NCR

NCR DECISION MATE V

System Technical Manual -MS™-DOS

MS-DOS is a trademark of Microsoft Inc.
Intel is a registered trademark of Intel Corporation.
Z-80 is a registered trademark of Zilog, Inc.
ADM-3A and ADM-31 are trademarks of Lear Siegler.

Second Edition, October 1984

It is the policy of NCR Coporation to improve products as new technology, components, software, and firmware become available. NCR Corporation, therefore, reserves the right to change specifications without prior notice.

All features, functions, and operations described herein may not be marketed by NCR in all parts of the world. In some incstances, photographs are of equipment prototypes. Therefore, before using this document, consult your nearest dealer of NCR office for information that is applicable and current.

FOREWORD

The NCR DECISION MATE V System Technical Manuals are designed to provide both hardware and software information: they are intended for designers, system integrators, programmers, and other interested persons who require detailed information on the construction and operation of the NCR DECISION MATE V.

Problems arising from any changes that you make to the hardware or software of the NCR DECISION MATE V are your responsibility. NCR cannot assist in resolving problems that may arise when making changes to the hardware or software.

The first manual provides general information on the NCR DECISION MATE V and its various options. Information is included on how to identify the various models and kits that are available. The hardware description includes information about the I/O bus, signal levels, power requirements, and plug/pin assignments.

The other manuals provide information on the various operating system software used with the NCR DECISION MATE V. The software descriptions include information for using system routines at machine code level.

The appendices provide schematics, component locations, software listings, and other information that may be helpful to the user of these manuals.

NCR DECISION MATE V
SYSTEM TECHNICAL MANUALS

System Technical Manual
Hardware

System Technical Manual
CP/M®-80

System Technical Manual
MS™ -DOS

System Technical Manual
CP/M®-86

System Technical Manual
p-System™

In the NCR DECISION MATE V System Technical Manual series, the chapters are arranged in numeric sequence and the appendices in alphabetic sequence:

Hardware — Chapters 1 and 2, Appendix A

CP/M-80 — Chapter 3, Appendix B

MS-DOS — Chapter 4, Appendix C

CP/M-86 — Chapter 5, Appendix D

MS-DOS SOFTWARE FOR INPUT/OUTPUT

CONTENTS

APPENDIX C

MS-DOS SOFTWARE FOR INPUT/OUTPUT

MS-DOS software is an operating system that loads from flexible disk into read/write memory. A set of fréquently used utilities reside in memory while others are loaded from disk as required.

I/O functions can be executed by means of system calls. These "calls" are, in fact, executed by using the 8086 software interrupt. Interrupt 21H is of particular interest as it provides access to a large number of Function Requests, simply by placing the number of the requested function in the register AH. Registers not used to return information are correctly restored upon return from the function call. The only exception is the register AX.

The MS-DOS user who wishes to make use of MS-DOS internal and external commands only, should refer primarily to the NCR MS-DOS User's Guide. The I/O examples in this chapter will be of interest mainly to machine code programmers. A full explanation of MS-DOS use of software interrupts is to be found in the NCR MS-DOS Programmer's Manual. Machine code programmers are recommended to refer to this manual, where you will also find information regarding file and disk management.

LOGICAL DISK LAYOUT

FLEXIBLE DISK (5 1/4-inch)

The drive for flexible disk is designed to make use of double-sided disks with double-density storage of data on 48 or 96 tracks per inch (TPI), i.e. 80 or 160 tracks per disk. The two surfaces are designated surface 0 and surface 1. Each track is divided into 5, 8 or 9 equal sectors. Each sector is further divided into an address area and a data area.

The following is a description of the logical layout and formatting requirements for flexible disks being used by MS-DOS. Figure 4.1 presents the corresponding schematic layout. Certain elements of formatting on the flexible disk are fixed and in-variable. This applies in particular to the address area (surface number, track number, etc.). However, the flexible disk has not been initialized at manufacture with this information. It is the user's responsibility to include this information in the initialization process. If you wish, the format utility will do this for you.

NOTE: With regard to hexadecimal values in the following description, the most significant bit (Bit 7) in each byte is recorded first.

Gap 4

This presents a filler immediately prior to the physical index hole. This gap is filled with bytes of hexadecimal 4E. The number of these bytes can vary, but a typical number is 873 for flexible disks formatted with 8 sectors per track, and 512 bytes per sector (48 TPI disk with 320 KB).

Gap 1

Immediately following the index hole: 80 bytes of 4E, then 12 bytes of zero, then 3 bytes of hexadecimal C2, then FC, then 50 bytes of 4E. This gap and Gap 4 serve to compensate for timing variations due mainly to rotational speed.

Sync Field

12 bytes of zero to resynchronize the PLO (phase locked oscillator) after encountering timing discrepancies resulting from in-place updates or re-initialization.

AM (Address Marker)

3 bytes of hexadecimal A1 followed by FE. The A1 bytes have a missing clock transition between bits 2 and 3. (Both these bits and the bit immediately above and below these bits are reset, i.e. value 0.) AM indicates that address information follows.

DM (Data Marker)

As with AM, except that FB follows the A1 bytes. DM indicates that data follows.

CM (Control Marker)

3 bytes of hexadecimal C2 followed by FC. The C2 bytes have a missing clock transition between bits 3 and 4. (Both these bits and the bit immediately above and below these bits are reset, i.e. value 0.) CM indicates that control information follows.

ID (Address) Field

The 4 bytes following the address marker (AM) must contain the following information:

Byte 1 Track (cylinder) number zero through 27H.
Byte 2 Surface (head) number: 00 = surface 0, 01=surface 1.
Byte 3 Sector number 01 through 05, 08 or 09.
Byte 4 Physical record length: 02 indicates 512 bytes per sector, 03 indicates 1024 bytes per sector

Data

The 512/1024 bytes following the data marker (DM) are available for data storage.

CRC (Cyclic Redundancy Check)

Polynomial codes are recorded in 2 bytes at the end of each address or data area for error checking purposes.

In the case of an address area, the CRC value is computed using the preceding 8 characters (i.e. A1, A1, A1, FE, and the 4 address bytes).

For a data area, the preceding 516/1028 bytes are used (i.e. A1, A1, A1, FB, and the 512/1024 data bytes).

Gap 2

22 bytes of hexadecimal 4E immediately following the address CRC.

Gap 3

80 bytes of hexadecimal 4E immediately following the data CRC.

The format utility on your NCR MS-DOS flexible disk functions in accordance with the following default parameters for 48/96 TPI disks, resp. Four other formats are available (see User's Guide and BIOS Parameter Block BPB in program LOADER/ FDBOOT in Appendix C).

Number of heads — 2
Number of sectors/track — 9/5
No hidden sectors.
Bytes per sector — 512/1024
Sectors per allocation unit — 2
Reserved sectors — 1
File allocation tables — 2
Root directory entries — 112/160

The first sector of the flexible disk formatted by the MS-DOS format utility contains a boot loader preceded by a BIOS parameter block (BPB). You can find details of the MS-DOS BPB and of how to set one up in the LOADER/FDBOOT module of the I/O software (see Appendix C) and the MS-DOS Programmer's Manual respectively.

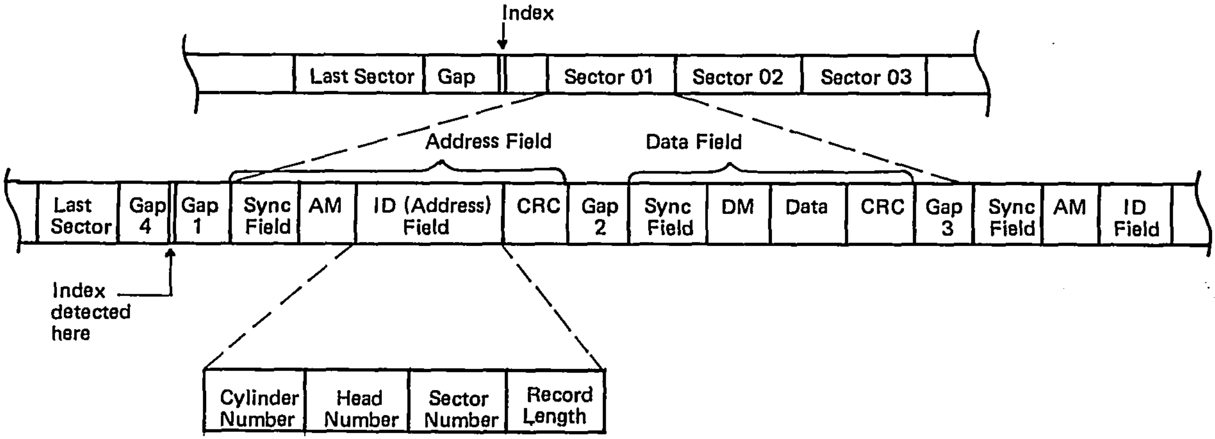


Figure 4.1

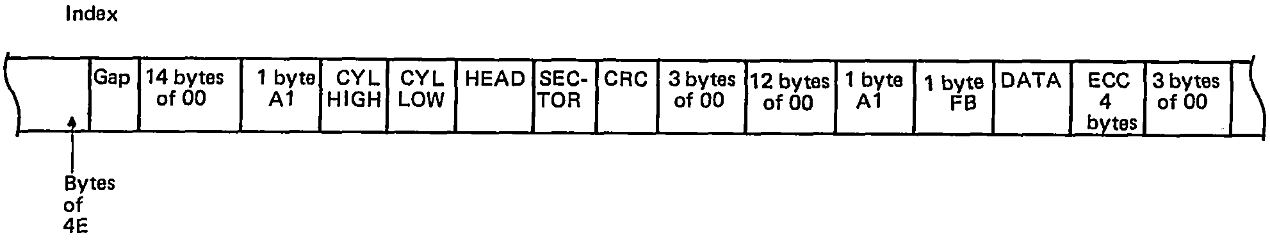


Figure 4.2

WINCHESTER DISK

The Winchester disk software format is similar to that of the flexible drive in that an index mark is recognized (a pulse of at least 200nS) followed by ID and Data Fields, including check bytes. Figure 4.2 shows this layout.

Gap

30 bytes of 4E for a sector length of 512 bytes.

