the input device options will appear as follows when selected:

```
INPUT DEVICE ASSIGNMENT
DEVICE Ø LOCAL SERIAL CRT PORT
DEVICE 1 LOCAL SERIAL CRT PORT
DEVICE 2 LOCAL SERIAL CRT PORT
DEVICE 3 LOCAL SERIAL CRT PORT
CHOOSE DEVICE Ø..3 OR RETURN TO PREVIOUS MENU
```

Select the device you want to edit for the input device you have selected from the first menu. If you want the assignment to remain unchanged, press Return. Otherwise, enter the number of the device you want to edit and you will see:

```
CHOOSE DEVICE TO ASSIGN OR RETURN UNCHANGED
1.  LOCAL SLAVE SERIAL CRT PORT
2.  LOCAL SLAVE SERIAL PRINTER PORT
3.  SYSTEMASTER SERIAL CRT PORT
4.  SYSTEMASTER SERIAL PRINTER PORT
```

Select the port you want to assign to this device and the display will return to the device assignment menu:

```
INPUT DEVICE ASSIGNMENT
DEVICE Ø LOCAL SERIAL CRT PORT
DEVICE 1 LOCAL SERIAL CRT PORT
DEVICE 2 LOCAL SERIAL CRT PORT
DEVICE 3 LOCAL SERIAL CRT PORT
CHOOSE DEVICE Ø..3 OR RETURN TO PREVIOUS MENU
```

Once all of the assignments are made, press Return to return to the first menu:

```
CHOOSE OPTION
1.  EDIT CONSOLE INPUT DEVICES
2.  EDIT CONSOLE OUTPUT DEVICES
3.  EDIT READER INPUT DEVICES
4.  EDIT PUNCH OUTPUT DEVICES
5.  EDIT LIST OUTPUT DEVICES
6.  SAVE EDITED FILE
```

Now you must select Option 6 to save the edited file. Select 6 and the display will read:

```
ENTER FILENAME,DRIVE OF DESTINATION FILE
FILE MUST BE ON THE SYSTEM SEGMENT TO USE AS AN OVERLAY
?
```

Enter the file name under which you want this assignment stored. An example might be CPMIO,5. To make these changes active, you must use this file name as the CP/M I/O overlay file from the menu system. See the USER manual for more information on using the menu system.

2.4 OLAY1

The OLAY1 program is used to overlay the cursor addressing sequence, clear screen codes, and other CRT-specific information. The OLAY1 program may be called either from MicroDoZ or through the slave turnkey editor (SLTKEYED,5). The overlay command is separated by a backslash (\) from the CRT file which is to be overlayed. The CRT file must have been created using the IOEDIT program or by programmer following the example in the Appendix using the monitor or an assembler.

A typical direct call to overlay the Z19 CRT file for any slave computer in the system would appear as follows:

```
1>OLAY1,5\Z19,5
```

A typical call of the overlay program from the slave turnkey editor would appear as follows:

```
OLAY1,5\ADDS,5\OLAY2,5\SLVIOX,5\BAZIC1Ø,5\CONTROL,5
```

2.5 OLAY2

The OLAY2 program is similar to the OLAY1 program except the OLAY2 program overlays the I/O drives and device assignments. OLAYS would be used to overlay a file created by the SLVEDIT program.

OLAY1 overlays the first 8Ø bytes of the I/O file (such as SLVIOX,5 or Z19,5), while OLAY2 overlays the remainder of the file.

A sample direct call to overlay the standard interrupt slave I/O would be as follows:

```
1>OLAY2,5\SLVIOX,5
```

The OLAY2 program can also be called from the slave turnkey editor program (SLTKEYED,5) as shown in the following example:

```
OLAY1,5\ADDS,5\OLAY2,5\SLVIOX,5\BAZIC10,5\CONTROL,5
```

2.6 SIOs

The SIO is the Z80 companion integrated circuit which is used to convert parallel data coming from the processor to a serial data stream which is used by most CRTs and printers. This device has two complete RS-232 C ports which support a variety of configurations. This "chip" is essentially a dedicated microprocessor which is programmable to meet the needs of many situations.

Items which can be programmed include the clock factor (actually on the counter timer chip which "feeds" a clock pulse to the SIO chip), the number of stop bits, parity, number of data bits, auto enable, and baud rate.

Since the SysteMaster and the SBC-1 slaves each use the SIO chip, there are two programs which have identical routines for programming the SIO chip; IOEDIT and SLVEDIT. From each program the serial port editing section appears as follows:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 9600

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

If you need to change the clock factor, select Option 1. This should be necessary only for baud rates less than 150. Once you select Option 1, you will see:

```
PRESENT CLOCK FACTOR = 16X
16X IS NORMAL
CHOOSE CLOCK FACTOR
Ø.  1X
1.  16X
2.  32X
3.  64X
```

Make your selection and the display will return to the previous menu. The top half of the display will now reflect your selection:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

If you need to change the number of stop bits, select Option 2 and the screen will display:

```
PRESENT STOP BITS = 2
CHOOSE STOP BIT CODE
1. = 1 STOP BIT
2. = 1 1/2 STOP BITS
3. = 2 STOP BITS
RETURN KEY TO LEAVE UNCHANGED
```

Select the number which corresponds to the number of stop bits you want. If you do not want to change the present setting, press the Return key. Once you make your selection the original menu will appear and will reflect the changes you have made:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

If you need to change the parity, select Option 3. The display will show:

```
PRESENT PARITY OPTION IS OFF
CHOOSE Ø) OFF  1) ODD ON  3) EVEN ON OR RETURN UNCHANGED
```

Press Return to leave the parity unchanged. Enter the number which corresponds to the type of parity you want. You may have parity disabled (OFF), parity enabled and odd, or parity enabled and even. Make your selection and the display will return to the first menu:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

To change the number of data bits, select Option 4. The display will read:

```
PRESENT DATA BITS = 8

CHOOSE 5), 7), 6), 8) OR RETURN UNCHANGED
```

The number of data bits can be 5, 6, 7, or 8. Select the number which represents the number of data bits you want. If you do not want to change the present number of data bits, press Return. Once you have made your selection, the display will return to the original menu:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 96ØØ

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

To change the auto enable flag, select Option 5. The auto enable flag is used to "enforce" hand-shaking between devices (printers, etc.) and the SIO chip. With auto enable on, the SIO will honor the hand-shaking protocols defined. As an example, with auto enable on, the SIO will stop sending characters to a printer when the printer informs the SIO that its buffer is full and it cannot except any more characters. The auto enable should be on when using the interrupt I/O overlays.

When Option 5 is selected, the program will display:

```
PRESENTLY AUTO ENABLE IS ON
USE AUTO ENABLE FOR PRINTER HANDSHAKING

INPUT Ø) OFF, 1) ON OR RETURN UNCHANGED
```

Select the situation you want and the program will return to the original menu and show your selection:

```
(Name of port being edited)
PRESENT CLOCK FACTOR = 16X
PRESENT STOP BITS = 2
PRESENT PARITY OPTION IS OFF
PRESENT DATA BITS = 8
PRESENTLY AUTO ENABLES IS OFF
BAUD RATE = 9600

SIO EDITOR
CHOOSE OPTION TO EDIT
1.  CLOCK FACTOR
2.  STOP BITS
3.  PARITY
4.  DATA BITS
5.  AUTO ENABLE
6.  BAUD RATE
RETURN FOR THE PREVIOUS MENU
```

The last item which can be edited is the baud rate. The baud is a measure of the speed of transmitting data down a line. To change the baud rate of a port, select Option 6. The display will show:

```
BAUD RATE = 9600

CHOOSE BAUD RATE OR RETURN UNCHANGED
1.  19200
2.  9600
3.  4800
4.  2400
5.  1200
6.  600
7.  300
8.  150
9.  110
```

Select the number which corresponds with the baud rate you want for this port. If you do not want to change the present baud rate, press Return. Once you have made your selection, the display will return to the original menu.

Once all items have been changed, you must press return and select the save option to place the changes on disk. You must reboot the master to enable any changes made for its SIO or reboot the slave to enable any changes made for its SIO.

2.7 PIO

The PIO is similar to the SIO in that it has two ports but the ports of the PIO are parallel ports. This means that they transmit data 8 bits at a time. Part of the second parallel port is used internally on the boards, so only one parallel port is supported on the master and each slave and this port is supported as an output port only at the present time.

Each PIO is currently supported with software to drive a Centronics-type parallel printer. The SETUP manual has a sample cable configuration and the Appendix contains a sample source listing of the driver routines.

2.8 Creating a New CRT File

New CRT files can be created with an assembler which generates a MicroDoZ file or by using the monitor provided to change one of the existing CRT files to match your CRT and then saving the modified file under the name of your CRT.

The procedure outlined here will follow the latter course. You need to know the cursor addressing sequence, the offsets (if any), and the clear screen code(s) for your CRT. These can generally be found in the manual which comes with your CRT.

You should keep the source listing located in the Appendix near at hand so that you can "see" what you are doing as the procedure progresses.

You should begin by getting into MicroDoZ. If you are in the menu system, execute a Control C to stop the program from executing and return to baZic. Leave baZic by typing BYE and a Return. You should now see the MicroDoZ copyright message and prompt:

```
     1>
```

We will begin by using the SLVIOX,5 file as our pattern. Load the file into RAM by using the LF command. The sequence will appear as follows:

```
     1>LF SLVIOX,5 1ØØ
```

Of course you must terminate all command lines with a Return for the line to be executed. The I/O sections normally run at E8ØØH, but for this modification process we are loading the routines at Ø1ØØH. Therefore you should substitute a "1" for "E8" for all addresses found in the source listing.

Examine the source listing and find the clear screen routine (16 bytes starting at E8Ø4), the cursor addressing sequence (16 bytes starting at E814), and the cursor addressing X and Y offset values located at E824 and E825. These are the routines which will be modified.

Now load the monitor by entering the following sequence:

```
     1>M2DØØM,5

     MONITOR 5.Ø
     >
```

You should now know your clear screen codes. The listing is for a Z19 which has 2 codes (1B and 45). Each sequence must end with a Ø to mark the end of the codes. As an example, we will change the Z19 listing to match the codes of the ADM 3A. The ADM 3A clear screen code is a 1A (1 code clears the screen).

From the monitor use the display and substitute command to change the codes. Remember the file is loaded at Ø1ØØH. The command will be:

```
    >DS 1Ø4
    Ø1Ø4 Ø2=
```

The monitor is now giving you a chance to make this byte equal to whatever value you want. In this case we want a "1" because the number of codes to clear the screen is 1. Enter a 1 and press the space bar (if you press the return key the display and substitute mode will be cancelled). The display will now show:

```
    >DS 1Ø4
    Ø1Ø4 Ø2= 1   1B=
```

Now we can change the 1B to the proper code for clearing the screen on the ADM 3A (1A). Remember to press the space bar once you have entered the code. The screen will now appear as follows:

```
    >DS 1Ø4
    Ø1Ø4 Ø2= 1   1B= 1A   45=
```

We have finished the clear screen sequence except we now need the Ø as the terminator. Enter a Ø and this time press Return to leave the display and substitute mode. The display will now show:

```
    >DS 1Ø4
    Ø1Ø4 Ø2= 1   1B= 1A   45= Ø
    >
```

The clear screen code has been modified so now we can tackle the cursor addressing sequence. In this example, the Z19 cursor addressing sequence is ESC, "Y" (1BH, 59H) and the ADM 3A is ESC, "=" (1B, 3D). Since the number of codes is the same and only the second code is different, we need to change only the second code. Display and substitute Address 116 and change the value from a 59H to a 3DH. The sequence will appear as follows:

```
    >DS 116
    Ø116 59= 3D
    >
```

This time enter a Return because we have only this one code to change. For this example we are finished modifying the code because the X and Y offsets are the same for both (as well as most) CRTs.

To save the modified file, exit the monitor by entering the operating system command as follows:

```
>OS
MICRODOZ COPYRIGHT (C) 1981, MICRO MIKE'S, INC.
1>
```

The next task is to create and type a file to hold the I/O file. The sequence is as follows:

```
1>CR ADM-3A,5 2
1>TY ADM-3A,5 1 E8ØØ
```

The first line creates a file named "ADM-3A" on Drive 5 which is 2 blocks long. The second line types the file as a Type 1 (machine language) file with a go address of E8ØØH.

To save the newly created file, enter the following command:

```
1>SF ADM-3A,5 1ØØ
```

You should now have a working CRT file for your CRT. For the file to be active, it must be included in the slave turnkey command for each slave computer which is to use that CRT.

MicroDoZ uses the cursor addressing and clear screen codes by examining the first byte at the beginning of each sequence. If the code is not an FF, the codes that follow will be output until a Ø is encountered (thus the need to terminate each sequence with a Ø). If the first byte of the sequence is a FF, MicroDoZ will assume that a machine language routine follows and will jump to the byte following the first byte and execute the routine.

This feature can be used to write routines for CRTs which have unusually complex clear screen or cursor addressing sequences. Be sure to terminate each routine with a return (C9H). You have 15 bytes to write each routine.

TURNKEY PROVISIONS

MDZ/OS has provisions to execute any turnkey (auto startup) command or program. Upon bootup, MicroDoZ consults the TKEYF file to determine its inital turnkey command. If this command is to execute the MDZ/OS software, MDZ/OS examines the slave turnkey command to determine the startup program or command for each slave in the system. At this time, each slave can branch to another MicroDoZ program, baZic, baZic program, or CP/M and any of its many commands or programs.