CYL HIGH

Value FF: cylinders 256 to 511

Value FE: cylinders 0 to 255

Value FD: cylinders 768 to 1023

Value FC: cylinders 512 to 767

HEAD

Bit 7 indicates a bad block

Bytes of 4E

A typical number of these bytes is 304 at 3600 r.p.m.

The capacity of the Winchester disk is 10Mbytes in 1224 tracks. Track density is 312 tracks per inch. The Winchester disk may optionally be regarded as two logical units, each with a capacity of 5Mbytes and 305 cylinders per logical unit (cylinder 306 is reserved for diagnostic purposes or as one logical unit with 10MB capacity and 610 cylinders (2 cylinders are reserved). MS-DOS uses the Winchester disk as follows:

Number of sectors/track — 17

Bytes per sector — 512

Sectors per allocation unit — 16 (8 if 10 MB disk)

Number of reserved sectors —1 (0 if non-bootable 5 MB disk)

File allocation tables — 1

Root directory entries -- 512 (496 if bootable 5 MB disk)

DEVICE DRIVERS

A special feature of MS-DOS is that it pe.mits the programmer to write specialized device drivers which can be activated via MS-DOS Function Request interrupts. It is possible to set up character device drivers for console, printer, and user-defined devices. Additionally, block device drivers for disk I/O can be created.

A Device Header assigns the logical I/O function, determines whether control strings can be handled, and contains pointers to its interrupt and strategy routines, as well as to other device headers. User-defined device drivers can be used in place of the standard I/O devices. The device driver can be installed anywhere in memory by MS-DOS.

Calling a device driver means that a Request Header at ES :BX is passed to the strategy entry point. The Request Header contains information regarding the exact type of I/O function to be performed. It is the user's repsonsibility to save any register contents he requires on the stack. If more than 20 pushes are needed, the driver should set up its own stack.

The user should refer to the NCR MS-DOS Programmer's Manual for details of Device and Request Headers.

The following description relates to the means — already provided by MS-DOS — of communication with devices. This I/O control for devices is the Function Request, activated by loading register AH with 44H and then issuing interrupt type 21H.

Function 44H sets or gets device information associated with an open handle, or sends/receives a control string to a device handle or device. The entry and return parameters are as follows:

Entry
  BX — Handle
  BL — Drive (for calls AL = 4 or 5: 0 = default, 1 = A, etc.)
  DS :DX — Data or buffer
  CX — Bytes to read or write
  AL — Function code; see text
Return
  Carry set:
  AX — 6 = invalid handle, 1 = invalid function, 13H = invalid data, 5 = access denied
  Carry not set:
  AL = 2, 3, 4, or 5:
  AX = Count transferred
  AL = 6 or 7: 00 = Not ready, FF = Ready

The following values are allowed as function code:

0 Get device information (returned in DX)
1 Set device information (as determined by DX)
2 Read CX number of bytes into DS :DX from device control channel
3 Write CX number of bytes from DS :DX to device control channel
4 Same as 2, only drive number in BL 0 = default, A: = 1, B: = 2, . . .

5 Same as 3, only drive number in BL 0 = default, A: = 1, B: = 2, . . .
6 Get input status
7 Get output status

This function can be used to get information about device channels. Calls can be made to regular files, but only calls 0, 6, and 7 are defined in that case (AL = 0, 6, or 7). All other entry values return an invalid function error.

Entry values AL = 0 and AL = 1
The bits of DX for entry values are defined as in Figure 4.3. Note that the upper byte must be zero on a set call (AL = 1).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	CTRL	Reserved						ISDEV	EOF	RAW	SPECL	ISCLK	ISNUL	ISCOT	ISCIN

Figure 4.3

ISDEV = 1 if this channel is a device, or 0 if this channel is a disk file (Bits 8-15 = 0 in this case)
If ISDEV = 1
EOF = 0 if End Of File on input
RAW = 1 if this device is in Raw mode
ISCLK = 1 if this device is the clock device
ISNUL = 1 if this device is the null device
ISCOT = 1 if this device is the console output
ISCIN = 1 if this device is the console input
SPECL = 1 if this device is special
CTRL = 0 if this device can not do control strings via calls AL = 2 and AL = 3.
CTRL = 1 if this device can process control strings via calls AL = 2 and AL = 3.
NOTE that bit 15 cannot be set.

If ISDEV = 0
EOF = 0 if channel has been written
Bits 0-5 are the block device number for the channel (0 =A:, 1 = B:, . . .)
Bits 15, 8-13, 4 are reserved and should not be altered.

Entry values AL = 2, 3, 4, or 5:

These four entry values allow arbitrary control strings to be sent or received from a device. The entry syntax is the same as for read and write, except for 4 and 5, which take a drive number in BL instead of a handle in BX.

An invalid function error is returned if the CTRL bit is 0.

An access denied is returned by calls AL = 4 or 5 if the drive number is invalid.

Entry values 6 or 7

These two values allow the user to check whether a file handle is ready for input or output. Status of handles open to a device is the intended use of these entries, but status of a handle open to a disk file is allowed, and is defined as follows:

Input:

Always ready (AL = FF) until EOF reached, then always not ready (AL = 0) unless current position changed.

Output:

Always ready (even if disk full).

NOTE: The status is defined at the time the function is activated. On future versions, by the time control is returned to the user from the system, the status returned may not correctly reflect the true current state of the device or file.

Error returns:

AX

6 = invalid handle — the handle passed in BX was not currently open.

1 = invalid function — the function passed in AL was not in the range 0 . . . 7.

13H = invalid data

5 = access denied (AL = 4, 5, 6, or 7 at entry)

Your NCR MS-DOS Programmer's Manual contains sample block and character device driver programs.

8086 INTERRUPT HANDLING

In addition to the hardware interrupt facility of the microprocessor, the 8086 provides additional capability of software interrupt setting and handling. A one or two byte interrupt instruction, INT or INT n, where n is the interrupt identification number (not a hexadecimal address in the interrupt vector), forces execution of the following:

- PUSH Flags
- Clear Interrupt Flag
- PUSH CS and Instruction Pointer
- CALL interrupt service routine.

Use of the IRET instruction subsequently ensures that Instruction Pointer, CS, and Flags are correctly POPped. The 8086 Flags Register contains a Trap Flag. If set, this flag will force an interrupt upon completion of an instruction cycle.

Figure 4.4 shows reserved areas in lower memory for interrupt handling. Interrupts type 0 through 31 (0 . . .19H) are either used by the CPU directly or are to be considered as reserved for similar purposes. Of the remaining interrupt types 32 through 255 (20H . . . 0FFH), types 32 through 39 (20H . . . 27H) are used by MS-DOS, and those between 40 and 64 (28H . . . 40H) should be considered as reserved for MS-DOS.

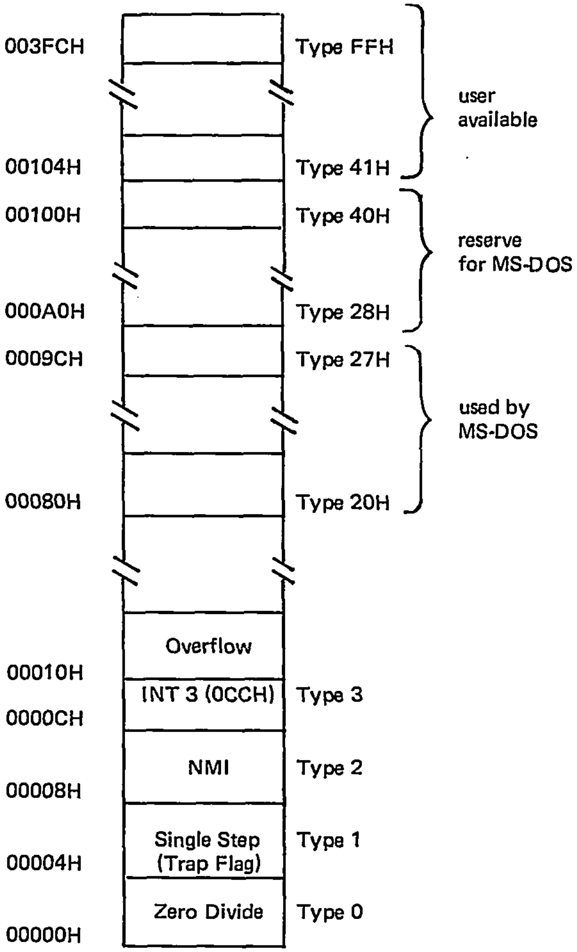


Figure 4.4 Interrupt vector

MS-DOS MEMORY MAP

With the exception of the transient part of COMMAND.COM and the user stack set up by MS-DOS, the MS-DOS I/O software and subsequently loaded files are situated towards the lower end of memory, as shown in Figure 4.5.

It must be appreciated that beyond specifying the end of the interrupt vector, absolute machine addresses for the various parts of MS-DOS cannot be given. This is because of the different hardware characteristics with which MS-DOS must interface.

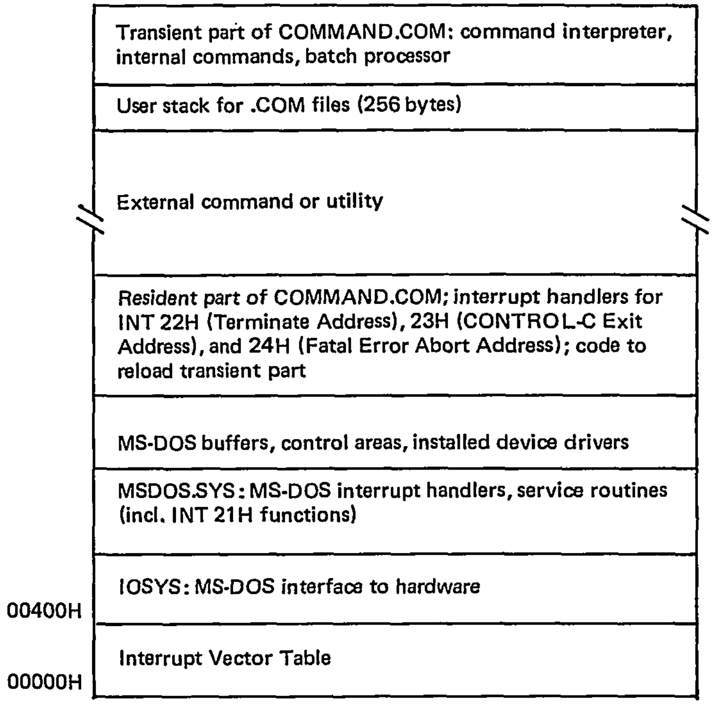

Transient part of COMMAND.COM: command interpreter, internal commands, batch processor
User stack for .COM files (256 bytes)
External command or utility
Resident part of COMMAND.COM; interrupt handlers for INT 22H (Terminate Address), 23H (CONTROL-C Exit Address), and 24H (Fatal Error Abort Address); code to reload transient part
MS-DOS buffers, control areas, installed device drivers
MSDOS.SYS: MS-DOS interrupt handlers, service routines (incl. INT 21H functions)
IOSYS: MS-DOS interface to hardware
00400H
Interrupt Vector Table
00000H

Figure 4.5 MS-DOS memory map

Top of memory is, of course, dependent on the physical size of RAM in your own NCR DECISION MATE V. If you are using 64Kbytes, MS-DOS makes approximately 34Kbytes available for MS-DOS external utilities or user programs.

MS-DOS INTERRUPTS

MS-DOS makes use of the following software interrupts. Of particular interest is INT 21H, which is the Function Request interrupt (see "Function Requests").

Before attempting to use these interrupts, the programmer should refer to the NCR MS-DOS Programmer's Manual.

Interrupt (Hex)	Description
20	Program Terminate
21	Function Request
22	Terminate Address
23	CONTROL-C Exit Address
24	Fatal Error Abort Address
25	Absolute Disk Read
26	Absolute Disk Write
27	Terminate But Stay Resident

Figure 4.6 MS-DOS interrupts

FUNCTION REQUESTS

MS-DOS offers the programmer a number of I/O functions via a common entry point in the interrupt vector. This is by far the most expedient way of activating I/O functions, as this entry point remains the same, irrespective of the hardware characteristics being interfaced by MS-DOS. The appropriate hexadecimal function number is placed in register AH. In addition, some functions require additional entry parameters in other registers. The function is activated by the two byte instruction INT 21H. Certain functions yield a return value or set flags.

A summary of the Function Requests is given in Figure 4.7. For full descriptions, especially with regard to file and directory handling functions, you should refer to the NCR MS-DOS Programmer's Manual.

Function (Hex)	Description	Entry/Return Parameters
00	Terminate program	
01	Read keyboard, echo to display	Return: next character pressed in AL
02	Display character	Entry: character in DL
03	Input from auxiliary device	Return: character in AL
04	Output to auxiliary device	Entry: character in DL
05	Send character to printer device	Entry: character in DL
06	Direct console I/O (CONTROL-C not recognized)	Entry: if DL < > 0FFH then character is displayed, otherwise... Return: character in AL and zero flag reset. If no character ready then zero flag set and AL = 0.
07	Read keyboard, no echo (CONTROL-C not recognized)	Return: next key pressed in AL
08	As 07, except CONTROL-C recognized	As 07
09	Display string until $ encountered ($ not displayed)	Entry: DS:DX segment: offset of beginning of string.
0A	Read keyboard input to memory buffer	Entry: DS:DX beginning of buffer; DS: [DX] max. no. of characters in buffer Return: DS: [DX+1] actual no. of characters in buffer
0B	Check keyboard status	Return: if AL = 0FFH, then character is ready, otherwise AL = 0
0C	Clear queue in type-ahead buffer	Entry: Function 1, 6, 7, 8, or 0AH in AL if subsequently required Return: 0 in AL indicates buffer flushed
0D	Disk reset, including writing of buffers recognized as modified	
0E	Select default disk	Entry: drive no. in DL (0 = A etc) Return: no. of logical drives available
0F	Open file	Entry: DS:DX address of FCB Return: AL = 0: directory entry found; otherwise 0FFH
10	Close file	As 0F
11	Search for first entry	Entry: DS:DX address of FCB Return: AL = 0: directory entry found; otherwise 0FFH

Figure 4.7 (1 of 6)

Function (Hex)	Description	Entry/Return Parameters
12	Search for next entry, using FCB specified in Function 11	Entry: DS:DX address of FCB Return: AL = 0: further matching directory entry found, otherwise 0FFH
13	Delete all matching directory entries (wildcard ? acceptable)	Entry: DS:DX address of FCB Return: AL = 0: at least one matching directory found, otherwise 0FFH
14	Sequential read according to current record at offset 20H, current block at offset 0CH, record size at offset 0EH, all offsets to FCB beginning	Entry: DS:DX address of FCB Return: AL = 0: normal read; AL = 1: EOF, no read; AL = 3: EOF, partial read; AL = 2: not enough room at Disk Transfer Address
15	Sequential write, FCB details as in Function 14	Entry: DS:DX address of FCB Return: AL = 0: normal write; AL = 1: disk full, no write; AL = 2: not enough room at Disk Transfer Address, no write
16	Create file, deleting matching directory entry, if any	Entry: DS:DX address of FCB Return: AL = 0: new entry; AL = 0FFH: no entry space, no deletion
17	Rename file to filename2 at offset 11H to FCB beginning (wildcard ? acceptable)	Entry: DS:DX address of FCB Return: AL = 0: OK. AL = 0FFH: no matching entry for filename1
19	Current disk drive	Return: current disk drive in AL (Ω = drive A etc)
1A	Set Disk Transfer Address to override default of 80H in Program Segment Prefix	Entry: Disk Transfer Address in DS:DX
21	Random read. Current block (offset 0CH to FCB beginning) and current record (offset 20H) are set to the Relative Record (offset 21H). Record thus addressed is loaded to DTA	Entry: DS:DX address of FCB Return: see Function 14 "Sequential Read"
22	Random write. FCB details as in Function 21, then write from DTA. Records buffered until record size (512 bytes) attained	Entry: DS:DX address of FCB Return: see Function 15 "Sequential Write"
23	File size in records returned to offset 21H to FCB beginning	Entry: DS:DX address of FCB Return: AL = 0: directory entry found, otherwise 0FFH
24	Set relative record at offset 21H to FCB beginning to same file address as at current block (offset 0CH) and current record (offset 20H)	Entry: DS:DX address of FCB

Figure 4.7 (2 of 6)

Function (Hex)	Description	Entry/Return Parameters
25	The contents of DS:DX replaces the 4 byte CS: IP value for the interrupt type no. contained in register AL	
27	Random block read. Starting at the record specified by relative record (offset 21H to beginning of FCB), CX no. of blocks are read to the Disk Transfer Address. Current block/record and Relative Record then set to address next record	Entry: DS:DX address of FCB; CX number of blocks to be read Return: see Function 21; also, CX = no. of blocks actually read
28	Random block write. Details as in Function 27, then write from Disk Transfer Address. Blocks allocated only as required	Entry: DS:DX address of FCB; CX number of blocks to be written Return: see Function 22; also CX = no. of blocks actually written
29	Parse command string for legal drive: filename.ext	Entry: DS:SI string to parse; ES:DI address for FCB; AL: parameters for filling in FCB Return: AL = 0FFH: drive letter invalid; AL = 1: wildcards used: AL = 0: no wildcards used
2A	Get date as binary numbers	Return: year in CX; month in DH; day in DL; day of week in AL
2B	Set date in binary numbers	Entry: year in CX; month in DH; day in DL Return: AL = 0: date valid, otherwise AL = 0FFH
2C	Get time as binary numbers	Return: hour in CH; minutes in CL; seconds in DH; 1/100 second in DL
2D	Set time in binary numbers	Entry: hour in CH; minutes in CL; seconds in DH; 1/100 second in DL Return: AL = 0: time valid, otherwise AL = 0FFH
2E	Set or reset Verify Flag checked after each disk write	Entry: AL = 0: no verify: AL = 1: verify
2F	Get Disk Transfer Address	Return: Disk Transfer Address in ES:BX
30	Get MS-DOS version number	Return: major version in AL; minor version in AH
31	Terminate but keep current process	Entry: exit code in AL; no. of paragraphs for initial allocation block in DX

Figure 4.7 (3 of 6)

Function (Hex)	Description	Entry/Return Parameters
33	Check for CONTROL-C during any function	Entry: if AL = 0 then current state will be returned; if AL = 1 then DL must be 1 (check on) or 0 (check off) Return: DL = 1: check is on; DL = 0: check is off
35	Get address of interrupt service routine	Entry: interrupt number in AL Return: ES:BX points to service routine
36	Get free disk space	Entry: drive in DL (0 = default, 1 = A, etc.) Return: no. of allocation units on drive in DX; sectors per allocation unit in AX or 0FFH if invalid drive; bytes per sector in CX; no. of free allocation units in BX
38	Get country-dependent information	Entry: country code in AL; address where information is to be written in DS:DX Return: AX = 2 and Carry Flag set: country not found
39	Create sub-directory	Entry: pointer to pathname in DS:DX Return: carry reset: OK; carry set and AX = 3 or 5: error
3A	Remove a directory entry	Entry: pointer to pathname in DS:DX Return: carry reset: OK; carry set and AX = 3 or 5 or 16: error
3B	Change the current directory	Entry: pointer to pathname in DS:DX Return: carry reset: OK; carry set and AX = 3: error
3C	Create a file. If file exists, then it is truncated to zero length	Entry: pointer to pathname in DS:DX; attribute in CX Return: carry reset: handle number in AX; carry set and AX = 3,4, or 5: error)
3D	Open a file	Entry: DS:DX points to name of file to be opened; AL contains 0 (r/o) or 1 (w/o) or 2 (r/w) Return: carry reset: handle number in AX; carry set and AX = 2, 4, 5 or 12: error
3E	Close a File Handle and flush internal buffers	Entry: handle number in BX Return: carry reset: OK; carry set and AX = 6: error

Figure 4.7 (4 of 6)