A versatile menu system is provided which is normally branched to upon bootup. This menu system controls passwords, user access, segment assignments, relative drive assignments, and turnkey commands for MicroDoZ and CP/M. This flexibility allows the system to be configured so that every user can be presented upon bootup with their own friendly environment.

Turnkey commands for each slave are stored in the SLVTKEY,5 file and are edited using the program SLTKEYED,5. Turnkey commands for CP/M are stored in the menu system files and can be executed by selecting the proper option from the menu. Also, custom startup programs can be quickly written in baZic to allow any conceivable startup sequence.

3.1 TKEYEDIT

TKEYEDIT is a baZic program and must be loaded and run. TKEY-EDIT's function is to modify the TKEYF file located on the floppy boot disk. To run the program you must first allocate physical Drive 1 from a menu (select the NORMAL option from the System Utility Menu) so that you can access the TKEYEDIT program and TKEYF file. Once you have allocated the floppy disk drive, enter the following sequence:

```
READY
LOAD TKEYEDIT
READY
RUN

TKEYEDIT
CHOOSE OPTION 1..3 TO EDIT OR RETURN TO SAVE
#

1.  HDISK DISABLED
2.  QUIET MODE OFF
3.  TURNKEY COMMAND=
```

The HDISK file is the hard disk operating system program which is "added" to MicroDoZ if HDISK is enabled, indicating the user has a hard disk on the system.

To set the or reset the HDISK flag, select Option 1. You will observe the following on the screen:

```
CHOOSE Ø TO DISABLE OR 1 TO ENABLE HDISK
```

Select 1 to enable or Ø to disable the hard disk software. Once you make your selection, the menu will appear as follows:

```
TKEYEDIT
CHOOSE OPTION 1..3 TO EDIT OR RETURN TO SAVE
#

1.  HDISK DISABLED
2.  QUIET MODE OFF
3.  TURNKEY COMMAND=
```

When HDISK is loaded, it may be running in a single-user or multi-processing mode. In the multi-processing mode there is usually not a CRT on the SysteMaster board. In this situation we don't want the HDISK program printing messages to the SysteMaster ports, so we must turn the quiet flag on. To change the status of the quiet flag, select Option 2. The display will show:

```
ENTER Ø TO DISABLE OR 1 TO ENABLE QUIET FLAG
```

If you enter a Ø you will disable the quiet flag and all messages will be displayed. If you enter 1, the quiet flag will be enabled and all messages will be suppressed. Once you have made your choice, you will be transferred to the menu:

```
TKEYEDIT
CHOOSE OPTION 1..3 TO EDIT OR RETURN TO SAVE
#

1.  HDISK DISABLED
2.  QUIET MODE OFF
3.  TURNKEY COMMAND=
```

The turnkey command is normally JOEMAST,5 but can be any MicroDoZ or baZic program. To change the turnkey command, select Option 3. The display will read:

```
ENTER NEW TURNKEY COMMAND
?
```

Enter the turnkey command of your choice (normally JOEMAST,5). Be sure to enter a return when you finish making your entry. If you want to erase the turnkey command, enter a space followed by a Return. Once you enter a turnkey command and press the Return key, the display will return to the original prompt:

```
TKEYEDIT
CHOOSE OPTION 1..3 TO EDIT OR RETURN TO SAVE
#

1.  HDISK ENABLED
2.  QUIET MODE ON
3.  TURNKEY COMMAND=JOEMAST,5
```

To save the changes, press return. The program will print the following message on the screen to indicate the changes are being saved:

```
SAVING TKEYF
PLACE CHAIN HERE
STOP IN LINE 26Ø
READY
```

To make the software changes active, you must reboot the system. The changes were made to the disk versions and not the RAM versions. You must reboot so that the hard disk software is loaded.

3.2 TKEYF

The TKEYF file contains the HDISK flag, the quiet flag, and the turnkey command. The first byte of the file is the HDISK flag. If this byte is Ø, the HDISK file will not be loaded at boot time. If this byte is a 1, the HDISK file will be loaded and executed upon a boot.

The second byte in the file is the quiet flag. If this byte is set to Ø, all messages will be displayed to the SysteMaster CRT serial port during the boot-up process. If this byte is set to 1, the quiet mode is enabled and all messages will be suppressed.

The two bytes are followed by the turnkey command string. This string is 128 characters long and is transferred to the MicroDoZ command buffer at 8ØH before execution begins of the turnkey command string.

3.3 SLTKEYED

The SLTKEYED program edits the SLVTKEY file which contains the turnkey commands for all 16 users on the system. Any slave can edit the turnkey command for any user. The program can be accessed from the System Utility Menu or can be loaded and run from baZic. To use the program from baZic, follow the procedure listed:

```
READY
LOAD SLTKEYED,5
READY
RUN

SLAVE TURNKEY EDITOR
ENTER SLAVE TO EDIT 1..16 OR RETURN TO EXIT
?
```

Enter the number of the slave you want to edit. The user number can be determined by examining the proper byte in the slave executive. Enter the number and a Return. The program will display the current turnkey command and the prompt:

```
ENTER NEW TURNKEY COMMAND OR RETURN TO LEAVE UNCHANGED
?

BAZIC1Ø,5
```

The turnkey command shown (BAZIC1Ø,5) is a very simple command. The command can be more complicated and can include CRT information as well as I/O overlays and turnkey commands for baZic to load programs.

Presently, the only CRTs implemented are the SOROC, ADDS Viewpoint, ADM-3A (Televideo 912 and 92Ø), and the Zenith Z19 (Heath WH19). If your CRT is not among these or the cursor addressing sequence and clear screen is not compatible, you must see Section 2.8 (Creating a New CRT File) for information on how to configure the system for your CRT.

The new turnkey command should include the overlay command (OLAY1,5) and the name of your CRT. The only names presently supported are: SOROC, Z19, ADDS, ADM3A, and ITUBE. Addendums will list additional terminals as they are added. To enter the turnkey command for the ADM3A, see the following example:

```
OLAY1,5\ADM3A,5\BAZIC1Ø,5\CONTROL,5
```

If you want interrupt-driven input, you should enter the following type of turnkey command:

```
OLAY2,5\SLVIOX,5\OLAY1,5\ADM3A,5\BAZIC1Ø,5\CONTROL,5
```

SLVIOX is the sample interrupt-driven I/O routine. This file can be modified by using the SLVEDIT program. Information can be found on this program in Section 2.2 (SLVEDIT).

If Pin 1Ø (hardware reboot) of the CRT or the printer cable is true, the slave may go into a continued reboot sequence. The system prints an asterisk (*) on the slave CRT when the slave is rebooted. If you boot the system and see several asterisks appearing on the CRT, you will have to cut the trace which comes from Pin 1Ø of each of the serial ports.

All of the additional CRTs listed would be entered the same way except the name of the CRT would be different.

Once you enter Return, the program will store the command and prompt you with the original prompt. Edit the turnkey command for every slave in your system and make the turnkey command the same as for Slave 1.

When you have finished editing all of the slaves in your system, press Return to exit. The program will return to the System Utility Menu.

3.4 SLVTKEY

This file contains room for 128 bytes of turnkey command for each of the 16 slaves in the system. This information is not stored as a string but in a byte manner. User 1's turnkey command is stored first, followed by the turnkey command for each of the additional 15 users.

3.5 CP/M Menu Turnkey Provisions

Turnkeying to CP/M programs is accomplished through the menu system using the menu editor. The editor appears as follows:

```
MDZ MENU EDITOR CPM ITEM 2
CHOOSE OPTION 1..14 TO EDIT OR RETURN TO SAVE
##

1.  OPTION NAME     MARY'S CP/M SEGMENTS
2.  PASSWORD
3.  USER ACCESS     1111111111111111
4.  PRECISION OF BAZIC  1Ø
                         SEGMENT NAME  PHYSICAL DRIVE  LOCK
5.  LOGICAL DRIVE 1      CPM                5            Ø
6.  LOGICAL DRIVE 2      CPM1               5            1
7.  LOGICAL DRIVE 3      FDCPM              2            1
8.  LOGICAL DRIVE 4
9.  LOGICAL DRIVE 5
1Ø. LOGICAL DRIVE 6
11. LOGICAL DRIVE 7
12. CPM COLDBOOT TURNKEY
13. CPMIO OVERLAY         CPMIOX,5
14. CPM TURNKEY CMD       WS
```

Provisions are made to turnkey from the menu system to MicroDoZ and CP/M. Under Option 12 of the menu editor, you may select the segment allocations to be MicroDoZ, CP/M warmboot, or CP/M coldboot. If MicroDoZ is selected the turnkey command can be entered under Option 13. Be sure to include the drive number with the program name.

If Option 12 is set to CP/M cold or warm boot, you can place the I/O file name under Option 13 while the turnkey command is placed under Option 14. The difference between the cold and warm boots as defined by Option 12, is that when a cold boot is specified, the turnkey program is loaded only when the slave is booted into CP/M. If the warm boot is specified, the turnkey program is loaded every time the slave does a warm boot (i.e. eXit from WordStar).

3.6 CONTROL,5

The CONTROL program is used as a branching point by many other programs in the MDZ/OS system. A listing of the CONTROL program is as follows:

```
10 APPEND "FUNCTION,5"
20 GOSUB 50000
30 A=FNA("SYSTEM1",5,4,0)\GOSUB70
40 A=FNB(0)
50 CHAIN "MENU1,5"
60 STOP
70 IF A=0 THEN RETURN
90 IF A=1 THEN RETURN
100 IF A<>-1 THEN 120
110 !A$," SEGMENT NOT FOUND"
120 !A$," WOULD VIOLATE SEGMENT LOCKS"
130 STOP
```

There are several variations you can make to this program to change your system. The first option is to disable the Control C feature of baZic. To make the menu system secure, you should add Line 1 to the CONTROL program which disables the Control C so that users may not stop the menu programs from operating. The line will appear in the program as follows:

```
1 FILL 280,1\REM DISABLE CONTROL C
10 APPEND "FUNCTION,5"
20 GOSUB 50000
30 A=FNA("SYSTEM1",5,4,0)\GOSUB70
40 A=FNB(0)
50 CHAIN "MENU1,5"
60 STOP
70 IF A=0 THEN RETURN
90 IF A=1 THEN RETURN
100 IF A<>-1 THEN 120
110 !A$," SEGMENT NOT FOUND"
120 !A$," WOULD VIOLATE SEGMENT LOCKS"
130 STOP
```

If you need the Control C enabled, simply write a short baZic program which FILLs 280, 0 and put this program as an option in one of the menus. Be sure to use a password for this option so that only those persons with the password will be able to select the enable Control C option.

The next option is to enable the break detect feature of MDZ/OS. When a break key is detected as being pressed by a user, MDZ/OS executes a jump to a specific location within the slave executive. Normally this area is a return and no action is taken. By modifying the CONTROL program, we can fill a jump into this location that jumps to the reboot me location.

You must be using interrupt I/O for this feature to work. Interrupt I/O is enabled by use of the OLAY2,5 program to overlay SLVIOX,5 or one of its derivatives. The CONTROL program will appear as follows with the addition of the break fills:

```
1 FILL 28Ø,1\REM DISABLE CONTROL C
2 A=61543\FILL A,195\FILL A+1,36\FILL A+2,24Ø
1Ø APPEND "FUNCTION,5"
2Ø GOSUB 5ØØØØ
3Ø A=FNA("SYSTEM1",5,4,Ø)\GOSUB7Ø
4Ø A=FNB(Ø)
5Ø CHAIN "MENU1,4"
6Ø STOP
7Ø IF A=Ø THEN RETURN
9Ø IF A=1 THEN RETURN
1ØØ IF A<>-1 THEN 12Ø
11Ø !A$," SEGMENT NOT FOUND"
12Ø !A$," WOULD VIOLATE SEGMENT LOCKS"
13Ø STOP
```

The next option is to change the chain program. If you want the CP/M menu to be branched to upon boot up, change Line 5Ø. Retype line 5Ø to read as follows to make the system come up running the CP/M menu:

```
5Ø CHAIN "CPM,4"
```

As a further option, you may want each user to come up running a different program or menu. This can be accomplished by examining the user byte in the slave executive (the user number can be determined by: !EXAM(61497)) and branching to the proper program. The following example shows a system where user 1 boots to the MENU1 while all other users boot to the CP/M MENU:

```
1 FILL 28Ø,1\REM DISABLE CONTROL C
2 A=61543\FILL A,195\FILL A+1,36\FILL A+2,24Ø
1Ø APPEND "FUNCTION,5"
2Ø GOSUB 5ØØØØ
3Ø A=FNA("SYSTEM1",5,4,Ø)\GOSUB7Ø
4Ø A=FNB(Ø)
5Ø IF EXAM(61497)=1 THEN CHAIN "MENU1,4" ELSE CHAIN "CPM,4"
6Ø STOP
7Ø IF A=Ø THEN RETURN
9Ø IF A=1 THEN RETURN
1ØØ IF A<>-1 THEN 12Ø
11Ø !A$," SEGMENT NOT FOUND"
12Ø !A$," WOULD VIOLATE SEGMENT LOCKS"\STOP
```

MASTER/SLAVE COMMANDS

This section details the commands which are passed from the slave computers to the master. Each command has a command number which is the first byte in the FIFO when the slave generates an interrupt to the master. The second byte may also be part of the command, depending upon the nature of the command. Following the command byte(s) are the parameters and data if applicable.

Commands in MDZ/OS presently originate in the slave and are passed to the master via the FIFO buffer. Later versions will also allow the master to originate commands to be sent to a slave.