Function (Hex)	Description	Entry/Return Parameters
3F	Read from a file or device	Entry: DS:DX points to buffer; no. of bytes to read in CX; file handle no. in BX Return: carry reset: no. of bytes read in AX; carry set and AX = 5 or 6: error
40	Write to a file or device	Analogous to Function 3F.
41	Delete a directory entry	Entry: pointer to pathname in DS:DX Return: carry reset: OK; carry set and AX = 2 or 5: error
42	Move file pointer according to one of three methods: 0: offset relates to beginning of file; 1: offset relates to current position; 2: offset relates to end	Entry: method in AL; CX:DX distance to move; BX handle number Return: carry reset: OK; new pointer location in DX:AX; carry set and AX = 1 or 6: error
43	Change or return attributes of a file	Entry: pointer to pathname in DS:DX; if AL = 1 then new attributes in CX, otherwise AL must be zero for return Return: carry reset: attributes in CX; carry set and AX = 1, 3 or 5: error
44	Set or get device information relating to an open handle; send or receive control string to a device handle or device. See "Device Drivers"	See "Device Drivers"
45	Duplicate a file handle	Entry: file handle in BX Return: carry reset: new file handle in AX; carry set and AX = 4 or 6: error
46	Force a duplicate of a handle	As Function 45, additionally new file handle in CX at entry
47	Get current directory	Entry: drive number in DL (0 = default, 1 = A etc); DS:DI points to 64 byte memory area Return: AX = 15: invalid drive
48	Allocate memory	Entry: size of requested allocation in BX Return: carry reset: AX points to allocated memory; carry set and AX = 7 or 8: error
49	Free allocated memory	Entry: segment address of area to be freed in ES Return: carry reset: OK; carry set: error

Figure 4.7 (5 of 6)

Function (Hex)	Description	Entry/Return Parameters
4A	Modify size of allocated memory area	Entry: segment address of memory area in ES; requested size in BX Return: carry reset: OK; carry set and A = 7, 8 or 9: error
4B	Load and execute a program using parameters: WORD segment address of environment, DWORD pointer to command line at 80H; DWORD pointers to default FCBs to be passed at 5CH and 5DH. Load only using parameters: WORD segment address where file will be loaded, WORD relocation factor	Entry: DS:DX points to name of file to be loaded; ES:BX points to a parameter block; AL = 0: load/ execute; AL = 1: load only Return: carry reset: OK; carry set and AX = 1, 2, 8, 10 or 11: error
4C	Terminate current process and revert control to invoking process. No need to reset CS to Program Header Prefix. Files automatically closed	Entry: optional return codes in AL
4D	Retrieve return code of a child process	Return: exit code in AL; reason for exit in AH: 0 = terminate/abort, 1 = CONTROL-C, 2 = hard error, 3 = terminate but stay resident
4E	Find matching file (wildcards allowed) where attributes also match. Data block is written to current DMA	Entry: DS:DX points to pathname; CX contains attributes Return: carry reset: OK; carry set and AX = 2: invalid path; carry set and AX = 18: no match found
4F	Find next matching entry in directory. DMA address must point at block written by Function 4E	Return: carry reset: OK; carry set and AX = 18: no match found
54	Get verify flag	Return: verify flag in AL
56	Move a directory entry to another path on same drive	Entry: DS:DX points to pathname of existing file; ES:DI points to new pathnames Return: carry reset: OK; carry set and AX = 2, 5 or 17: error
57	Get or set date/time of file as soon as it is closed	Entry: if AL = 1 then BX contains file handle number, CX time to be set, DX date to be set Return: error if AX = 1 or 6; time/ date in CX/DX if AL was 0 at entry

Figure 4.7 (6 of 6)

CP/M-86 COMPATIBILITY

In the interest of compatibility with CP/M-86 programs, MS-DOS offers an alternative, restricted access to the Function Requests. This involves loading the function number into the CL register. Entry parameters can be loaded into the appropriate registers. Instead of issuing INT 21H, the program must call (short) location 0005 in the current Code Segment.

NOTE: This method can be used only with functions 00 through 24H, and only if register AL is not required for parameter transfer.

THE MS-DOS PROGRAM SEGMENT

When an external command is typed, or when you execute a program through the EXEC system call, MS-DOS determines the lowest available free memory address to use as the start of the program. This area is called the Program Segment.

The first 256 bytes of the Program Segment are set up by the EXEC system call for the program being loaded into memory. The program is then loaded following this block. An .EXE file with minalloc and maxalloc both set to zero is loaded as high as possible.

At offset 0 within the Program Segment, MS-DOS builds the Program Segment Prefix control block (see Figure 4.8). The program returns from EXEC by one of four methods:

1. A long jump to offset 0 in the Program Segment Prefix
2. By issuing an INT 20H with CS:0 pointing at the PSP
3. By issuing an INT 21H with register AH = 0 with CS:0 pointing at the PSP, or 4CH and no restrictions on CS
4. By a long call to location 50H in the Program Segment Prefix with AH = 0 or Function Request 4CH

NOTE: All programs must ensure that the CS register contains the segment address of the Program Segment Prefix when terminating via any of these methods, except Function Request 4CH. For this reason, using Function Request 4CH is the preferred method.

All four methods result in transferring control to the program that issued the EXEC. During this returning process, Interrupts 22H, 23H, and 24H (Terminate Address, CONTROL-C Exit Address, and Fatal Error Abort Address) addresses are restored from the values saved in the Program Segment Prefix of the terminating program. Control is then given to the terminate address. If this is a program returning to COMMAND.COM, control transfers to its resident portion. If a batch file was in process, it is continued; otherwise, COMMAND.COM performs a checksum on the transient part, reloads it if necessary, then issues the system prompt and waits for you to type the next command.

When a program receives control, the following conditions are in effect.

The segment address of the passed environment is contained at offset 2CH in the Program Segment Prefix.

The environment is a series of ASCII strings (totaling less than 32K) in the form:

NAME = parameter

Each string is terminated by a byte of zero, and the set of strings is terminated by another byte of zero. The environment built by the command processor contains at least a COMSPEC = string (the parameters on COMSPEC define the path used by MS-DOS to locate COMMAND.COM on disk). The last PATH and PROMPT commands issued will also be in the environment, along with any environment strings defined with the MS-DOS SET command.

The environment that is passed is a copy of the invoking process environment. If your application uses a "keep process" concept, you should be aware that the copy of the environment passed to you is static. That is, it will not change even if subsequent SET, PATH, or PROMPT commands are issued.

Offest 50H in the Program Segment Prefix contains code to call the MS-DOS function dispatcher. By placing the desired function request number in AH, a program can issue a far call to offset 50H to invoke an MS-DOS function, rather than issuing an Interrupt 21H. Since this is a call and not an interrupt, MS-DOS may place any code appropriate to making a system call at this position. This makes the process of calling the system portable.

The Disk Transfer Address (DTA) is set to 80H (default DTA in the Program Segment Prefix).

File control blocks at 5CH and 6CH are formatted from the first two parameters typed when the command was entered. If either parameter contained a pathname, then the corresponding FCB contains only the valid drive number. The filename field will not be valid.

An unformatted parameter area at 81H contains all the characters typed after the command (including leading and imbedded delimiters), with the byte at 80H set to the number of characters. If the <, >, or parameters were typed on the command line, they (and the filenames associated with them), will not appear in this area; redirection of standard input and output is transparent to applications.

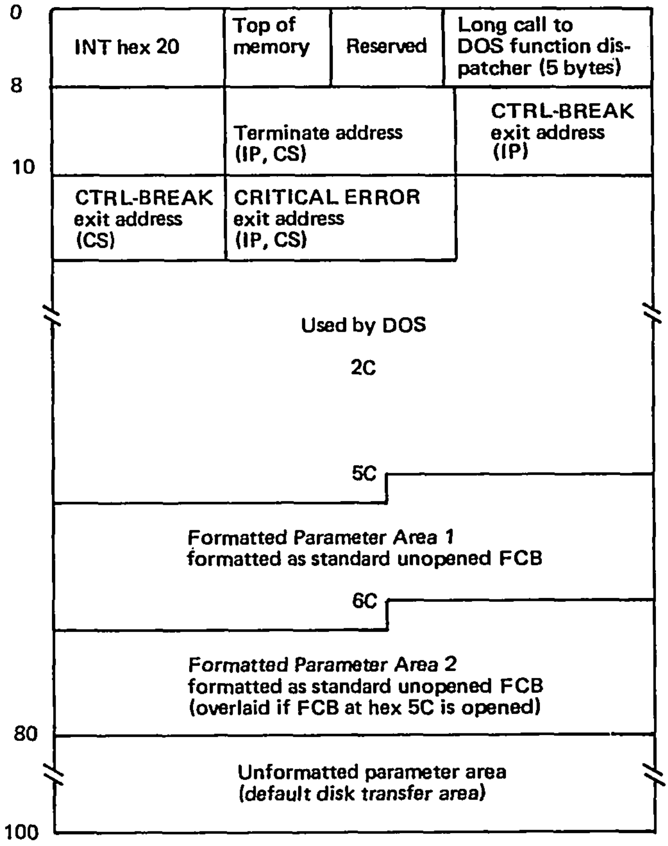

0
INT hex 20
Top of memory
Reserved
Long call to DOS function dispatcher (5 bytes)
8
Terminate address (IP, CS)
CTRL-BREAK exit address (IP)
10
CTRL-BREAK exit address (CS)
CRITICAL ERROR exit address (IP, CS)
Used by DOS
2C
5C
Formatted Parameter Area 1 formatted as standard unopened FCB
6C
Formatted Parameter Area 2 formatted as standard unopened FCB (overlaid if FCB at hex 5C is opened)
80
Unformatted parameter area (default disk transfer area)
100

Figure 4.8 MS-DOS Program Segment Prefix

Offset 6 (one word) contains the number of bytes available in the segment.

Register AX indicates whether or not the drive specifiers (entered with the first two parameters) are valid, as follows:

AL = FF if the first parameter contained an invalid drive specifier (otherwise AL = 00)
AH = FF if the second parameter contained an invalid drive specifier (otherwise AH = 00)

Offset 2 (one word) contains the segment address of the first byte of unavailable memory. Programs must not modify addresses beyond this point unless they were obtained by allocating memory via the Allocate Memory system call (Function Request 48H).

Figure 4.8 illustrates the layout of the Program Segment Prefix. All offsets are in hexadecimal. You should note:

- First segment of available memory is in segment (paragraph) form (for example, hex 1000 would represent 64K).
- The word at offset 6 contains the number of bytes available in the segment.
- Offset hex 2C contains the segment address of the environment.

SETTING OF CPU REGISTERS

Upon loading an EXE or COM program, MS-DOS sets the segment registers, instruction pointer, and stack pointer.

EXE PROGRAMS

DS and ES registers are set to point to the Program Segment Prefix.

CS, IP, SS, and SP registers are set to the values passed by MS-LINK.

COM PROGRAMS

All four segment registers contain the segment address of the initial allocation block that starts with the Program Segment Prefix control block.

All of user memory is allocated to the program. If the program invokes another program through Function Request 4BH, it must first free some memory through the Set Block (4AH) function call, to provide space for the program being executed.

The Instruction Pointer (IP) is set to 100H.

The Stack Pointer register is set to the end of the program's segment. The segment size at offset 6 is reduced by 100H to allow for a stack of that size.

A word of zeros is placed on top of the stack. This is to allow a user program to exit to COMMAND.COM by doing a RET instruction last. This assumes, however, that the user has maintained his stack and code segments.

TERMINAL FUNCTIONS

This section concerns the possibilities of software manipulation of the CRT display and loudspeaker output. MS-DOS recognizes a number of codes which are applicable to cursor movement, partial or whole screen clearance, variation of CRT intensity, and activating the loudspeaker. One or more functions are possibly not implemented on some machines. This section summarizes the function codes. It must be appreciated that functions cannot be attributed to specific keys on the keyboard. This is because there is a wide variety of keyboards available for different parts of the world. By checking in the relevant column for a particular keyboard in the chapter "Keyboard Codes" in the Hardware Description, it is, however, possible to find keys which can be used in setting a particular function.

MS-DOS on your NCR DECISION MATE V recognizes the function codes used by the Lear Siegler ADM-31™ terminal, with the following exceptions: 17H (Clear to End of Line), 1BH with 2AH, and 1BH with 3AH (Clear Screen and Cursor Home), and 1BH with 4DH (Play Music) are implemented in your NCR DECISION MATE V. The Lear Siegler ADM-3A terminal uses the functions which do not commence with 1BH (exception: 17H — Clear to End of Line). Figure 4.9 presents a summary of the function codes which, with the exceptions stated here, are compatible with these Lear Siegler terminals.

MS-DOS on your NCR DECISION MATE V also recognizes the MS-DOS ANSI function codes. Figure 4.10 summarizes these functions.

If your NCR DECISION MATE V has a color CRT, you can refer to Figure 4.11 for details of graphic settings. (Those codes which do not involve color also work with a monochrome CRT.)

Using the function codes from Figure 4.11 you can concatenate any number of graphic parameter settings without repeating the introductory 1BH 5BH sequence. In that case, 3BH must be

TERMINAL FUNCTION CODES (1)	
Function	**Hexadecimal Code**
POSITION CURSOR* row + offset col + offset	1B 3D followed by row + 20 followed by col + 20
CURSOR LEFT (non-destructive backspace)	08
CURSOR DOWN (line feed)	0A
CURSOR RIGHT (non-destructive forward space)	0C
CURSOR UP (reverse line feed)	0B
CURSOR HOME (top left corner)	1E
CLEAR SCREEN and CURSOR HOME	1A or 1B 2A or 1B 3A
CLEAR TO END OF LINE	17 or 1B 54 or 1B 74
CLEAR TO END OF SCREEN	1B 59 or 1B 79
CARRIAGE RETURN	0D
ESCAPE	1B
INSERT LINE	1B 45
INSERT CHARACTER	1B 51
DELETE LINE	1B 52
DELETE CHARACTER	1B 57
HALF INTENSITY OFF	1B 28
HALF INTENSITY ON (Yellow on color CRT)	1B 29
NORMAL VIDEO & BLINKING OFF	1B 47 30
REVERSE VIDEO	1B 47 34
BLINKING ON	1B 47 32
RING THE BELL	07
MUSIC (see Figure 4.12)	1B 4D followed by Frequency in the range 21 to 4A, or 20 = no tone followed by Length in the range 20 to FF (steps of 20ms)

Figure 4.9

TERMINAL FUNCTION CODES (2)	
Function	**Hexadecimal Code preceded by 1BH 5BH**
POSITION CURSOR*	row 3B col 48 or row 3B col 66
CURSOR LEFT	# of cols 44
CURSOR RIGHT	# of cols 43
CURSOR DOWN	# of rows 42
CURSOR UP	# of rows 41
DEVICE STATUS REPORT	36 6E
CURSOR POSITION REPORT* returned after Device Status Report	row 3B col 52
SAVE CURSOR POSITION	73
RESTORE CURSOR POSITION	75
ERASE SCREEN	32 4A
ERASE TO END OF LINE	4B

*NOTE: In terms of the Lear Siegler codes, the cursor origin is designated 0,0. ANSI describes this position as 1,1.

Figure 4.10

TERMINAL FUNCTION CODES (3)	
Function	**Hexadecimal Code preceded by 1BH 5BH, concluded by 6DH**
GRAPHIC ATTRIBUTES OFF	30
HALF INTENSITY OFF	31
BLINKING ON	35
INVERSE VIDEO ON	37
HALF INTENSITY ON	38
BLACK FOREGROUND	33 30
RED FOREGROUND	33 31
GREEN FOREGROUND	33 32
YELLOW FOREGROUND	33 33
BLUE FOREGROUND	33 34
MAGENTA FOREGROUND	33 35
CYAN FOREGROUND	33 36
WHITE FOREGROUND	33 37
BLACK BACKGROUND	34 30
RED BACKGROUND	34 31
GREEN BACKGROUND	34 32
YELLOW BACKGROUND	34 33
BLUE BACKGROUND	34 34
MAGENTA BACKGROUND	34 35
CYAN BACKGROUND	34 36
WHITE BACKGROUND	34 37

Figure 4.11

MUSIC CODES

NOTE	FREQUENCY	CYCLES
PAUSE	20	–
A	21	110
A#	22	116.5
B	23	123.5
C	24	131
C#	25	138.6
D	26	146.8
D#	27	155.8
E	28	164.8
F	29	174.6
F#	2A	185
G	2B	196
G#	2C	208
A	2D	220
A#	2E	233
B	2F	246.9
C (Middle C)	30	261.6
C#	31	277.4
D	32	293.7
D#	33	311
E	34	329.6
F	35	349.2
F#	36	370
G	37	392
G#	38	415
A	39	440
A#	3A	465
B	3B	493.9
C	3C	523.2
C#	3D	553
D	3E	587.3
D#	3F	622
E	40	659.3
F	41	698.5
F#	42	740
G	43	784
G#	44	830
A	45	880
A#	46	932
B	47	987.8
C	48	1046.5
C#	49	1108.7
D	4A	1174.7

Figure 4.12

present as a separator between each item. Note that the Graphic Attributes Off function does not disturb color settings, except where this results from the resetting of inverse or intensity attributes.

Figure 4.12 relates to the Play Music function (see Figure 4.9). The hexadecimal numbers in the Frequency column correspond to the frequency which is entered following the 1B 4D introductory string.

It is possible to program the Function Keys (F1 . . . F20) of your keyboard by use of the following hexadecimal sequence:

1B

5B

30

3B

Function number, using the appropriate value 0 . . . 14 (Hex).

3B

22

Text of new function

22

70

To disable the Function Keys, use the sequence 1B 5B 30 3B 30 70. To re-enable use 1B 5B 30 3B 39 39 70.

The advantage to the programmer of this method is that there is no need to return to the MS-DOS system level in order to program a Function Key via the CONFIG utility.

SOME I/O EXAMPLES

This section contains some short examples using the MS-DOS system calls. These examples are written in 8086 assembly language, using hexadecimal values only. If you are fortunate enough to have an assembler which can be used in conjunction with the operating system, you can, of courses, use the assembler and then load the executable file into memory for testing with the DEBUG utility. This utility is provided on your MS-DOS flexible disk, but a separate assembler utility is not. You can, however, use the limited assembler facility provided by the DEBUG (Command A). It is recommended that you first become thoroughly acquainted with the DEBUG before attempting to test these or other software examples.

CAUTION

Before experimenting with the I/O functions, it is advisable to make an additional copy of your system flexible disk and to work with this copy. In addition to the risk of data loss when accidentally activating disk controller routines, the DEBUG utility poses a danger in that the Write Command can bypass the file handler.

The main routine of each example can be written to CS:100. DEBUG notifies the user of the contents of CS at the beginning of assembly. As the DEBUG utility does not provide for the use of symbols, equate statements have not been used. However, address symbols have been used to give the reader a better overview. The hexadecimal suffix "H" is included, but you should note that the DEBUG assembler interprets all numbers as hexadecimal.

Your NCR MS-DOS Programmer's Manual contains a sample program for disk access.

MUSIC1

This example enables you to compose simple tunes on your NCR DECISION MATE V. Using the read keyboard function (01), the individual notes are entered via the keyboard and stored in memory. Following entry of the tune, it is played through the loudspeaker, using the function for console output (02).

The ASCII code for each key pressed is accepted by the program. From this value 20H is subtracted. This means, for example, that upper case A will yield the value 21H, which is the lowest frequency provided for by the music function. The "@" sign represents a pause (20H). The length of each note (or pause) is fixed in the LENGTH subroutine, but you can alter this value to any value between 20H (20ms) and 0FFH (4480ms). Entering the same note more than once consecutively will, of course, produce notes of different length. Similarly, pauses of different length can be produced. If you wish to correct an entry, you can do so by using the backspace key.

When you press the space bar, the program understands that you have finished entering your tune. Your tune will then be played through the loudspeaker. To play the tune again you should set the Instruction Pointer to PLAY.

To make alterations to the tune or alter LENGTH, you can make use of the Dump (D) and Enter (E) facilities provided by DEBUG. If you have stored your tune on disk, you will have to set the DS register yourself using the (R)egister facility after any subsequent reloading (L), unless you are intending to write a new tune.