The slave fills a 32-byte area with a command. The slave then resets the FIFO address and writes the 32 bytes to the FIFO along with any additional data bytes (512 usually) required. The slave issues an interrupt to the master and waits for the attention bit of the parallel port to indicate completion by the master. The slave then resets the FIFO address and reads 32 bytes back into the 32 byte buffer. The slave may read additional bytes of data determined by the command and the results of the command.

The master, on receipt of a slave interrupt, resets the FIFO on the interrupting slave. The master then reads the first 32 bytes from that slave into a holding buffer. The master then places the slave number into a queue and returns from the interrupt.

The main master task retrieves a slave number from the queue and uses the information in the corresponding holding buffer to parse and perform one of the 16 (presently implemented) tasks.

Illegal primary command numbers in the 32-byte command string are presently designed to reboot the calling slave.

The slave PROM is designed to issue Command Ø to the master upon boot of the slave. The slave PROM is waiting for the master to reply or a Control X from the slave CRT. The Control X allows the slave to enter a local monitor.

For all command strings, the first byte is the primary command.

4.1 Command Ø (BOOTME)

Command Ø is issued by each slave when its PROM is reset. The master then places 256 bytes of boot code into the slave FIFO and toggles the attention bit of slave. This booter code is loaded at ØH in each slave and loads the remainder of the slave executive and MicroDoZ software by making 32-byte calls to the master executive.

4.2 Command 1 (DISK ACCESS)

Command 1 is used when a slave wants to access the disk system. This command is used to read, write, or initialize disk drives or segments. The command summary is as follows:

Command byte number	Command String
1	DEFB 1 *PRIMARY COMMAND
2	DEFB DISK CMD Ø=READ 1=WRITE -1 OR -2= INITIALIZE
3	DEFB LOGICAL DRIVE BIT 7=1
4-5	DEFW SLAVE DMA ADDRESS
6	DEFB NUMBER OF 512 BYTE SECTORS TO TRANSFER
7	DEFB PHYSICAL DRIVE BIT 7=1
8-9	DEFW -SIZE OF DISK OR SEGMENT
1Ø	DEFB NUMBER OF DIRECTORY SECTORS
11-12	DEFW OFFSET Ø FOR FLOPPIES
13-14	DEFW SPARES (LOCKS AND R/O)
15-32	DEFS 16 USED FOR ERROR REPORT (REGISTERS AFTER DISK CALL)

On a successful completion of each call of Command 1, the command will decrement the number of sectors to transfer and increment the disk address. The data is held in the FIFO immediately after the 32-byte command string.

A read operation involves placing the appropriate command string in the FIFO, and then issuing an interrupt to the master computer and waiting for the master to toggle the attention bit which indicates the operation has been performed. Once the attention bit is toggled, the slave then reads the results checking for error conditions. If the operation was performed correctly, the 512 bytes of data are read from the FIFO and placed in the slave memory.

A write operation involves placing the appropriate command string in the FIFO followed by the 512 bytes of data to be written. The master is interrupted and the slave waits for the attention bit to go true before the slave reads the command string to determine if any errors occurred during the write operation.

If interrupt I/O is enabled on the slave, input can be accepted by the slave while waiting for the disk operation to finish.

4.3 Command 2 (DISK SIZE)

The disk size command is used to return the size of a floppy disk drive. The size of a hard disk segment is determined by looking the segment up in the slave disk table. The command parameters are:

Command byte number	Command String
1	DEFB 2 PRIMARY COMMAND
2	DEFB LOGICAL DRIVE +80H
3-4	DEFW DISKTABLE ADDRESS FOR LOGICAL DRIVE
5-12	DEFS 8 DISKTABLE ENTRY
13-32	DEFS 20 WHICH ON RETURN CONSIST OF DEFB DRIVE CODE DEFW AF, DEFW HL, DEFW BC, DEFW BC

On return from the call, the DEFS 20 consists of the information about the floppy drive. The following table lists the representation of each of the registers:

A = SIZE/10
HL = -SIZE
C = DIRECTORY SECTORS
E = FLOPPY TYPE where

0 = 128 bytes per sector 1 side
1 = 256 bytes per sector 1 side
2 = 512 bytes per sector 1 side
4 = 128 bytes per sector 2 sides
5 = 256 bytes per sector 2 sides
6 = 512 bytes per sector 2 sides

NOTE: On floppies, some "garbage" is returned to the disk table using the disksize call (0F021H) in the slave executive. The size and directory sectors are in the registers correctly but the disk table does not reflect the correct values and is not used.

4.4 Command 3 (FILELOCKING)

Command 3 is used to lock and unlock files. The primary command is 3 but the subcommand determines whether the file is opened (openlock), locked, unlocked,or closed. The command summary for Command 3, is:

Command byte number	Command String
1	DEFB 3 PRIMARY COMMAND
2	DEFB SUBCMD 0=OPENLOCK
3	DEFB PHYSICAL DRIVE
4-5	DEFW DISKADDRESS+OFFSET
6	DEFB ERROR CODE 0=OK 1=BAD SUBCMD 2=NO ROOM 3=BAD ADDRESS 4=FILE LOCKED
7-8	DEFW RETURNED FILELOCK ADDRESS

OR

1	DEFB 3 PRIMARY CMD
2	DEFB SUBCMD 1=LOCK 2=UNLOCK 3=CLOSE
3-4	DEFW FILELOCK ADDRESS RETURNED BY OPENLOCK
5	DEFB UNUSED
6	DEFB ERROR CODE

The master maintains a table of 16 users allowing 8 files per user. This table consists of:

Byte number	Command String
1	DEFB NUMBER OF USERS IN FILE
2	DEFB USER LOCKING FILE
3	DEFB PHYSICAL DRIVE
4-5	DEFW DISKADDRESS+OFFSET

This method of file locking does not support overlapping files. Overlapping files are not locked. Files are locked only if all users are using the file-locking calls and refer to the same physical drive and absolute disk address for the start of the file.

4.5 Command 4 (RETURN USER)

Command 4 returns the slave number from the master computer. The command parameters are as follows:

Command byte number	Command String
1	DEFB 4 PRIMARY COMMAND
2	DEFB RETURNED USER (SLAVE) NUMBER

4.6 Command 5 (FETCH DISK TABLE)

This command is used to fetch and lock that portion of the master disk table which concerns the slave making the fetch call. This command moves 56 bytes from the master disk table (for the calling slave) to the slave disk table. Issuing this command places the slave number in a lock byte in the master executive. A lock error occurs if the lock byte isn't Ø or the same as the calling slave.

The calling parameters are:

Command byte number	Command String
1	DEFB 5 PRIMARY COMMAND
2	DEFB ERROR CODE Ø=OK 1=BAD USER NUMBER 2=LOCKERROR

4.7 Command 6 (STORE DISK TABLE)

Command 5 is used to fetch the disk table from the master. Once the disk table is within the slave, it is usually modified to reflect changing conditions. Once the changes have been made, the disk table is restored to the master by Command 6. The lock byte is unlocked if the operation is successful. The parameters passed are:

Command byte number	Command String
1	DEFB 6 PRIMARY COMMAND
2	DEFB ERROR CODE Ø=OK 1=BAD USER# 2=LOCKING ERROR

This command moves 56 bytes from the slave disk table to the master disk table (for that slave). A lock error occurs if the master disk table lock byte is not equal to the calling slave number.

4.8 Command 7 (SEARCH DISK TABLE)

This command is used to search for a specific entry in the disk-table. This command is used directly from baZic in the menu system. A fetch of the disk table is a prerequisite to a search. The basic principle is find out if other users are using a particular drive or segment of a hard disk drive. This command helps control the segment locking feature of MDZ/OS.

A typical situation would be to fetch and lock the disk table using Command 5. The segment file is opened and a particular segment to be enabled is searched for. If found, the search disk table command is executed to determine if the segment is locked or not locked. If everything is OK, the slave disk table is filled to represent the assignment of the segment. When all segments are placed in the slave disk table, the store command is sent.

Note that if the search finds a locked segment, the lock can be violated deliberately. If the menu systems are used, this condition will not occur. Remember to end the sequence with a "store disk table" command so that other users will not be locked out of the disktable.

The command summary is as follows:

Command byte number	Command String
1	DEFB 7 PRIMARY COMMAND
2	DEFB NUMBER OF ITEMS TO SEARCH (USE 1)
3	DEFB PHYSICAL DRIVE TO SEARCH FOR
4-5	DEFW OFFSET OF SEGMENT TO SEARCH FOR
6	DEFB ERROR CODE Ø=NOT FOUND 1=INVALID ARGUMENTS 2=DISK TABLE NOT LOCKED BY FETCH DISKTABLE 3=FOUND BUT NOT LOCKED 4=FOUND AND LOCKED BY ANOTHER USER

4.9 Command 8 (DEVICE LOCKS)

Command 8 is used to store or fetch two 512 byte buffers kept in the master executive. Each buffer has an associated lock byte. When the buffer is fetched, the lock byte is set so that other users cannot access the buffer. One buffer is used for I/O device locking while the other buffer is used as a user-defined locking buffer.

The user-defined lock buffer can be used in any manner desired by the system programmer. As a convention, the programmer should store a Ø in a specific byte to indicate the buffer is unlocked and the user number (1 to 16) to indicate which user has any portion of the buffer. The meaning associated with each byte of the buffer is left to the system programmer.

The I/O lock table consists of two-byte entries for every I/O device in the system. The first byte codes the condition of the device while the second byte is the lock byte. This is the table which is cleared by the printer release routines.

This table will ultimately be prefilled by the master executive editor when completed. Presently only the bytes controlling master I/O devices are used. Room is reserved so that all slaves', masters', and PSIOs' I/O ports can be defined. Some room for expansion is also reserved in the I/O lock buffer.

The first byte for each I/O device codes the state of the I/O device such that:

Ø=NOT IN THE SYSTEM
1=ALLOCATED TO A SPECIFIC USER
2=SHARED DEVICE LOCKED BY THE FOLLOWING BYTE
OTHER VALUES MAY BE USED AS THE SYSTEM GROWS
SPECIFICALLY VALUES FOR DEVICES USED FOR THE SPOOLER

The command summary for Command 8 is:

Command byte number	Command String
1	DEFB 8 PRIMARY COMMAND
2	DEFB SUBCMD Ø=FETCH 1=STORE
3	DEFB BUFFER NUMBER Ø=IODEV 1=USER
4	DEFB ERROR CODE Ø=OK 1=BAD ARG 2=LOCKED BY ANOTHER

NOTE: Storing the buffers without fetching returns a lock error.

4.1Ø Command 9 (DRIVE LOCK)

This command allows the user to fetch or store a drive lock table kept in the master executive. A lock byte is associated with this table so that only one user can access the table at a time. When the table is fetched, the lock is set until that user calls the command to store the table to the executive. All slave disk accesses look at this table and if locked by another slave, return a disk error.

Command byte number	Command String
1	DEFB 9 PRIMARY COMMAND
2	DEFB SUBCMD Ø=FETCH 1=STORE
3	DEFB ERROR CODE Ø=OK 1=BAD ARG 2= LOCKERROR
4-32	DEFS 29 PHYSICAL DRIVE 1..29

NOTE: A Ø means the drive is unlocked or the numbers 1 to 16 indicate the slave number which has the drive locked. A ØFFH indicates the drive is locked because it is being formatted. Storing the drive table without fetching it causes a lock error to be generated.

4.11 Command 1Ø (FORCE BOOT)

Command 1Ø is used to force the reboot of a slave. A normal use of this command would be to reboot the slave when the BREAK key is pressed and the break mode is enabled.

Since the slave will be rebooting, nothing is returned to the slave when this command is executed. The master will print the slave booted message and user number to the master CRT serial port if the quiet byte of MDZ/OS is set to Ø. The slave is then reset so that the slave will issue Command Ø. MDZ/OS will then consult the SLVTKEY file to determine the proper boot commands for that slave.

The only parameter is the command number. The parameter summary is as follows:

Command byte number	Command String
1	DEFB 1Ø

4.12 Command 11 (LOCK DISK TABLE)

This command is used when the system is booting a slave. This command locks the disk table to preclude a "fetch disk table" command before the disk table has been stored properly. Command 11 allows the freshly booted slave to store a clean image of its disk table in the master disk table.

The command parameters are:

Command byte number	Command String
1	DEFB 11 PRIMARY COMMAND
2	DEFB ERROR CODE Ø=OK 1=BAD ARG 2=LOCKERROR

4.13 Command 12 (TIME OF DAY)

Command 12 allows the slave to set or read the time of day clock on the SysteMaster. The master uses one CTC channel pulsing at a 1Hz rate on interrupts to keep the time. See the baZic program CLOCK for an example of how to use this command to call for the time of day.

The command parameters are:

Command byte number	Command String
1	DEFB 12 PRIMARY COMMAND
2	DEFB SUB CMD Ø=MASTER TO SLAVE (READ) <>Ø=SLAVE TO MASTER(SETTIME)
3	SECONDS
4	MINUTES
5	HOURS
6	DAY OF MONTH
7	MONTH
8	YEAR

4.14 Command 13 (FORMAT FLOPPY DISK)

This command allows a slave to format a floppy disk. The FORMATFD program uses this command, if the program is executed from a slave. A format floppy disk program could be written from baZic or an other high-level language. Care should be used with this command because data can be lost if the "wrong" floppy disk is formatted.