Subroutines:

```
     NOTE:      MOVE  DL,1BH
                MOV   AH,02
                INT   21H
                MOV   DL,4DH
                MOV   AH,02
                INT   21H
                RET

     ; The above subroutine transmits the string 1BH 4DH to the console to indicate
     ; that the following two bytes represent frequency and length respectively (see
     ; Terminal Functions)

     INKEY:     MOV   AH,01      ; read key and echo to CRT
                INT   21H
                RET

     LENGTH:    MOV   DL,30H     ; length of each note – can be varied from
                MOV   AH,02H     ; 20H to 0FFH
                INT   21H
                RET
```

Main Routine:

```
                MOV   AX,CS      ; set position in memory where notes can be
                ADD   AX,200H    ; written.
                MOV   DS,AX
                XOR   BX,BX      ; points to memory byte for first note.

     NEXTIN:    CALL  INKEY
                CMP   AL,08      ; check for backspace.
                JNE   NOBKSP     ; if backspace then decrement pointer and
                DEC   BX         ; do not store.
                JMP   NEXTIN
     NOBKSP:    MOV   [BX],AL    ; store.
                CMP   AL,20H     ; space bar? if yes, start playing.
                JE    PLAY
                INC   BX         ; otherwise increment pointer and get next
                JMP   NEXTIN     ; note.
     PLAY:      XOR   BX,BX      ; reset pointer to first note.
     NXTOUT:    CALL  NOTE
                MOV   DL,[BX]    ; fetch note
                SUB   DL,20H
                CMP   DL,0       ; if no more notes then jump.
                JE    OVER
                MOV   AH,02      ; output function
                INT   21H
                CALL  LENGTH
                INC   BX         ; point to next note
                JMP   NXTOUT
     OVER:      NOP
```

Perhaps you would like to try this short tune:

```
N N I I N N @ @ R S P R N @ @ @
N N I I N N @ @ R S P R N @ @ @
R R R S U U U U W U S W U U @ R
R R P R S U U @ U W U S W U U @
@ N N N M N P R R @ R R R P R S
U U W S R R P P N N N N
```

MUSIC2

This example is really a planning aid for use with MUSIC1. The details are the same as for MUSIC1, except that the note requested is transmitted to the loudspeaker immediately after pressing the key without being stored.

Subroutines:

```
    NOTE:       MOV   DL,1B      ; see MUSIC1
                MOV   AH,02
                INT   21H
                MOV   DL,4D
                MOV   AH,02
                INT   21H
                RET

    INKEY:      MOV   AH,01
                INT   21H
                RET

    LNGTH2:     MOV   DL,28H
                MOV   AH,02
                INT   21H
                RET
```

Main Routine:

```
    NXTIN2:     CALL  INKEY
                CMP   AL,20H
                JE    OVER2      ; jump if space bar
                SUB   AL,20H
                PUSH  AX         ; preserves note
                CALL  NOTE
                POP   AX
                MOV   DL,AL      ; output note
                MOV   AH,02
                INT   21H
                CALL  LNGTH2
                JMP   NXTIN2
    OVER2:      NOP
```

KEYBOARD

This example reads each character as it is typed in from the keyboard and displays that character on the screen. Before the first character is accepted, the screen is cleared and the cursor set top left. If a numeric sign (0 . . . 9) is entered, inverse video is activated temporarily. The program terminates when a dollar sign ($) is entered, and normal video is restored if necessary.

Subroutines:

```
    C1C2:       MOV   DL,1BH    ; first 2 bytes of control sequence for inverse/
                MOV   AH,02     ; reverse
                INT   21H
                MOV   DL,47H
                MOV   AH,02
                INT   21H
                RET

    INVERS:     CALL  C1C2
                MOV   DL,34H    ; for inverse video
                MOV   AH,02
                INT   21H
                RET

    REVERS:     CALL  C1C2
                MOV   DL,30H    ; for normal video
                MOV   AH,02
                INT   21H
                RET
    CLSCRN:     MOV   DL,1BH    ; clear screen and cursor home
                MOV   AH,02
                INT   21H
                MOV   DL,3AH
                MOV   AH,02
                INT   21H
                RET

    INKEY:      MOV   AH,07     ; read keyboard
                INT   21H
                RET
```

Main Routine:

```
                CALL  CLSCRN
    NEXTCH:     CALL  REVERS    ; ensure normal video
                CALL  INKEY
                CMP   AL,24H    ; check for $
                JE    DONE
                CMP   AL,'0'    ;
                JC    PRINT     ; if ASCII <30H, no reverse
                CMP   AL,'9'
                JE    INVT
                JNC   PRINT     ; if ASCII >39H, no reverse
```

```
INVT:     PUSH  AX        ; save character
          CALL  INVERS
          POP   AX
PRINT:    MOV   DL,AL
          MOV   AH,02
          INT   21H
          JMP   NEXTCH
DONE:     NOP
```

If you substitute 32H for 34H in the subroutine INVERS, the display will blink. You do not have to alter the reset (REVERS) subroutine.

DUPLICATE

This example of I/O functions stores keyboard input in memory and duplicates the stored data on the printer as often as you wish. Starting with a clear screen you can enter data which is echoed to the screen. Carriage Return is recognized and also noted in the storage area, which means that you do not have to fill remaining line space with individual spaces via the keyboard. You may write more than one full screen, normal scrolling will then occur. Deletions using the backspace key are noted in memory.

To terminate data input, enter a dollar sign ($). Your data will now be directed to the printer, recognizing Carriage Return and Line Feed as previously entered from the keyboard. When the printer has finished you need only press R or r for a further print copy. You may repeat this as often as you wish.

Subroutines:

```
READIN:   MOV   AH,01     ; read keyboard
          INT   21H
          RET

CRTLF:    MOV   AH,02     ; produce line feed on CRT in response to CR
          MOV   DL,0AH
          INT   21H
          RET

PRTCR:    MOV   AH,05     ; carriage return on printer
          MOV   DL,0DH
          INT   21H
PRTLF:    MOV   AH,05     ; line feed on printer
          MOV   DL,0AH
          INT   21H
          RET
```

```
BLANK:    MOV   AH,02      ; delete previous character on CRT
          MOV   DL,20H
          INT   21H
          MOV   AH,02
          MOV   DL,08
          INT   21H
          RET

CLSCRN:   MOV   AH,02      ; clear screen and cursor home
          MOV   DL,1BH
          INT   21H
          MOV   AH,02
          MOV   DL,3AH
          INT   21H
          RET
```

Main Routine:

```
          MOV   AX,CS
          ADD   AX,200H
          MOV   DS,AX      ; set own DS.
          XOR   BX,BX      ; zero pointer to storage area
          CALL  CLSCRN
NEXT:     CALL  READIN
          MOV   [BX],AL    ; save key entry.
          INC   BX         ; increment pointer.
          CMP   AL,08      ; check for backspace.
          JNE   NOBACK
          DEC   BX         ; if backspace, do not record in memory
          DEC   BX
          CALL  BLANK
          JMP   NEXT
NOBACK:   CMP   AL,0DH     ; check for CR
          JNE   NOLF
          CALL  CRTLF      ; if CR, add LF
          JMP   NEXT
NOLF:     CMP   AL,24H     ; check for $
          JNE   NEXT
PRINT:    XOR   BX,BX      ; reset pointer
NEXTP:    MOV   DL,[BX]    ; fetch character from memory
          CMP   DL,24H     ; check for $
          JE    DONE
          MOV   AH,05      ; print if not $
          INT   21H
          CMP   DL,0DH     ; check for stored CR
          JNE   NONLIN
          CALL  PRTLF      ; if CR, add LF
NONLIN:   INC   BX         ; increment pointer
          JMP   NEXTP
DONE:     CALL  PRTCR      ; 4 clear lines
          CALL  PRTCR
          CALL  PRTCR
          CALL  PRTCR
          CALL  READIN
          CMP   AL,52H     ; if R then reprint
          JE    PRINT
          CMP   AL,72H     ; if r then reprint
          JE    PRINT
```

COLOR

This example is for the NCR DECISION MATE V with color CRT. It accepts input from the keyboard and echoes the data to the screen using the foreground and background colors of your choice. You can change the foreground (writing) color by entering the @ sign followed by the number of the color (0 . . . 7, see Figure 4.11). To set the background color, enter $ instead of @ Enter $$ to terminate the program.

Subroutines:

```
READIN:   MOV   AH,01      ; read key and echo to CRT
          INT   21H
          RET

CRTLF:    MOV   AH,02      ; produce line feed on CRT in response to CR
          MOV   DL,0AH
          INT   21H
          RET

CLSCRN:   MOV   AH,02      ; clear screen and cursor top left
          MOV   DL,1BH
          INT   21H
          MOV   AH,02
          MOV   DL,3AH
          INT   21H
          RET

BLANK:    MOV   AH,02
          MOV   DL,08
          INT   21H        ; backspace
          MOV   AH,02
          MOV   DL,20
          INT   21H        ; erase character
          MOV   AH,02
          MOV   DL,08      ; put cursor at erased character
          INT   21H
          RET

BEGCOL:   MOV   AH,02      ; control sequence to introduce color setting
          MOV   DL,1BH
          INT   21H
          MOV   AH,02
          MOV   DL,5BH
          INT   21H
          RET

ENDCOL:   MOV   AH,02      ; control sequence to conclude color setting
          MOV   DL,6DH
          INT   21H
          RET
```

```
QCOLOR:  MOV    AL,0FFH
         CMP    CH,40H     ; check for @
         JE     CHANGE
         CMP    CH,24H     ; check for $
         JNE    OVER
         CMP    CL,24H     ; $$?
         JNE    CHANGE
         MOV    AL,01
         JMP    OVER
CHANGE:  CMP    CL,30H
         JB     OVER
         CMP    CL,37H
         JA     OVER
         MOV    AL,0       ; only if number 0 . . . 7 for new color
OVER:    RET

; The above subroutine returns status of the last two keys pressed in AL. If a
; valid color change, then AL = 0; if terminate, then AL = 1; otherwise AL = FF
```

The main routine:

```
         CALL   CLSCRN
         XOR    CX,CX
NEXT:    MOV    CH,CL
         CALL   READIN
         MOV    CL,AL      ; last two keys in CX
         MOV    DL,AL
         CMP    DL,0DH
         JNE    NOLF
         CALL   CRTLF      ; if CR, then LF
         JMP    NEXT
NOLF:    CALL   QCOLOR
         CMP    AL,1
         JE     DONE       ; jump if CX = $$
         CMP    AL,0
         JNE    NEXT       ; jump if no color change
         CALL   BLANK      ; erase color change sequence
         CALL   BLANK
         CMP    CH,24H
         JNE    FOREGR
         MOV    CH,34H     ; color change is background
         JMP    COLSET
FOREGR:  MOV    CH,33H     ; color change is foreground
COLSET:  CALL   BEGCOL
         MOV    AH,02
         MOV    DL,CH
         INT    21H
         MOV    AH,02
         MOV    DL,CL
         INT    21H
         CALL   ENDCOL
         JMP    NEXT
DONE:    NOP
```

THE LOADING PROCEDURE

The NCR DECISION MATE V firmware (of which a listing is included in the Hardware Description) brings the bootloader from flexible disk into memory at machine address 2000H. A copy of the loader module is then made in upper memory to preserve it during the subsequent loading procedures.