Command byte number	Command String
1	DEFB 13 PRIMARY COMMAND
2	DEFB "A","B","C" SIZE CODE A=128 B=256 C=512 BYTES PER SECTOR
3	DEFB DRIVE NUMBER Ø..3 Ø=DRIVE1
4	DEFB FILL CHARACTER 2ØH=MDOZ ØE5H=CPM OR ANY OTHER VALUE YOU WANT Ø..255
5	DEFB ERROR CODE Ø=OK 1=LOCKED DRIVE 2=WRITE PROTECTED DISK 3=DISKERROR 4=BAD SIZE CODE 5=BAD DRIVE# 6=FORMAT IS IN USE BY ANOTHER SLAVE
6	DEFB TRACK ØFFH ON FIRST CALL
7	DEFB COUNTER ØFFH ON FIRST CALL

This command is issued one track at a time. If the track is formatted without problem, the routine will return without an error message. If no errors are reported, repeat the command using the returned command string until all tracks have been formatted. The counter will count down to zero. If the disk is a two-sided disk, the drive number will change to 4 plus the original drive number.

4.15 Command 14 (DISPATCH I/O)

This command is used when the absolute I/O device is NOT local to the slave. Provision has been made for the later addition of slave-to-slave communications. The spooler will not be involved in this command except for using some of the subroutines in printing.

The command parameters are:

Command byte number	Command String
1	DEFB 14 PRIMARY COMMAND
2	DEFB SUB COMMAND Ø= DO OUTPUT 1= DO INPUT 2= DO PANIC 3= DO INSTAT 4= DO OUTSTAT
3	DEFB IO DEVICE NUMBER
4	DEFB ORIGINATING SLAVE NUMBER 1..16
5	DEFB RETURN CODE Ø=OK 1=DEVICE NOT READY OR BUFFER IS FULL 2=ILLEGAL DEVICE (FROM 512 BYTE IO DEVICE TABLE) 3=DEVICE SPECIFIC TO ONE USER 4=DEVICE IS LOCKED BY ANOTHER USER 5=PARALLEL PORT NOT IN CORRECT MODE 6=OUTPUT IN BURST LEFT DATA NOT

```
                              OUTPUTTED
                              (NOTE BURST NOT IMPLEMENTED)
                              7=INVALID SUB COMMAND
                              8=INVALID SLAVE NUMBER
                              ORIGINATING OR DESTINATION
                              FROM DEVICE CODE
                              9=IOLOCK LOCKED BY ANOTHER

6                        DEFB NUMBER OF BYTES TO FOLLOW
7-32                     DEFS DATA BYTES
```

Odd numbers are outputs when even numbers are inputs. The I/O device numbers are defined as follows:

Ø..5 LOCAL SLAVE IO
6..1Ø1 OTHER SLAVES (6 DEVICES PER SLAVE)
1Ø2.113 PSIO1
114..125 PSIO2
126..131 SYSTEMASTER

Slave device numbers (which are not yet implemented) are defined by the slave number times 6 plus the following to arrive at any specific I/O device:

Ø=SIOA INPUT
1=SIOA OUTPUT
2=SIOB INPUT
3=SIOB OUTPUT
4=PIOA INPUT
5=PIOA OUTPUT

The assignment of PSIO ports (not yet implemented) are defined as the PSIO number (Ø or 1) times 12 plus 1Ø2 plus the following:

Ø=SIO1A INPUT
1=SIO1A OUTPUT
2=SIO1B INPUT
3=SIO1B OUTPUT
4=SIO2A INPUT
5=SIO2A OUTPUT
6=SIO2B INPUT
7=SIO2B OUTPUT
8=PIOA INPUT
9=PIOA OUTPUT
1Ø=PIOB INPUT
11=PIOB OUTPUT

For the I/O on the SysteMaster, add 126 to the following port assignments:

Ø=SIOA INPUT
1=SIOA OUTPUT
2=SIOB INPUT
3=SIOB OUTPUT
4=PIOA INPUT (NOT YET IMPLEMENTED)
5=PIOA OUTPUT (CENTRONICS)

NOTE: The PIO is output-only in the current version.

4.16 Command 15 (PRINTER RELEASE)

This command is used to release the printer so that another user may have access to the printer. This command uses the calling slave number to unlock the I/O devices listed in the I/O lock table.

The command parameters are as follows:

Command byte number	Command String
1	DEFB 15 PRIMARY COMMAND
2	DEFB DEVICE NUMBER ØFFH FOR ALL
3	DEFB ERROR CODE Ø=OK 1= LOCK ERROR

The printer release command is sent to MDZ/OS by each slave when one of the following conditions is met:

1. User program prints a release character
2. User reboots
3. A baZic program executes a CHAIN or READY
4. A CP/M program executes a warm boot

4.17 Additional Commands

More commands will be added when slave-to-slave communications and the spooler are completed.

MENU SYSTEM

The menu system is used to control the access of multiple users to the hard disk. The menu system provides for password protection, restricted user access, segment and drive assignments, relative drive assignments, segment locking, and turnkey commands for MicroDoZ and CP/M. For the menu system to provide security, the Control C Flag in baZic must be set so that Control C is not enabled. If the flag is not set to disable Control C, any user can stop the menu system and by pass the security. With the Control C disabled, the user must find some program which "bombs" before he can gain access to the system.

The menu system consists of several programs and files. The MENU program is the heart of the entire system. Each defined menu APPENDs the MENU program. The only difference between the menu programs and the menu editor programs is the Z9 variable defined in each menu or menu editor program. All menus are created with the same six first letters. The menu program name can consist of up to six letters. As an example, if the menu file name is TEST, the menu editor program is called TESTED and the file which contains the information collected by the editor is called TEST&. The MENUCR program is used to create new menus.

5.1 MENU

The MENU program is a baZic program so the source listing is available by loading the program and making a listing. If the menu system is used to allocate all segments, the segment locking will be controlled completely by the menu system. Segments specified as locked will keep more than one user from using the segment at any one time.

The MENU program is appended to every menu or menu editor program. The menu and menu editor programs simply supply the name of the file which stores the menu parameters and a variable to signify if the MENU program is to allow the editor mode.

When an option is selected from the menu system, the MENU program consults the SEGMENT file to determine if all of the segments actually exist. MENU also checks the segment locking to determine if any of the segments are locked. Next the segments are allocated and if the menu item specifies a turnkey command, the proper program is loaded and run.

5.2 MENU1

MENU1 is the control menu and is used as an example. A complete listing of this program is:

```
     5 APPEND "MENU,4"
     1Ø A$="MENU1&,4"\A1$="MENU1"
     2Ø Z9=Ø
```

Line 5 appends the menu program which actually does the work. Line 1Ø defines the file name (MENU1&,4) where the menu parameters are stored and the name which is to be displayed when the menu is run (MENU1). Line 2Ø indicates that this is to be a menu program with no ability to edit the menu selections.

5.3 MENU1ED

This is the editor portion of MENU1. If you select the editor option from the control menu, you will chain to this program. The only difference between this program and the menu program is the Z9 variable. In this case the variable is set to 1 which indicates to the MENU program that editing is allowed. A complete listing of the MENU1ED program is as follows:

```
     5 APPEND "MENU,4"
     1Ø A$="MENU1&,4"\A1$="MENU1"
     2Ø Z9=1
```

5.4 MENU1&

This is the file which stores the parameters edited by the MENU1ED program. This is an example file, but all of the menu files have the same format. All of the menu files and programs are currently restricted to the SYSTEM1 Segment of the hard disk.

The first byte in each file contains the number of records in the file. For all present MDZ/OS files the number will be 1Ø since there are 1Ø options on each menu. The remaining parameters are stored in a single string. In this example, the string is called C$ and the items contained within the file are defined by their positions within the string . The string format for the menu system files is:

Position	Description
C$(1,2Ø)	Option Name
C$(21,26)	Pass Word
C$(27,33)	User Access Code
C$(34,35)	Precision of baZic

For Logical Drive 1

C$(36,43)	Segment Name
C$(44,44)	Physical Drive Number

For Logical Drive 2

C$(45,52)	Segment Name
C$(53,53)	Physical Drive Number

For Logical Drive 3

C$(54,61)	Segment Name
C$(62,62)	Physical Drive Number

For Logical Drive 4

C$(63,7Ø)	Segment Name
C$(71,71)	Physical Drive Number

For Logical Drive 5

C$(72,79)	Segment Name
C$(8Ø,8Ø)	Physical Drive Number

For Logical Drive 6

C$(81,88)	Segment Name
C$(89,89)	Physical Drive Number

For Logical Drive 7

C$(9Ø,97)	Segment Name
C$(98,98)	Physical Drive Number

C$(99,1Ø8)	Chain Program
C$(1Ø9,117)	User Access Codes
C$(118,118)	CP/M Flag (Ø=MicroDoZ, 1=CPMCold, 2=CPMwarm)
C$(119,138)	2Ø Byte CP/M Command String
C$(139,145)	Lock Code for each Logical Drive
C$(146,146)	If=1 then use IO Manipulation Table
C$(147,154)	Input Device Table
C$(155,17Ø)	Output Device Table
C$(171,255)	Reserved for future expansion

5.5 MENUCR

MENUCR is a baZic program which defines new menu systems. This program works by asking the user the name of the menu he wants to define. The program then "writes" two baZic programs; the menu program and the menu editor program. When these programs are written, MENUCR the creates the file and makes two automatic entries. The menu editor is defined as Option Ø and a Return to Menu 1 is defined as Option 9.

SYSTEM HANDLES

The executive software has several locations which might need to be examined or modified by the end-user. These locations are documented in this section. A handle is defined simply as any location in memory that is or points to a routine that might be needed by software other than MicroDoZ, baZic or MDZ/OS. All handles are supplied with Hex addresses unless specifically stated otherwise.

This section is designed for programmers who have a basic understanding of 8Ø8Ø/Z8Ø machine language programming. If you have problems interfacing this software please contact the dealer from whom this software was purchased or Micro Mike's, Inc. All references to single letters (A, HL, etc.) are references to 8Ø8Ø or Z8Ø registers.

6.1 DOS Handles

See the **MicroDoZ** manual for information.

6.2 Master Executive Handles

The following section deals with the DISKTABLE. This is the section of the executive that directly controls the offsets of the segments. The FUNCTION programs FILL this table to change the allocation of segments dynamically.

The DISKTABLE element consists of 8 bytes:

```
DB         *Physical Drive Number
DW         *-Segment Size (256*256-segment size)
DB         *The number of sectors used for directory
           *Sector is 512 bytes for hard disk and double
            density floppy disks
DW OFFSET  *The amount of offset added to disk addresses
            within the segment
DW Ø       *Reserved for future lock-out routines
```

The DISKTABLE is composed of seven elements for each user. Each element corresponds to a drive number one to seven. Thus each user has 56 bytes reserved. Since there can be sixteen users on the system simultaneously, there are 56*16 bytes reserved in this table.

6.3 Slave Executive Handles

The slave executive software resides in each slave's memory from ØFØØØH to ØFFFFH. The first portion of the slave executive is described here and will remain unchanged as to the location of the handle through future revisions. Additional handles will be defined as features are added to the software.

ØFØØØH JMP INIT

This routine presently sets up IM2 with I register = ØFFH but may do more in future releases.

The following I/O calls are active only if the interrupt I/O is being used.

ØFØØ3H JMP DOPANIC

This routine does the panic detect (Control C) for MicroDoZ and baZic. The calling parameters are:

```
HL POINTS TO A DEVICE TABLE
A= Ø..7 CHANNEL NUMBER
ABSOLUTE DEVICE=(HL+A)
RET NC NZ NO CHARACTER DETECTED
RET C CHARACTER DETECTED Z IF CONTROL C NZ IF NOT
```

ØFØØ6H JMP DOINPUT

This routine waits for input and returns the character in the A register with Bit 7 equal to Ø. The calling parameters are:

```
HL POINTS TO DEVICE TABLE
A= Ø..7 CHANNEL NUMBER (INPUT#7 FROM BAZIC)
ABSOLUTE DEVICE=(HL+A)
RET NC A=DATA BIT7=Ø
RET C  A= ERROR CODE
```

ØFØØ9H JMP DOOUTPUT

This routine outputs a character to the proper device. If the device is defined but not ready, the routine waits until the device comes ready to output the character. If interrupt input is enabled, data can still be entered into the slave while the slave is waiting for the print device to come ready. The routine outputs the character in the B register. Upon return, the B register is equal to the A register.