The I/O system modules are then loaded from machine address 400H (i.e. immediately above the interrupt vector) upwards. A number of initialization modules are then placed above the I/O elements, followed by a Microsoft system module and the Winchester Disk driver module of BIOS Version 1, if applicable. The uppermost system module is the resident part of the Microsoft COMMAND.COM. The final stage of initialization after loading is the block transfer of system and Winchester driver modules downwards in memory, with the effect that modules no longer required are overwritten.

HOW TO READ THE I/O PROGRAM

Appendix C, which is included in this manual, contains the listings of the input/output software developed by NCR in 8086 assembly language. This is the software which enables the MS-DOS software to be used in the NCR DECISION MATE V hardware environment.

The I/O software can be considered as a number of modules. The initialization modules — LOADER (FDBOOT in Version 2), BASINIT, SYSINIT, SYSIMES, FWVERRD (firmware version read, in Version 2 contained in BASINIT) — are overwritten by resident system software modules, so that the lower memory from 400H is occupied as follows:

BIOS Version 1 (MS-DOS 2.0)	BIOS Version 2 (MS-DOS 2.11)
KBD-DRV (keyboard driver) DSKDRV (flex. disk driver) followed by the standard MS-DOS I/O software and, if required, by WIDRV (installable Winchester driver)	IOBASE, KBDCRT, COMDRV, LPDRV, TIMDRV, DSKDRV and, if required, WIDRV5 or WIDRV10 (5 or 10MB Winchester driver) followed by the standard MS-DOS I/O software

and by the resident part of COMMAND.COM (MS-DOS).

The link maps of both BIOS versions are included in Appendix C as part of the program listings. Using these maps, you can resolve public symbol definitions as absolute values.

The I/O software listings contained in the Appendix relate directly to the hardware characteristics of your NCR DECISION MATE V. It is important to remember that different hardware requires different I/O software. For example, the presence of a Winchester disk unit requires an additional software driver; a color CRT requires more extensive drive parameters than a monochrome CRT. Your MS-DOS software, and the listings in the Appendix of this manual, provide for the maximum demand on I/O software, for example, the I/O software for both a Winchester disk unit and a color CRT is included. However, if you wish to make use of the I/O routines without using the MS-DOS standard Interrupts and Function Requests, it is advisable to check the exact contents and machine address of the routine required. The following section "Displaying the I/O software on the screen" can assist you in this.

If you are intending to obviate the MS-DOS standard Interrupts and Function Requests, you must bear in mind that you are at the same time failing to take advantage of the upwards compatibility offered by your operating system. As hardware developments take place, you may well have to alter machine addresses used in your own software so that your software can be used on other machines or with other versions of the operating system.

DISPLAYING THE I/O SOFTWARE ON THE SCREEN

Included in your MS-DOS flexible disk is a utility program DEBUG. This program enables you to view areas of memory as a hexadecimal dump or as 8086 assembly language, and to inspect the CPU registers. For full details regarding use of the DEBUG utility, please refer to the description situated near the end of your NCR MS-DOS manual.

The IO.SYS module is loaded to machine address 400H. By adding the length of the preceding modules to this figure, you can calculate the start address of the module you wish to examine. Having loaded DEBUG from disk, you can set the Code Segment accordingly. There are two ways of viewing memory. The D command produces a hexadecimal dump on the screen, while the U command disassembles memory (without provision of symbols). The disassembly is obviously more readable, however, you must ascertain first, that disassembly begins with the first byte of an MS-DOS machine code instruction, and second, that the area of memory to be disassembled contains no data or unused areas. Otherwise, disassembly will be incorrect or fail altogether.

Therefore, it is advisable to use the D command for the initial orientation in the machine memory, comparing the screen dump with the hexadecimal codes contained in the listings. If you do not find the byte sequence you expected to find at a particular address, you should look in the vicinity of that address: A good point of orientation is often a "literal" such as an error message, as these can be easily recognized without closer examination of the screen dump. You should also remember that the DEBUG utility provides you with a search facility (S command), with which you can search for the first occurrence of a byte sequence in a chosen area of memory.

PORTS

The following is a summary of the I/O ports used by the MS-DOS software. For each port, the hexadecimal port number is given, as well as information regarding its use.

CAUTION

> The ports in your NCR DECISION MATE V are used not only by your operating system, but also by the firmware which becomes active at power up. Under no circumstances should you attempt to make use of IN or OUT (including block transfer) instructions at ports which are connected to Timer functions, otherwise permanent damage to your computer may result. A detailed map of the NCR DECISION MATE V ports is given at the end of this section (Figure 4.13).

OUT 10
Switches the firmware ROM into the address area 0-1FFFH, thus de-selecting RAM in this area.

OUT 11
De-selects the firmware ROM, thus assigning the address area 0-1FFFH to main memory.

IN 13
Interrupt signal from the disk controller sets bit 3. Bit 0 is used to check whether the motor is switched on (set = not on).

OUT 14
Bit 0 is used to turn the motor on.

OUT 26

The DMA address is transmitted via this port, first the low byte followed by the high byte without any intervening command output.

OUT 27

The DMA length is transmitted via this port, first the low byte followed by the high byte without any intervening command output.

OUT 2A

Bits 0 and 1 are set to enable the FCD channel following initialization of the DMA. Setting bit 0, 1, and 2 disables the FDC channel.

OUT 2B

Sets the DMA mode. To set the read mode, bits 0, 1, 2, 3, and 6 are set, the others reset. For the write mode, bits 0, 1, 2, and 6 are set, the others reset.

IN 40

Reads a character from the keyboard.

IN 41

A character from the keyboard is ready if bit 0 is set. The language code is ready if bit 7 is set.

OUT 41

Drives the loudspeaker. Output value 1 constitutes an instruction to return the country code during keyboard initialization.

IN 50

Bit 7 set indicates that flexible disk is ready.

IN 51

Used to read information from the flexible disk controller.

OUT 51

Used in the transmission of disk, head, and track number to the flexible disk controller. Also used to transmit formatting information.

IN 60

Reads in data from the serial interface, including XON/XOFF status.

OUT 60

Output port for parallel data transmission.

IN 61

This status port for the serial interface is used to detect overrun, parity, or framing errors. Bit 3 set indicates a framing error, bit 5 a parity error, and bit 4 an overrun. Bit 1 set is used to indicate that a character has been received. Bit 0 set indicates that the transmit holding register is empty.

For the parallel interface, bit 1 set or bit 5 set indicates that the device is not yet ready.

IN 63, OUT 67

Read and write command information. Out 37H enables transmitter and receiver.

OUT 63

Used to initialize the parallel interface.

OUT 64

Output port for serial data transmission.

OUT 66

Used to initialize the serial interface. The first of the two output commands determines stop bits, parity, and character length. The second command determines the baud rate.

IN A0

Used to determine whether the graphics display controller can accept a character. Bit 1 reset means a character can be transmitted. Bit 0 set means that data is ready for transmission to the GDC. Bit 3 set means that drawing is actually being carried out.

OUT A0

Used for output of drawing parameters to the GDC.

IN A1

Read GDC-RAM contents.

OUT A1

Output of command information to the GDC.

IN C0

Block input of data from the Winchester disk controller (512 bytes at a time).

OUT C0

Block output of data to the Winchester disk controller (512 bytes at a time).

IN C1

Yields a detailed definition of an error detected upon reading from a Winchester disk. Bit 5 set denotes an error in the ID field revealed by the Cyclic Redundancy Check. Bit 6 set indicates an error in the data field. If neither of these two bits is set, the error cannot be defined.

OUT C2

Used in formatting the Winchester disk.

OUT C3

Used to set a sector number of the Winchester disk. Output 0AAH used for drive ready check.

LOW HIGH	0	1	2	3	4	5	6	7
0	ERROR LEDS							
1	RAMSEL	ROMSEL	SETTC	SYSSTAT	MOTOR			
2								
3	IFSEL 2A K806							
4	KEY: R/W DATA	KEY: R/W COMMAND						
5	FDC: R-MAIN STATUS	FDC: R/W DATA						
6	IFSEL 0A K210, K211, K213							
7	IFSEL 1A K211, K215							
8	TIMER: R/W COUNTER 0	TIMER: R/W COUNTER 1	TIMER: R/W COUNTER 2	TIMER: W-MODE				
9								
A	GDC R-STATUS W-PARAM	GDC R-DATA W-COMMAND	ZOOM					
B	IFSEL 3A							
C	IFSEL 4A WINCHESTER DISK							
D	16-BIT SWITCH							
E	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM
	RAM BANKS 0 - 7							
F	I/O EXPANSION							

NOTE: In Figure 4.13, the numbers prefixed with K refer to kits available for the NCR DECISION MATE V. A switchable RS-232C interface (K801) which can use any of the IFSEL codes, is also available. When using this kit, you should bear in mind that both hardware and software strapping are required.

This applies also to K215, K803, K804, and K806, as these kits are switchable in the same way. You should therefore ensure that your software can use all the IFSEL codes. The IFSEL codes given in Figure 4.13 for these kits constitute default suggestions.

When using K801 with a plotter, you should select IFSEL 0A.

IFSEL 0B is required by hardware in conjunction with K210, K212, and K213.

Figure 4.13 (1 of 2)

LOW HIGH	8	9	A	B	C	D	E	F
0	TIMER COUNTER 0	TIMER COUNTER 1	TIMER COUNTER 2	TIMER WRITE MODE	8255 PORT A: LED	8255 PORT B: SWITCH	8255 PORT C: CONTROL	8255 COMMAND
	DIAGNOSER							
1								
2	DMA: R-STATUS W-COMMAND	DMA: W-REQ. REG.	DMA: W-FDC ENABLE	DMA: W-MODE	DMA: CLR POINTER	DMA: R-MASTER CLEAR	DMA: CLR MASK REG.	DMA: W-ALL MASK BITS
3	IFSEL 2B K804							
4								
5								
6	IFSEL 0B							
7	IFSEL 1B							
8								
9								
A								
B	IFSEL 3B K600							
C	IFSEL 4B K803							
D								
E								
F								

Figure 4.13. (2 of 2)

OUT C4

Used to set a cylinder number. Output 55H used for drive ready check.

OUT C5

The higher order part of the cylinder number.