The calling parameters are:

```
HL POINTS TO DEVICE TABLE
A= Ø..7 CHANNEL NUMBER
IF BIT 7 OF A=Ø THEN 1 DEVICE IN TABLE
IF BIT 7 OF A=1 THEN 2 DEVICES IN TABLE PER CHANNEL
RET NC A=B= DATA OUTPUT
RET C A=ERROR CODE LOCKED ILLEGAL DEVICE ETC.
```

ØFØØCH JMP DOINSTAT

This routine returns the status of the requested input port. The return code is in Register A and has the following meaning:

```
A=Ø DATA READY(INP) OR READY FOR DATA (OUTPUT)
A=1 DATA NOT READY
A=2 NON EXISTENT DEVICE
A=3 NOT A SHARED DEVICE
A=4 DEVICE IS SHARED BUT LOCKED BY ANOTHER USER
A=5 PARALLEL PORT IN INCORRECT MODE
A=6 BURST MODE HAS DATA REMAINING (NOTE BURST NOT
IMPLEMENTED YET)
A=7 INVALID SUB CODE SENT TO MASTER
A=8 INVALID SLAVE NUMBER ORIGIN OR DESTINATION
A=9 IOLOCK TABLE WAS LOCKED BY ANOTHER.
```

ØFØØFH JMP DOOUTSTAT

This routine returns the status of the requested output port. The calling parameters are:

```
HL POINTS TO DEVICE TABLE
A=Ø..7 CHANNEL NUMBER
BIT 7 OF A=Ø THEN 1 DEVICE PER CHANNEL
BIT 7 OF A=1 THEN 2 DEVICES PER CHANNEL
```

The code returned from this routine is in Register A and has the following meaning:

```
A=Ø DATA READY(INP) OR READY FOR DATA (OUTPUT)
A=1 DATA NOT READY
A=2 NON EXISTENT DEVICE
A=3 NOT A SHARED DEVICE
A=4 DEVICE IS SHARED BUT LOCKED BY ANOTHER USER
A=5 PARALLEL PORT IN INCORRECT MODE
A=6 BURST MODE HAS DATA REMAINING (NOTE BURST NOT
IMPLEMENTED YET)
A=7 INVALID SUB CODE SENT TO MASTER
A=8 INVALID SLAVE NUMBER ORIGIN OR DESTINATION
A=9 IOLOCK TABLE WAS LOCKED BY ANOTHER.
```

ØFØ12H JMP DISK ROUTINES

This routine is used to read or write information to the disk system. The calling parameters are:

```
HL= STARTING DISK ADDRESS
DE= DMA ADDRESS
B= COMMAND 0=WRITE 1= READ -1 OR -1 INITIALIZE
C= LOGICAL DRIVE NUMBER BIT 7=1
A= NUMBER OF 512 BYTE SECTORS
```

The disk routines translate logical to physical drives and then look up the segment offset from the slave disk table. Upon completion of the command, the following represents the error condition:

```
RET NC OPERATION WAS OK
RET C ERROR CODE IN A REGISTER.
```

0F015H JMP DEISM

This routine executes the following Z80 machine code sequence:

```
LD E,(HL)\ INC HL\ LD D,(HL)\ INC HL\ RET
```

0F018H JMP ARRAY0

This routine is used to locate values within an array. The routine is as follows:

```
0F018H JMP ARRAY0
        ARRAY0 INC A
        ARRAY DEC A\ RET Z\ ADD HL,DE\ JR ARRAY
0F01BH JMP ARRAY
```

0F01EH JMP GET DISK TABLE

This routine is used to return the starting position for the requested drive in the disk table.

```
A= DRIVE 1..7
CALL DTAB
RET HL= 1ST BYTE OF 8 FOR LOGICAL DRIVE IN DISK TABLE
```

0F021H JMP DISKSIZE

This routine returns the size of the disk. The logical drive number is passed in Register C.

```
C= LOGICAL DRIVE+80H
```

0F024H JMP REBOOTME CALLS MASTER WITH CMD 10

This routine is used to reboot a slave. The routine can be called by performing a machine language jumpt to this location. The sequence of events that follow the execution of this command are:

1. The master resets the slave
2. The slave PROM sends Command Ø to the master
3. The master loads the boot code into the slave
4. The boot code makes disk calls to the master to reboot the slave

ØFØ27H DEFINE WORD (LOW,HIGH) THE ADDRESS OF THE SLAVE DISK TABLE

These two bytes contain the address of the slave disk table.

ØFØ29H DEFB Ø HARD DISK INITIALIZE

This byte is used as a flag to determine if the MicroDoZ initialize command is to be allowed to execute on the hard disk. If the byte is Ø then the executive will NOT allow a hard disk segment to be initialized. If this byte is not equal to Ø, the executive will permit the initialize command to be executed in a hard disk segment.

The following are used for file locking. They have been set up so calls from baZic can affect the file lock. In the future, baZic will have reserved words which accomplish these functions.

ØFØ2AH JMP OPENLOCK

This routine opens a file in the lock mode. This means the file is to be locked as it is used by a slave. The calling parameters are:

```
A= LOGICAL DRIVE 1..7 +8ØH
DE= DISK ADDRESS
CALL OPENLOCK
(PUTS AN ENTRY INTO FILELOCK TABLE IN MASTER)
RET NC HL= ADDRESS IN MASTER FILE LOCK TABLE TO BE USED
```

In subsequent calls for lock, unlock, and close:

```
RET C HL= ERROR CODE
1= BAD MODE 2=NOROOM IN TABLE 3= BAD ADDRESS 4= FILE LOCKED
BY ANOTHER USER (USED WITH ALL FILELOCK CALLS)
```

ØFØ2DH JMP LOCKFILE

```
DE= ADDRESS RETURNED BY OPENLOCK FOR THAT FILE
CALL LOCKFILE
RET NC = FILE LOCKED
RET C HL= ERROR CODE
```

ØFØ3ØH JMP UNLOCKFILE

```
DE=ADDRESS RETURNED BY OPENLOCK
CALL UNLOCKFILE
RET NC = FILE UNLOCKED
RET C HL=ERROR CODE
```

ØFØ33H JMP CLOSEFILE

This routine removes the file entry from the master file lock table.

```
DE= ADDRESS RETURNED BY OPENLOCK
CALL CLOSEFILE
RET NC= OK
RET C HL=ERROR
```

61494 **ØFØ36H JMP RETUSER**

Returns the user number.

61497 **ØFØ39H DEFB USER**

The system on booting calls RETUSER and saves the user number in USER. USER will be a number between 1 and 16 and can be examined by a program to determine the user number.

ØFØ3AH JMP FETCH DISK TABLE

This routine copies 56 bytes from the master disk table to the user (slave) disk table. It invokes a lock in the master to keep other users from accessing the disk table until the present user is finished. The resulting code is returned in the HL register pair as follows:

```
Ø=OK, 1= BAD USER ,2= LOCKED BY ANOTHER
```

ØFØ3DH JMP STORE DISK TABLE

This routine uses Command 6 to copy 56 bytes from the slave disk table to the master disk table. The master disk table is unlocked. The following error code is returned in the HL register pair:

```
ERRCODE Ø=OK 1=BADUSER 2= LOCKED
```

61504 - **ØFØ4ØH JMP PRINTER RELEASE**

This routine issues Command 15 and clears all shared devices locked by this user. In later releases, this routine will also end the acquire phase of the spooler.

The following fetch or store the two 512-byte buffers from the master. Locking and unlocking is implicit in the call. The result is returned in HL.

```
61507-ØFØ43H JMP FETCH BUFFER Ø
61510-ØFØ46H JMP FETCH BUFFER 1
61513-ØFØ49H JMP STORE BUFFER Ø
61516-ØFØ4CH JMP STORE BUFFER 1
```

The following routines fetch or store 29 bytes of master memory. The DE register pair equal the address in the slave to fetch to or restore from. After Command 9 is invoked, HL returns such that:

HL= RETURN CODE Ø=OK 1=INVALID CMD 2= LOCKING ERROR

Fetching locks the master table while storing unlocks the master table. The array of bytes transferred allow the locking of physical drives where Byte 1 equals Physical Drive 1, etc. The byte values are:

Ø= NOT LOCKED TO ANYONE
1..16= LOCKED BY ONE USER
ØFFH= LOCKED FOR FORMATING OR OTHER REASON.

ØFØ4CH JMP DRIVE LOCK FETCH
ØFØ4FH JMP DRIVE LOCK STORE

ØFØ52H JMP RETURN FIFO BUFFER ADDRESS IN HL

The FIFO buffer is 32 bytes long and is used to send or receive commands to and from the master.

ØFØ58H JMP DIRECT COMMAND

This routine allows the user to call the command buffer directly. The following parameters are used:

DE= ADDRESS IN SLAVE MEMORY OF 32 BYTE BUFFER FILLED CORRECTLY WITH MASTER SLAVE COMMAND
CALL DIRECT COMMAND
RESULT IS RETURNED IN THE 32 BYTE BUFFER.

This command sequence must be used with discretion and should not be used with commands that involve more than 32-byte transfers. Note that formatting of floppy disks can be accomplished from baZic using this feature.

ØFØ5BH JMP LOCK DISKTABLE CMD 11

This routine is used when booting to lock the disk table for a subsequent store disk table without a previous fetch.

The following are used when interrupt I/O is active:

ØFØ5EH DEFB Ø PIOSTATUS Ø=RDY FOR CHAR 1=BUSY 2=NOT FOR OUTPUT
ØFØ5FH DEFB Ø PRESENTLY UNUSED BUT RESERVED

ØFØ6ØH RXTABLE PATTERNS USED ON SIOS INITIALIZED BY SLVIOX

```
DEFB ØEAH DTR,8BITS,TXEN,RTS
DEFB Ø *NOTE SET DTR OR RTS FOR HANDSHAKING ON INPUT
DEFB ØEAH
DEFB Ø
```

The following vectors are used when a break condition is received by the SIO. These locations are normally filled by a return, but the BREAK program fills them with a jump to the reboot me location. The user may fill these locations with a jump to any location or routine.

ØFØ64H JMP REBOOT *SIO ON PRINTER
ØFØ67H JMP REBOOT *SIO ON CRT

These vectors are similar to those described above except they are used if data carrier detect is to be used to reboot the system or cause another function to happen. The user may insert a jumpt to the reboot me location or to any other routine.

ØFØ6AH RET DEFW Ø *PRINTER SIO PORT
ØFØ6DH RET DEFW Ø *CRT SIO PORT

61553- **ØFØ71H JMP PRINT RELEASE1**

By calling this routine with the device number in A, you may release the print device for any specified printer.

ØFØ74H JMP REPORT (MAY BE FILLED WITH A RETURN)

This routine may be used to report system errors which may occur. It can be used to report errors in I/O device selection such as locked devices, illegal devices, etc. If these errors are not to be reported, fill this location with a machine language return.

61559- **ØFØ77H JMP DORELP**

This is a table-driven printer release routine used when a printer release character is detected in the I/O routine. The following parameters apply:

```
HL POINTS TO A DEVICE TABLE
A=Ø..7 CHANNEL NUMBER
A=A+8ØH IF 2 DEVICES PER CHANNEL
```

6.4 baZic Handles

See the baZic manual for information.

6.5 CP/M Handles

The handles for the 6ØK version of CP/M are:

CCØØH	Start of CCP
D4ØØH	Start of BDOS
E2ØØH	Start of BIOS
EFØØH-EFFFH	I/O overlay area

APPENDIX 1

Slave MicroDoZ I/O Listing

```
0000             10 *SLVIO1
0000             20 *6-18-82
0000             30 IOORG EQU 0E800H
0000             40       ORG IOORG
E800             50 TSP EQU IOORG+510
E800             60 *
E800 18          70 PAGE DEFB 24 *LINES PER CRT PAGE
E801 01          80 VERIFY DEFB 1 *READ AFTER WRITE IF 1
E802 00          90 CONFIG DEFB 0 *CONFIGURATION BYTE FOR QUAD
                    5 INCH DRIVE
E803 00          100 DSYS DEFB 0 *IF NON ZERO DON'T LIST SYSTEM
                     FILES
E804             110 *CSCR AND GOTO 1ST BYTE IS STRING LENGTH
E804             120 *IF 1ST BYTE = 0FFH THEN MACHINE CODE MUST
                     FOLLOW
E804 02          130 CSCR DEFB 2 *CLEAR SCREEN
E805 1B          140.      DEFB 27
E806 45          150       DEFB "E"
E807 00          160       DEFB 0
E808             170       ORG CSCR+16
E814 02          180 GOTO DEFB 2 *POSITION CURSOR
E815 1B          190       DEFB 27
E816 59          200       DEFB "Y"
E817 00          210       DEFB 0
E818             220       ORG GOTO+16
E824             230 *XOFF AND YOFF OFFSET ADDED TO VALUE TO
                     POSITION CURSOR
E824 1F          240 XOFF DEFB 31
E825 1F          250 YOFF DEFB 31
E826             260 *TKEY TURNKEY THE DOS HERE
E826             270 1ST BYTE IS LENGTH OF STRING
E826             280 *MUST END WITH 0DH (CARRIAGE RETURN)
E826 00          290 TKEY DEFB 0
E827             300       DEFS 80
E877 0D          310       DEFB 0DH
E878             320 *BKSP THE STRING USED TO BACKSPACE ENDS
                     WITH 0
E878 08          330 BKSP DEFB 8
E879 20          340       DEFB 32
E87A 08          350       DEFB 8
E87B 00          360       DEFB 0
E87C             370       DEFS 4
E880             380 *BKSPSET PARSE KEYBOARD CHARACTERS USED TO
                     BACKSPACE
E880 FE 03       390 BKSPSET CP 8
E882 C8          400             RET Z
```

```
E883 FE 5F         41Ø          CP 95
E885 C8            42Ø          RET Z
E886 FE 11         43Ø          CP 11H
E888 C8            44Ø          RET Z
E889 FE 7F         45Ø          CP 127
E88B C9            46Ø          RET
E88C               47Ø *
E88C 21 94 E9      48Ø TINTO LD HL,INTDAT
E88F F3            48Ø       DI
E89Ø 7E            49Ø TINT2        LD A,(HL)
E891 FE FF         49Ø              CP -1
E893 2Ø 3Ø         49Ø              JR NZ,TINT1
E895 3A A4 E9      5ØØ              LD A,(QED1)
E898 32 6Ø FØ      5ØØ              LD (RXTABLE),A
E89B 3A B1 E9      51Ø              LD A,(QED2)
E89E 32 62 FØ      51Ø              LD (RXTABLE+2),A
E8A1 3A CD E8      52Ø              LD A,(QED1M)
E8A4 32 61 FØ      52Ø              LD (RXTABLE+1),A
E8A7 3A CE E8      53Ø              LD A,(QED2M)
E8AA 32 63 FØ      53Ø              LD (RXTABLE+3),A
E8AD 3A DB E8      54Ø              LD A,(PARAOUT)
E8BØ 32 5E FØ      54Ø              LD (PIOSTAT),A
E8B3 21 CF E8      55Ø              LD HL,DOBREAK
E8B6 11 64 FØ      55Ø              LD DE,BREAKØ
E8B9 Ø1 ØC ØØ      55Ø              LD BC,12
E8BC ED BØ         55Ø              LDIR
E8BE 3E FF         56Ø              LD A,-1
E8CØ 32 7Ø FØ      56Ø              LD (OVERLAYFLG),A
E8C3 FB            57Ø              EI
E8C4 C9            57Ø              RET
E8C5 46            58Ø TINT1 LD B,(HL)
E8C6 23            58Ø       INC HL
E8C7 4E            58Ø       LD C,(HL)
E8C8 23            58Ø       INC HL
E8C9 ED B3         59Ø       OTIR
E8CB 18 C3         59Ø       JR TINT2
E8CD ØØ            6ØØ QED1M DEFB Ø *MASK FOR INPUT HANDSHAKE IF
                       BUFFER FULL
E8CE ØØ            61Ø QED2M DEFB Ø
E8CF               62Ø *FOLLOWING MOVED INTO BREAKØ,BREAK1,DCDØ
                       AND DCD1 BY
E8CF ØØ            63Ø DOBREAK NOP
E8DØ ØØ            63Ø         NOP
E8D1 C9            63Ø         RET
E8D2 ØØ            64Ø         NOP
E8D3 ØØ            64Ø         NOP
E8D4 C9            64Ø         RET
E8D5 ØØ            65Ø         NOP
E8D6 ØØ            65Ø         NOP
E8D7 C9            65Ø         RET
E8D8 ØØ            66Ø         NOP
E8D9 ØØ            66Ø         NOP
E8DA C9            66Ø         RET
E8DB ØØ            67Ø PARAOUT DEFB Ø *2 IF PARALLEL INPUT
E8DC               68Ø *
```

```
E8DC C3 4Ø FØ      69Ø PRINTREL1 JP PRINTREL2
E8DF               7ØØ *
E8DF CD 74 FØ      71Ø DOERR CALL REPORT
E8E2 3E Ø3         71Ø       LD A,3
E8EF ED 7B FE E9   72Ø DORET LD SP,(TSP)
E8E8 C9            72Ø       RET
E8E9               73Ø *
E8E9               74Ø       ORG IOORG+1ØØH
E9ØØ 18 ØC         75Ø INP JR INPUT1
E9Ø2 18 1D         76Ø OUT JR OUTPUT
E9Ø4 18 3E         77Ø PANIC JR PANICS
E9Ø6 18 84         78Ø TINT JR TINTO
E9Ø8 18 4C         79Ø INSTAT JR INSTATS
E9ØA 18 48         8ØØ OUTSTAT JR OUTSTATØ
E9ØC 18 CE         81Ø PRINTREL JR PRINTREL1
E9ØE               82Ø *
E9ØE               83Ø *VECTORS TO SLAVE EXEC
E9ØE               84Ø SLAVE EQU ØFØØØH
E9ØE               85Ø *TABLE DRIVEN IO HL=TABLE
E9ØE               86Ø *A=DEVICE IF BIT7=1 THEN DOUBLE TABLE ON
                       OUTPUT ONLY
E9ØE               87Ø *B=CHAR TO OUTPUT
E9ØE               88Ø DOPANIC EQU SLAVE+3
E9ØE               89Ø DOINPUT EQU SLAVE+6
E9ØE               9ØØ DOOUTPUT EQU SLAVE+9
E9ØE               91Ø DOINSTAT EQU SLAVE+ØCH
E9ØE               92Ø DOOUTSTAT EQU SLAVE+ØFH
E9ØE               93Ø ARRAYØ EQU SLAVE+18H
E9ØE               94Ø RXTABLE EQU SLAVE+6ØH
E9ØE               95Ø PIOSTAT EQU SLAVE+Ø5EH
E9ØE               96Ø PRINTREL2 EQU SLAVE+4ØH
E9ØE               97Ø *PRINTREL USED TO RELEASE ALL SHARED
                       DEVICES ON BOOT
E9ØE               98Ø DOREL EQU SLAVE+77H *TABLE DRIVEN RELEASE
E9ØE               99Ø REPORT EQU SLAVE+74H *REPORTS LOCKED OR
                       BAD IO DEVICES
E9ØE               1ØØØ *OVERLAYFLG<>Ø IF INTERRUPT IO OVERLAYED
E9ØE               1Ø1Ø OVERLAYFLG EQU SLAVE+7ØH
E9ØE               1Ø2Ø *BREAK IS RET OR JMP VECTOR
E9ØE               1Ø3Ø *BREAKØ IS SIOA BREAK1 IS SIOB OR CRT
E9ØE               1Ø4Ø *USED TO VECTOR IF BREAK DETECTED
E9ØE               1Ø5Ø BREAKØ EQU SLAVE+64H
E9ØE               1Ø6Ø BREAK1 EQU SLAVE+67H
E9ØE               1Ø7Ø *DCD IS RET OR JMP VECTOR
E9ØE               1Ø8Ø *DCDØ IS SIOA DCD1 IS SIOB
E9ØE               1Ø9Ø *USED TO VECTOR IF DEVICE DISCONNECTS
E9ØE               11ØØ DCDØ EQU SLAVE+6AH
E9ØE               111Ø DCD1 EQU SLAVE+6DH
E9ØE               112Ø *
E9ØE               113Ø *
E9ØE ED 73 FE E9   114Ø INPUT1 LD (TSP),SP
E912 31 FE E9      114Ø        LD SP,TSP
E915 E5            114Ø        PUSH HL
E916 21 7C E9      114Ø        LD HL,INTAB
E919 CD Ø6 FØ      115Ø        CALL DOINPUT
```

```
E91C E1             1150          POP HL
E91D 38 C0          1150          JR C,DOERR
E91F 18 C3          1150          JR DORET
E921 ED 73 FE E9    1160 OUTPUT   LD (TSP),SP
E925 31 FE E9       1160          LD SP,TSP
E928 F6 80          1160          OR 80H
E92A E5             1160          PUSH HL
E92B 21 84 E9       1160          LD HL,OUTTAB
E92E F5             1170          PUSH AF
E92F 78             1170          LD A,B
E930 FE 03          1170          CP 3 *PRINTREL CHAR
E932 20 07          1180          JR NZ,OUTPUT1
E934 F1             1180          POP AF
E935 CD 77 F0       1180          CALL DOREL
E938 E1             1180          POP HL
E939 18 3C          1180          JR DORET1
E93B F1             1190 OUTPUT1  POP AF
E93C CD 09 F0       1200           CALL DOOUTPUT
E93F E1             1200           POP HL
E940 38 9D          1200           JR C,DOERR
E942 18 33          1200           JR DORET1
E944 ED 73 FE E9    1210 PANICS   LD (TSP),SP
E948 31 FE E9       1210          LD SP,TSP
E94B E5             1210          PUSH HL
E94C 21 7C E9       1210          LD HL,INTAB
E94F CD 03 F0       1220 PANIC2   CALL DOPANIC
E952 18 23          1220          JR DORET1
E954 18 10          1230 OUTSTAT0 JR OUTSTATS
E956 ED 73 FE E9    1240 INSTATS  LD (TSP),SP
E95A 31 FE E9       1240           LD SP,TSP
E95D E5             1240           PUSH HL
E95E 21 7C E9       1240           LD HL,INTAB
E961 CD 0C F0       1250           CALL DOINSTAT
E964 18 11          1250           JR DORET1
E966 ED 73 FE E9    1260 OUTSTATS LD (TSP),SP
E96A 31 FE E9       1260            LD SP,TSP
E96D F6 80          1260            OR 80H
E96F E5             1260            PUSH HL
E970 21 84 E9       1260            LD HL,OUTTAB
E973 CD 0F F0       1270            CALL DOOUTSTAT
E976 E1             1270            POP HL
E977 ED 7B FE E9    1280 DORET1   LD SP,(TSP)
E97B C9             1280          RET
E97C                1290 *
E97C                1300 *
E97C                1310 *SLAVE IO PORTS
E97C                1320 SIOADAT EQU 0
E97C                1330 SIOAST EQU 1
E97C                1340 SIOBDAT EQU 2
E97C                1350 SIOBST EQU 3
E97C                1360 PIOADAT EQU 4
E97C                1370 PIOAST EQU 5
E97C                1380 PIOBDAT EQU 6
E97C                1390 PIOBST EQU 7
E97C                1400 CTC0 EQU 8
```

```
E97C             1410 CTC1 EQU 9
E97C             1420 CTC2 EQU 10
E97C             1430 CTC3 EQU 11
E97C             1440 SIOVECTOR EQU 0E0H
E97C             1450 PIOAOUT EQU 0F0H
E97C             1460 PIOAINP EQU 0F2H
E97C             1470 *
E97C 02          1480 INTAB DEFB 2 *SLV CRT
E97D 02          1490       DEFB 2 *SLV CRT
E97E 02          1500       DEFB 2 *SLV CRT
E97F 02          1510       DEFB 2 *SLV CRT
E980 02          1520       DEFB 2 *SLV CRT
E981 7E          1530       DEFB 7EH *MASTER CRT PORT
E982 80          1540       DEFB 80H *MASTER 2ND SERIAL PORT
E983 02          1550       DEFB 2 *SLVCRT
E984             1560 *
E984 03          1570 OUTTAB DEFB 3
E985 FF          1570        DEFB -1 *SLAVE CRT
E986 01          1580        DEFB 1
E987 FF          1580        DEFB -1 *SLAVE SERIAL PRINTER
E988 03          1590        DEFB 3
E989 01          1590        DEFB 1 *SLAVE CRT AND SERIAL
                             PRINTER
E98A 05          1600        DEFB 5
E98B FF          1600        DEFB -1 *SLAVE PARALLEL PRINTER
E98C 03          1610        DEFB 3
E98D 05          1610        DEFB 5 *SLV CRT AND PARA PRINTER
E98E 7F          1620        DEFB 7FH
E98F FF          1620        DEFB -1 *MASTER CRT PORT
E990 81          1630        DEFB 81H
E991 FF          1630        DEFB -1 *MASTER SERIAL PRINTER PORT
E992 83          1640        DEFB 83H
E993 FF          1640        DEFB -1 *MASTER PARALLEL OUT
E994             1650 *
E994 02          1660 INTDAT DEFB 2  *NUM OF BYTES
E995 08          1670        DEFB CTC0 *FOR SIO A
E996 47          1680        DEFB 47H
E997 08          1680        DEFB 8 *9600 BAUD
E998 02          1690        DEFB 2
E999 09          1700        DEFB CTC1 *FOR SIO B
E99A 47          1710        DEFB 47H
E99B 08          1710        DEFB 8 *9600 BAUD
E99C             1720 *
E99C 09          1730        DEFB 9
E99D 01          1740        DEFB SIOAST
E99E 18          1750        DEFB 18H *RESET
E99F 14          1760        DEFB 14H *RESET EXTERNAL STATUS
E9A0 4C          1770        DEFB 4CH *X16 2STOP NO PARITY
E9A1 03          1780        DEFB 3 *WR REG 3
E9A2 E1          1790        DEFB 0E1H *8BIT RXEN AUTO EN
E9A3 05          1800        DEFB 5 *WR REG 5
E9A4 EA          1810 QED1      DEFB 0EAH *8BIT DTR RTS TXEN
E9A5 11          1820           DEFB 11H *RESET EXT STAT
E9A6 19          1830           DEFB 18H+1 *INT ALL RX AND
                                EXTERNAL STATUS
```

```
E9A7           1840 *
E9A7 ØB        1850              DEFB 11
E9A8 Ø3        1860              DEFB SIOBST
E9A9 18        1870              DEFB 18H *RESET
E9AA Ø2        1880              DEFB 2 *IVECT
E9AB EO        1890              DEFB SIOVECTOR
E9AC 14        1900              DEFB 14H *RESET EXT STAT
E9AD 4C        1910              DEFB 4CH *X16 2STOP NO PARITY
E9AE Ø3        1920              DEFB 3 *WR REG 3
E9AF E1        1930              DEFB ØE1H *8 BIT RX EN AUTO EN
E9BØ Ø5        1940              DEFB 5 *WR REG 5
E9B1 EA        1950 QED2         DEFB ØEAH *8BIT DTR RTS TX EN
E9B2 11        1960              DEFB 11H *RESET EXT STAT
E9B3 1D        1970              DEFB 1CH+1 *INT RX STAT AFFECT
                                 VECTORS EXST
E9B4           1980 *
E9B4 Ø3        1990              DEFB 3
E9B5 Ø5        2000              DEFB PIOAST
E9B6 FØ        2010              DEFB PIOAOUT *IVECT
E9B7 ØF        2020              DEFB ØFH *OUTPUT
E9B8 87        2030              DEFB 87H *ENABLE INTS
E9B9           2040 *
E9B9 FF        2050              DEFB -1
E9BA           2060 *
E9BA           2070 *NOTE TSP AT END OF PAGE AND
E9BA           2080 *TEMPORARY STACK OVERWRITES INITIALIZE
                    TABLE
E9BA           2090 *
```

Slave CP/M I/O Listing

```
0000                 10 *CPMIOS
0000                 20 *COPYRIGHT 1982 MICRO MIKE'S INC.
0000                 30 *6-23-1982
0000                 40 *
0000                 50 *EXECUTIVE HANDLES
0000                 60 SLAVE EQU 0F000H
0000                 70 DOPANIC EQU SLAVE+3
0000                 80 DOINPUT EQU SLAVE+6
0000                 90 DOOUTPUT EQU SLAVE+9
0000                 100 DOINSTAT EQU SLAVE+0CH
0000                 110 DOOUTSTAT EQU SLAVE+0FH
0000                 120 RXTABLE EQU SLAVE+60H
0000                 130 PRINTREL2 EQU SLAVE+40H
0000                 140 DORELP EQU SLAVE+77H
0000                 150 REPORT EQU SLAVE+74H
0000                 160 OVERLAYFLG EQU SLAVE+70H
0000                 170 BREAK0 EQU SLAVE+64H
0000                 180 BREAK1 EQU SLAVE+67H
0000                 190 DCD0 EQU SLAVE+6AH
0000                 200 DCD1 EQU SLAVE+6DH
0000                 210 *
0000                 220   ORG 0EEE9H
EEE9                 230 IOBYTE EQU 3
EEE9                 240 *
EEE9 C3 98 EF        250 PUNCH JP PUNCH1
EEEC C3 8D EF        260 READER JP READER1
EEEF C3 A3 EF        270 CONST JP CONST1
EEF2 C3 4F EF        280 CONIN JP CONIN1
EEF5 C3 61 EF        290 CONOUT JP CONOUT1
EEF8 C3 82 EF        300 LIST JP LIST1
EEFB C3 C4 EF        310 LISTST JP LISTST1
EEFE C3 40 F0        320 PRINTREL JP PRINTREL2
EF01 22 15 EF        330 SUBSTACK LD (TEMP),HL
EF04 E1              330          POP HL
EF05 ED 73 FE EF     330          LD (TSP),SP
EF09 31 FE EF        330          LD SP,TSP
EF0C C5              340          PUSH BC
EF0D D5              340          PUSH DE
EF0E E5              350          PUSH HL
EF0F 2A 15 EF        350          LD HL,(TEMP)
EF12 E3              350          EX (SP),HL
EF13 E5              350          PUSH HL
EF14 C9              350          RET
EF15 00 00           360 TEMP DEFW 0
EF17                 370 TSP EQU 0EFFEH
EF17 E1              380 GETSTACK POP HL
EF18 D1              380          POP DE
EF19 C1              380          POP BC
EF1A ED 7B FE EF     380          LD SP,(TSP)
EF1E C9              380          RET
EF1F                 390 *
EF1F                 400 *
```

```
EF1F 3A 03 00      410 PARSE LD A,(IOBYTE)
EF22 A6            410       AND (HL)
EF23 23            410       INC HL
EF24 46            410       LD B,(HL)
EF25 23            410       INC HL
EF26 04            420       INC B
EF27 05            430 PARSE1 DEC B
EF28 28 04         430        JR Z,PARSE2
EF2A 0F            440        RRCA
EF2B 0F            440        RRCA
EF2C 18 F9         440        JR PARSE1
EF2E E6 03         450 PARSE2 AND 3
EF30 C9            450        RET
EF31               460 *
EF31 03            470 CONINTAB DEFB 3 *MASK
EF32 00            480          DEFB 0 *ROTATES
EF33 02            490          DEFB 2 *SIOA
EF34 02            500          DEFB 2
EF35 02            510          DEFB 2
EF36 02            520          DEFB 2
EF37               530 *
EF37 03            540 CONOUTAB DEFB 3 *MASK
EF38 00            550          DEFB 0 *ROTATES
EF39 03            560          DEFB 3 *SIOA
EF3A 03            570          DEFB 3
EF3B 03            580          DEFB 3
EF3C 03            590          DEFB 3
EF3D               600 *
EF3D 0C            610 READTAB DEFB 0CH *MASK
EF3E 01            620         DEFB 1 *ROTATES
EF3F 02            630         DEFB 2 *SIOA
EF40 02            640         DEFB 2
EF41 02            650         DEFB 2
EF42 02            660         DEFB 2
EF43               670 *
EF43 30            680 PUNCHTAB DEFB 30H
EF44 02            690          DEFB 2 *ROTATES
EF45 03            700          DEFB 3 *SIOA
EF46 03            710          DEFB 3
EF47 03            720          DEFB 3
EF48 03            730          DEFB 3
EF49               740 *
EF49 C0            750 LISTTAB DEFB 0C0H
EF4A 03            760         DEFB 3 *ROTATES
EF4B 01            770         DEFB 1 *LOCAL SERIAL PRINTER SIOB
EF4C 05            780         DEFB 5 *LOCAL PARALLEL PRINTER
EF4D 7F            790         DEFB 07FH *MASTER CRT PORT
EF4E 81            800         DEFB 81H *MASTER SERIAL PRINTER
                               PORT
EF4F               810 *
EF4F CD 01 EF      820 CONIN1 CALL SUBSTACK
EF52 21 31 EF      830        LD HL,CONINTAB
EF55 CD 1F EF      830        CALL PARSE
EF58 CD 06 F0      840 INP        CALL DOINPUT
EF5B DC 74 F0      840            CALL C,REPORT
```

```
EF5E C3 17 EF      84Ø                 JP GETSTACK
EF61               85Ø *
EF61 CD Ø1 EF      86Ø CONOUT1 CALL SUBSTACK
EF64 21 37 EF      86Ø          LD HL,CONOUTAB
EF67 CD 1F EF      86Ø          CALL PARSE
EF6A F5            87Ø OUTPUT PUSH AF
EF6B 79            87Ø        LD A,C
EF6C FE ~~Ø3~~ FF   ~~87Ø~~        CP 3 *PRINTREL CHAR
EF6E 28 ØB         88Ø        JR Z,RELP
EF7Ø F1            88Ø        POP AF
EF71 41            89Ø        LD B,C
EF72 CD Ø9 FØ      89Ø        CALL DOOUTPUT
EF75 DC 74 FØ      89Ø        CALL C,REPORT
EF78 C3 17 EF      89Ø        JP GETSTACK
EF7B F1            9ØØ RELP POP AF
EF7C CD 77 FØ      9ØØ      CALL DORELP
EF7F C3 17 EF      9ØØ      JP GETSTACK
EF82               91Ø *
EF82 CD Ø1 EF      92Ø LIST1 CALL SUBSTACK
EF85 21 49 EF      93Ø       LD HL,LISTTAB
EF88 CD 1F EF      93Ø       CALL PARSE
EF8B 18 DD         94Ø       JR OUTPUT
EF8D               95Ø *
EF8D CD Ø1 EF      96Ø READER1 CALL SUBSTACK
EF9Ø 21 3D EF      96Ø         LD HL,READTAB
EF93 CD 1F EF      96Ø         CALL PARSE
EF96 18 CØ         97Ø         JR INP
EF98               98Ø *
EF98 CD Ø1 EF      99Ø PUNCH1 CALL SUBSTACK
EF9B 21 43 EF      99Ø        LD HL,PUNCHTAB
EF9E CD 1F EF      99Ø        CALL PARSE
EFA1 18 C7         99Ø        JR OUTPUT
EFA3               1ØØØ *
EFA3 CD Ø1 EF      1Ø1Ø CONST1 CALL SUBSTACK
EFA6 21 31 EF      1Ø1Ø        LD HL,CONINTAB
EFA9 CD 1F EF      1Ø1Ø        CALL PARSE
EFAC CD ØC FØ      1Ø2Ø        CALL DOINSTAT
EFAF B7            1Ø3Ø CONST3    OR A
EFBØ 28 Ø5         1Ø3Ø           JR Z,CONST2
EFB2 3E ØØ         1Ø3Ø           LD A,Ø
EFB4 C3 17 EF      1Ø3Ø           JP GETSTACK
EFB7 3E FF         1Ø4Ø CONST2 LD A,-1
EFB9 C3 17 EF      1Ø4Ø        JP GETSTACK
EFBC CA 17 EF      1Ø5Ø        JP Z,GETSTACK
EFBF 3E FF         1Ø5Ø        LD A,-1
EFC1 C3 17 EF      1Ø5Ø        JP GETSTACK
EFC4               1Ø6Ø *
EFC4 CD Ø1 EF      1Ø7Ø LISTST1 CALL SUBSTACK
EFC7 21 49 EF      1Ø7Ø         LD HL,LISTTAB
EFCA CD 1F EF      1Ø7Ø         CALL PARSE
EFCD CD ØF FØ      1Ø8Ø         CALL DOOUTSTAT
EFDØ 18 DD         1Ø9Ø         JR CONST3
EFD2               11ØØ *
```

change from 03
to FF for
Microterm ACT 5A

SysteMaster I/O Listing

```
0000                10 *TKEY
0000                20 *6-22-82 ADD DUMIO MOD
0000                30 *FOR SYSTEMASTER TELETEK MICRODOZ SYSTEM
0000                40 *COPYRIGHT 4-15-1982 MICRO MIKE'S INC.
0000                50 *
0000                60 *MUST LF TKEY 100H LF DUMIO AT DUMIO THEN
                       SF TKEY 100
0000                70 *READS FILE TKEYF
0000                80 *TKEYF FORMAT=
0000                90 *DEFB HDISKFLG IF 1 THEN LOAD HDISK
0000                100 *DEFB QUITE IF 1 THEN NO OUTPUT
0000                110 *BAZIC STRING UP TO 127 CHARACTERS
0000                120 *
0000                130   ORG 100H
0100                140 MDOS EQU 5
0100                150 IOBYTE EQU 3
0100                160 LOGDISK EQU 4
0100                170 EXITMDOS EQU 0
0100 C3 B7 01       180          JP BEGIN
0103 0E 07          190 INPUT LD C,7
0105 C3 05 00       200       JP MDOS
0108 0E 08          210 OUTPUT LD C,8
010A C3 05 00       220        JP MDOS
010D 0E 09          230 PANIC LD C,9
010F C3 05 00       240       JP MDOS
0112 0E 0A          250 INSTAT LD C,10
0114 C3 05 00       260        JP MDOS
0117 0E 0B          270 OUTSTAT LD C,11
0119 C3 05 00       280         JP MDOS
011C 0E 0C          290 CLS LD C,12
011E C3 05 00       300     JP MDOS
0121 0E 0D          310 GOTOXY LD C,13
0123 C3 05 00       320        JP MDOS
0126 0E 0E          330 CRLF LD C,14
0128 C3 05 00       340      JP MDOS
012B 0E 0F          350 MSG LD C,15
012D C3 05 00       360     JP MDOS
0130 0E 10          370 HEX16 LD C,16
0132 C3 05 00       380       JP MDOS
0135 0E 11          390 HEX8 LD C,17
0137 C3 05 00       400      JP MDOS
013A 0E 12          410 DEC16 LD C,18
013C C3 05 00       420       JP MDOS
013F 0E 13          430 DEC8 LD C,19
0141 C3 05 00       440      JP MDOS
0144 0E 14          450 HEXSTR LD C,20
0146 C3 05 00       460        JP MDOS
0149 0E 15          470 DECSTR LD C,21
014B C3 05 00       480        JP MDOS
014E 0E 16          490 TAB LD C,22
0150 C3 05 00       500     JP MDOS
0153 0E 17          510 DOSCMD LD C,23
```

```
0155 C3 05 00      520          JP MDOS
0158 0E 18         530 SELDRV LD C,24
015A C3 05 00      540          JP MDOS
015D 0E 19         550 DCOM LD C,25
015F C3 05 00      560        JP MDOS
0162 0E 1A         570 DLOOK LD C,26
0164 C3 05 00      580         JP MDOS
0167 0E 1B         590 DWRIT LD C,27
0169 C3 05 00      600         JP MDOS
016C 0E 1C         610 SAVE LD C,28
016E C3 05 00      620        JP MDOS
0171 0E 1D         630 LOAD LD C,29
0173 C3 05 00      640        JP MDOS
0176 0E 1E         650 NEXTCMD LD C,30
0178 C3 05 00      660           JP MDOS
017B 0E 1F         670 MDOSSIZ LD C,31
017D C3 05 00      680           JP MDOS
0180 0E 20         690 PRINTREL LD C,32
0182 C3 05 00      700            JP MDOS
0185               710            DEFS 50
01B7               720 STACK DEFS 0
01B7               730 PATCH EQU 0C902H
01B7 31 B7 01      740 BEGIN LD SP,STACK
01BA 21 DF 02      750       LD HL,DUMIO
01BD 11 00 D7      750       LD DE,0D700H
01C0 01 00 02      750       LD BC,512
01C3 ED B0         750       LDIR
01C5 CD 06 D8      760       CALL 0D806H *TINT
01C8               770 *
01C8 21 94 02      780       LD HL,TKEY
01CB CD 62 01      790       CALL DLOOK
01CE DA 9C 02      800       JP C,NOTFND
01D1 11 08 00      810       LD DE,8
01D4 19            820       ADD HL,DE
01D5 CD DA 02      830       CALL DEISM *DA
01D8 EB            840       EX DE,HL
01D9 E5            850       PUSH HL
01DA 3E 81         850       LD A,81H
01DC CD 58 01      850       CALL SELDRV
01DF E1            860       POP HL
01E0 11 DF 04      870       LD DE,BUFFER
01E3 06 01         880       LD B,1
01E5 3E 01         890       LD A,1
01E7 CD 5D 01      900       CALL DCOM
01EA DA A5 02      910       JP C,HDERR
01ED 3A DF 04      920       LD A,(BUFFER)
01F0 B7            920       OR A
01F1 CA 3A 02      920       JP Z,BEG1
01F4               930 *MUST LF HDISK,1 E700
01F4 21 50 02      940       LD HL,LFHDISK
01F7 CD 62 01      950       CALL DLOOK
01FA DA 9C 02      950       JP C,NOTFND
01FD 11 08 00      960       LD DE,8
0200 19            960       ADD HL,DE
0201 CD DA 02      960       CALL DEISM
```

```
Ø2Ø4 D5            96Ø        PUSH DE *DA
Ø2Ø5 CD DA Ø2      97Ø        CALL DEISM
Ø2Ø8 D5            97Ø        PUSH DE *SIZE
Ø2Ø9 3E 81         98Ø        LD A,81H
Ø2ØB CD 58 Ø1      98Ø        CALL SELDRV
Ø2ØE D1            99Ø        POP DE
Ø2ØF 7B            99Ø        LD A,E
Ø21Ø Ø6 Ø1         99Ø        LD B,1
Ø212 E1            1ØØØ       POP HL *DA
Ø213 11 DF Ø6      1Ø1Ø       LD DE,BUFFER+2ØØH
Ø216 CD 5D Ø1      1Ø2Ø       CALL DCOM
Ø219 DA A5 Ø2      1Ø2Ø       JP C,HDERR
Ø21C 21 DF Ø6      1Ø3Ø       LD HL,BUFFER+2ØØH
Ø21F 11 ØØ E7      1Ø3Ø       LD DE,ØE7ØØH
Ø222 Ø1 FF 18      1Ø4Ø       LD BC,ØFFFFH-ØE7ØØH
Ø225 ED BØ         1Ø5Ø       LDIR
Ø227               1Ø6Ø *
Ø227 CD Ø2 C9      1Ø7Ø       CALL PATCH
Ø22A 3A EØ Ø4      1Ø8Ø       LD A,(BUFFER+1)
Ø22D B7            1Ø8Ø       OR A
Ø22E C2 3A Ø2      1Ø8Ø       JP NZ,BEG1
Ø231 CD 1C Ø1      1Ø9Ø       CALL CLS
Ø234 21 58 Ø2      11ØØ       LD HL,WMSG
Ø237 CD 2B Ø1      11ØØ       CALL MSG
Ø23A 21 E3 Ø4      111Ø BEG1 LD HL,BUFFER+4
Ø23D 11 8Ø ØØ      112Ø       LD DE,8ØH
Ø24Ø Ø1 7F ØØ      113Ø       LD BC,7FH
Ø243 ED BØ         114Ø       LDIR
Ø245 36 ØD         115Ø       LD (HL),ØDH
Ø247 21 8Ø ØØ      116Ø       LD HL,8ØH
Ø24A CD 53 Ø1      116Ø       CALL DOSCMD
Ø24D C3 ØØ ØØ      116Ø       JP Ø
Ø25Ø 48 44 49 53   117Ø LFHDISK "HDISK,1"
Ø257 ØD            118Ø          DEFB ØDH
Ø258 ØD ØA         119Ø WMSG DEFW ØAØDH
Ø25A 43 4F 5Ø 59   12ØØ       "COPYRIGHT 1982 MICRO MIKE'S INC."
Ø27A ØD ØA         121Ø       DEFW ØAØDH
Ø27C 57 41 49 54   122Ø       "WAITING FOR HARD DISK"
Ø291 ØD ØA         123Ø       DEFW ØAØDH
Ø293 ØØ            124Ø       DEFB Ø
Ø294 54 4B 45 59   125Ø TKEY "TKEYF,1"
Ø29B ØD            126Ø       DEFB ØDH
Ø29C 21 AE Ø2      127Ø NOTFND LD HL,NF
Ø29F CD 2B Ø1      127Ø         CALL MSG
Ø2A2 C3 ØØ ØØ      127Ø         JP Ø
Ø2A5 21 CB Ø2      128Ø HDERR LD HL,HDM
Ø2A8 CD 2B Ø1      128Ø        CALL MSG
Ø2AB C3 ØØ ØØ      128Ø        JP Ø
Ø2AE ØD ØA         129Ø NF DEFW ØAØDH
Ø2BØ 54 4B 45 59   13ØØ    "TKEYF OR HDISK NOT FOUND"
Ø2C8 ØD ØA         131Ø    DEFW ØAØDH
Ø2CA ØØ            132Ø    DEFB Ø
Ø2CB ØD ØA         133Ø HDM DEFW ØAØDH
Ø2CD 44 49 53 4B   134Ø     "DISK ERROR"
Ø2D7 ØD ØA         135Ø     DEFW ØAØDH
```

```
Ø2D9 ØØ          136Ø     DEFB Ø
Ø2DA 5E          137Ø DEISM LD E,(HL)
Ø2DB 23          137Ø       INC HL
Ø2DC 56          137Ø       LD D,(HL)
Ø2DD 23          137Ø       INC HL
Ø2DE C9          138Ø       RET
Ø2DF             139Ø DUMIO DEFS 512
Ø4DF             14ØØ BUFFER DEFS Ø



ØØØØ             1Ø *DUMIO
ØØØØ             2Ø *FOR SYSTEMASTER MICRODOZ
ØØØØ             3Ø SAINP EQU ØDCBBH
ØØØØ             4Ø XPOUT EQU ØDCEØH
ØØØØ             5Ø ASOUT EQU ØDCC1H
ØØØØ             6Ø BSOUT EQU ØDCC4H
ØØØØ             7Ø ASTATA EQU ØDCB5H
ØØØØ             8Ø XPSTAT EQU ØDCE3H
ØØØØ             9Ø *
ØØØØ             1ØØ  ORG ØD7ØØH
D7ØØ 18          11Ø PAGE DEFB 24 *LINES PER CRT PAGE
D7Ø1 Ø1          12Ø VERIFY DEFB 1 *READ AFTER WRITE IF 1
D7Ø2 ØØ          13Ø CONFIG DEFB Ø *CONFIGURATION BYTE FOR QUAD
                     5 INCH DRIVE
D7Ø3 ØØ          14Ø DSYS DEFB Ø *IF NON ZERO DON'T LIST SYSTEM
                     FILES
D7Ø4             15Ø *CSCR AND GOTO 1ST BYTE IS STRING LENGTH
D7Ø4             16Ø *IF 1ST BYTE = ØFFH THEN MACHINE CODE MUST
                     FOLLOW
D7Ø4 Ø2          17Ø CSCR DEFB 2 *CLEAR SCREEN
D7Ø5 1B          18Ø      DEFB 27
D7Ø6 45          19Ø      DEFB "E"
D7Ø7 ØØ          2ØØ      DEFB Ø
D7Ø8             21Ø      ORG CSCR+16
D714 Ø2          22Ø GOTO DEFB 2 *POSITION CURSOR
D715 1B          23Ø      DEFB 27
D716 59          24Ø      DEFB "Y"
D717 ØØ          25Ø      DEFB Ø
D718             26Ø      ORG GOTO+16
D724             27Ø *XOFF AND YOFF OFFSET ADDED TO VALUE TO
                     POSITION CURSOR
D724 1F          28Ø XOFF DEFB 31
D725 1F          29Ø YOFF DEFB 31
D726             3ØØ *TKEY TURNKEY THE DOS HERE
D726             31Ø *1ST BYTE IS LENGTH OF STRING
D726             32Ø *MUST END WITH ØDH (CARRIAGE RETURN)
D726 ØØ          33Ø TKEY DEFB Ø
Ø727 ØD          34Ø      DEFB ØDH
D728             35Ø      DEFS 79
D777 ØD          36Ø      DEFB ØDH
D778             37Ø *BKSP THE STRING USED TO BACKSPACE ENDS
                     WITH Ø
D778 Ø8          38Ø BKSP DEFB 8
D779 2Ø          39Ø      DEFB 32
```

```
D77A Ø8            400      DEFB 8
D77B ØØ            410      DEFB Ø
D77C               420      DEFS 4
D78Ø               430 *BKSPSET PARSE KEYBOARD CHARACTERS USED TO
                       BACKSPACE
D78Ø FE Ø8         440 BKSPSET CP 8
D782 C8            450         RET Z
D783 FE 5F         460         CP 95
D785 C8            470         RET Z
D786 FE 11         480         CP 11H
D788 C8            490         RET Z
D789 FE 7F         500         CP 127
D78B C9            510         RET
D78C               520 *
D78C               530         ORG ØD8ØØH
D8ØØ 18 ØC         540 INP JR INPUT1
D8Ø2 18 ØD         550 OUT JR OUTPUT
D8Ø4 18 26         560 PANIC JR PANICS
D8Ø6 18 62         570 TINT JR TINTO
D8Ø8 18 53         580 INSTAT JR INSTATS
D8ØA 18 2F         590 OUTSTAT JR OUTSTATS
D8ØC ØØ            600 PRINTREL DEFB Ø *USED IN TIMESHARE SYSTEM
D8ØD C9            610          RET
D8ØE               620 *NOTE MIXUP OF SIOA AND SIOB
D8ØE C3 BB DC      630 INPUT1 JP SAINP
D811 FE Ø1         640 OUTPUT CP 1
D813 28 14         650        JR Z,PRINTER
D815 FE Ø2         660        CP 2
D817 F5            670        PUSH AF
D818 CC 29 D8      680        CALL Z,PRINTER
D81B F1            690        POP AF
D81C FE Ø3         690        CP 3
D81E CA EØ DC      690        JP Z,XPOUT
D821 FE Ø4         700        CP 4
D823 CC EØ DC      700        CALL Z,XPOUT
D826 C3 C1 DC      710 CRT JP ASOUT
D829 C3 C4 DC      720 PRINTER JP BSOUT
D82C CD B5 DC      730 PANICS CALL ASTATA
D82F 3C            730        INC A
D83Ø B7            730        OR A
D831 CØ            730        RET NZ
D832 CD BB DC      740        CALL SAINP
D835 E6 7F         740        AND 7FH
D837 FE Ø3         740        CP 3
D839 37            740        SCF
D83A C9            740        RET
D83B FE Ø1         750 OUTSTATS CP 1
D83D 28 13         760          JR Z,OUTSTATS1
D83F FE Ø3         770          CP 3
D841 28 17         770          JR Z,OUTSTATS3
D843 B7            780          OR A
D844 28 Ø4         790          JR Z,OUTSTATSØ
D846 3E FF         800          LD A,-1
D848 37            810          SCF
D849 C9            820          RET
```

```
D84A AF           83Ø OUTSTATSØ XOR A
D84B D3 Ø1        83Ø           OUT (1),A
D84D DB Ø1        83Ø           IN A,(1)
D84F E6 Ø4        83Ø           AND 4
D851 C9           83Ø           RET
D852 AF           84Ø OUTSTATS1 XOR A
D853 D3 Ø3        84Ø           OUT (3),A
D855 DB Ø3        84Ø           IN A,(3)
D857 E6 Ø4        84Ø           AND 4
D859 C9           84Ø           RET
D85A C3 E3 DC     85Ø OUTSTATS3 JP XPSTAT
D85D B7           86Ø INSTATS OR A
D85E 28 Ø4        87Ø         JR Z,INSTATSØ
D86Ø 3E FF        88Ø         LD A,-1
D862 37           89Ø         SCF
D863 C9           9ØØ         RET
D864 CD B5 DC     91Ø INSTATSØ CALL ASTATA
D867 3C           91Ø          INC A
D868 B7           91Ø          OR A
D869 C9           91Ø          RET
D86A              92Ø *
D86A 21 78 D8     93Ø TINTO LD HL,PIOTB
D86D 4E           94Ø TELINIT1 LD C,(HL)
D86E 23           94Ø          INC HL
D86F 79           94Ø          LD A,C
D87Ø 3C           94Ø          INC A
D871 C8           94Ø          RET Z
D872 46           95Ø          LD B,(HL)
D873 23           95Ø          INC HL
D874 ED B3        95Ø          OTIR
D876 18 F5        95Ø          JR TELINIT1
D878 Ø5           96Ø PIOTB DEFB 5 *PORT
D879 Ø3           97Ø       DEFB 3 *NUM BYTES
D87A FC           98Ø       DEFB ØFCH *PIOA IVECTOR
D87B ØF           99Ø       DEFB ØFH *OUTPUT
D87C 87           1ØØØ      DEFB 87H *EN INTS
D87D              1Ø1Ø *PIOB
D87D Ø7           1Ø2Ø       DEFB 7 *PORT
D87E Ø4           1Ø3Ø       DEFB 4 *NUMBER
D87F FA           1Ø4Ø       DEFB ØFAH *IVECT
D88Ø FF           1Ø5Ø       DEFB ØFFH *MODE3
D881 88           1Ø6Ø       DEFB 88H *CONTROL WORD
D882 Ø7           1Ø7Ø       DEFB 7 *NO INTS
D883              1Ø8Ø *CTC Ø..3
D883              1Ø9Ø CT EQU 47H
D883              11ØØ TM EQU 7
D883 Ø8           111Ø    DEFB 8 *PORT
D884 Ø3           112Ø    DEFB 3 *NUMBER
D885 FØ           113Ø    DEFB ØFØH *IVECTOR
D886 47           114Ø    DEFB CT
D887 Ø8           115Ø    DEFB 8 *96ØØ BAUD
D888 Ø9           116Ø    DEFB 9 *PORT
D889 Ø2           117Ø    DEFB 2 *NUM
D88A 47           118Ø    DEFB CT
D88B Ø8           119Ø    DEFB 8 *96ØØ BAUD
```