OUT C6

Transmits information to the Winchester disk controller regarding drive, head, sector size, and error checking. All this information is passed in a single output.

IN C7

Accepts status information from the Winchester disk controller. Bit 7 set indicates that the controller is busy. Bit 6 set indicates that the drive is not ready. Bit 4 set indicates that the drive search is not completed. Bit 0 set indicates an error (see IN C1).

OUT C7

Selects the Winchester disk read (20H) or write (30H) function.

OUT D0

Bit 0 set switches to the Z-80® processor. If the Z-80 processor is presently activated, the 16-bit processor becomes active in its place.

Figure 4.13 provides a complete map of the port addresses used by your NCR DECISION MATE V.

INTERFACING PRINTERS

The following presents a brief summary of the signals essential to the operation of the user's serial or parallel printing device. The exact pin configuration and cable requirements are given in the "Hardware Description."

This is the sequence of signals between NCR DECISION MATE V and a serial printer:

NCR DECISION MATE V	PRINTER
1.	Printer sets XON signal to enable computer to transmit data.
2. Transmission is enabled, so data are transmitted bit by bit via the TxD line.	

3. When the printer buffer is nearly (typically 3/4) full, an XOFF signal is generated.

4. The computer waits with further data . . . . . .while the printer empties its buffer.

5. When the buffer is empty, XON is once again generated.

6. Data transmission is once again enabled.

The XOFF status is equivalent to 13H being read IN at port 60. Otherwise XON is assumed. The DTR and DSR lines are connected together inside the serial printer interface kit. In addition CTS and RTS should be connected together. Both these combinations and the CD line should be at +12V (i.e. ON).

For the parallel (Centronics) interface the procedure is similar. Printer Busy or Printer Buffer Full return 20H and 02H respectively. Therefore, if neither bit 1 nor bit 5 is set upon a read IN at port 61, the printer is ready to receive data.

2651 REGISTER ADDRESSING						
Port (Hex) K212 K213	K211	Signals Required * CE	BA0	BA1	BA2	Function
–	–	1	X	X	X	Tri-state data bus
60	74	0	0	0	0	Read receive holding register
64	70	0	0	0	1	Write transmit holding register
61	71	0	1	0	0	Read status register
65	75	0	1	0	1	Write SYN1/SYN2/DLE registers
62	72	0	0	1	0	Read mode registers 1/2
66	76	0	0	1	1	Write mode registers 1/2
63	73	0	1	1	0	Read command register
67	77	0	1	1	1	Write command register

* These pin designations (see Hardware Description) correspond to the following bus lines: BA0 - A0, BA1 - A1, BA2 - $\overline{R}$/W.

Figure 4.14

For full details of interface connections and the significance of the individual control lines, you can refer to the Hardware Section. Users of non-NCR serial printers which do not use XON/XOFF protocol can, with the aid of the printer manufacturer's description, find suitable lines for connection to the K211, K212, or K213 adapter.

For details of the serial and parallel interface integrated circuits and their programming procedures, advanced programmers should refer to the manufacturers' software descriptions of the integrated circuits used (not included in this description). The serial interface IC is the 2651, the parallel interface IC is the 8255.

A 2651 is used not only for the serial printer interface, but also for the serial communications interface kit (K211, see Hardware Description). Figure 4.14 summarizes the actual port addresses used by these two interfaces.

CAUTION

The user must take extreme care when connecting an external device to a peripheral adapter. You should not only read the relevant parts of the "Hardware Description" in this manual, but also the equivalent information concerning the external device to be connected. Failure to take device characteristics into consideration will mean that the software will not function. It may also result in permanent damage to your computer, adapter, or external device.

LEVEL ZERO DIAGNOSTICS

Output to port 00 controls the LED panel situated next to peripheral adapter slot 7. Output zero turns all LEDs on, output FF turns all LEDs off. Figure 4.15 shows the errors indicated by various LED-on combinations. The LED numbers refer to the numbers printed on the LED panel.

LED ON	OUT PORT 00	SIGNIFICANCE
None	FF	Check complete
1+8	7E	Sumcheck error
2+8	BE	GDC error
3+8	DE	Disk drive error
4+8	EE	16-bit processor error
5+8	F6	Keyboard error
6+8	FA	DMA error
7+8	FC	Memory error
All	00	Processor error

Figure 4.15

GRAPHICS

The operating system software provides you with full access to the character set of your NCR DECISION MATE V. The parameters used in the generation of the CRT display are contained in a 32KB RAM (96KB for color CRTs) accessed via the ports A0 and A1.

A graphics utility program such as NCR-GRAPH provides you with comfortable access to the full graphic capacity beyond that of the character generator contained in the firmware.

If you otherwise wish to access the Graphics Display Controller (GDC), you will find this section especially useful.

The PD7220-1 GDC integrated circuit has an addressing capacity of 256K words of 16 bits each. Facilities provided by the GDC include light pen input, figure drawing of lines, arcs, rectangles, and graphic characters, area filling, and zoom magnification. Communication between GDC and CPU is via the GDC's first-in-first-out buffer. Commands to determine a particular mode of operation are received by the GDC at port A1 (i.e. via the processor OUT AL,0A1H instruction). Data and other parameters following a particular command are received at port A0. Status information can be read at port A0 (IN AL,0A0H instruction), and data from the GDC can be read via port A1.

This section deals with the aspects of programming the GDC which relate to its environment in your NCR DECISION MATE V. Following this, you will find a sample programming session consisting of graphic producing routines which you may wish to adapt and expand for your own applications.

THE GRAPHICS DISPLAY CONTROLLER

The GDC integrated circuit in your NCR DECISION MATE V addresses a CRT display consisting of 640 pixels in the horizontal, and 400 pixels in the vertical direction. The top left-hand corner of the CRT is regarded as the origin of the GDC map. The top (horizontal) line of the screen is represented by the first 640 pixels, the next pixel addresses the far left of the second line, and so on. The GDC makes use of a two-level addressing mode: a word address refers to 16 consecutive pixels, while a 4-bit dot position (values 0-15) refers to an individual pixel within that word. A FIFO buffer is used to pass commands and data to and from the CPU. (Use of the DMA option bypasses this buffer). The contents of this buffer are destroyed only upon a reset or reversal of the direction from read to write or vice versa.

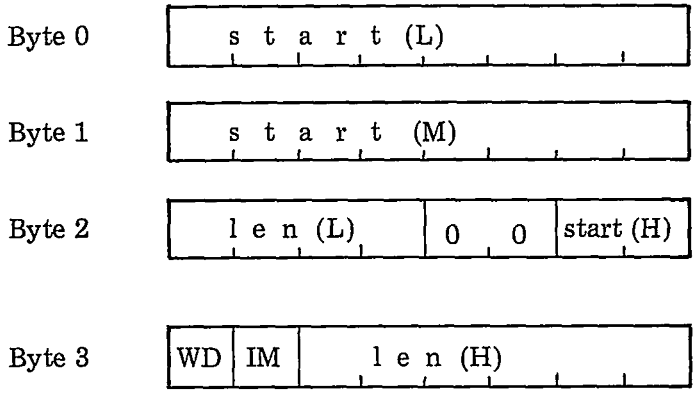
The GDC includes a second buffer, the parameter RAM, in which parameters for figure and character drawing can be loaded and retained. GDC commands which do not explicitly load the parameter RAM do not affect its contents. Therefore, it is possible to make repeated use of the parameter RAM contents without having to reload it. It is even possible to load a specified part of the parameter RAM without altering the rest of its contents.

The GDC has two basic modes of operation, namely the Character Mode and the Mixed (Graphics and Character) Mode. The power-up initialization procedure automatically sets the Mixed Mode, as this results in the most efficient non-graphic screen writing in the NCR DECISION MATE V hardware environment. To enable figure drawing it is sufficient to set a flag in the appropriate GDC command. Some additional parameters significant for CRT operation are also sent to the GDC during the power-up initialization. They include horizontal and vertical sync width, horizontal and vertical front and back porch width, type of video framing (non interlaced), type of RAM (dynamic), and the drawing time mode (drawing only during retrace). In the normal course of graphics programming you do not need to set or alter these parameters. However, if you wish to investigate in detail this hardware-related initialization procedure, you can refer to the Hardware Description which comprises the first volume of the System Technical Manual. This first volume includes a listing of the initialization program of the NCR DECISION MATE V firmware in Z-80 assembly language. You may also wish to refer to the manufacturer's description of the PD7220-1 integrated circuit.

The Parameter RAM

This 16-byte memory area, which is included within the integrated circuit, is used in the Mixed Mode to define two display partition areas and to hold an 8 x 8 pixel graphics character ready for transmission to the display memory. If a figure, and not a graphics character, is to be drawn, the parameter RAM can be used to store a drawing pattern of dots and dashes. The exact layout of the parameter RAM is as follows. Remember that to use the addressing capability of the GDC to the full, an address may consist of up to 18 bits.

Bytes 0-3: these four bytes define the display partition area 1. The start address of this area in display memory is contained in 18 bits. Bytes 0 and 1 contain the least and medium significant byte respectively, while the two most significant bits of the address are contained at bits 0 and 1 of byte 2. The length of this display partition is held in 10 bits (bits 4-7 of byte 2 and, more significant, bits 0-5 of byte 3).



The bit at IM must be set to indicate a bit-mapped graphics area (reset would denote a character area). The bit at WD, which indicates whether 32-bit (wide = set) or 16-bit accessing is activated, should be 0 (reset).

Bytes 4-7: identical structure, this time for definition of display partition area 2.

Bytes 8-15: this area can be used for storing a bit-mapped graphic character in an 8 x 8 pixel format. Upon execution of the appropriate drawing instruction, this area of the parameter RAM is scanned from the least significant bit of byte 15 towards its most significant bit. Scanning then continues from the most significant bit of byte 14 towards its least significant bit, and so on. If the area to be filled by the parameter RAM is greater than the 8-pixel square, a further subset of the RAM is transmitted to the CRT. If the screen area to be filled is smaller than the 8-pixel square, only a subset of the parameter RAM will appear. Later in this section, you can read how to determine the area on the CRT to be filled, and how to create a slanting (italics) effect.

If you instruct the GDC to do figure drawing instead of drawing a graphic character from the parameter RAM, you can use bytes 8 and 9 for pattern purposes, e.g. to draw dotted or dashed lines.

Remember that the parameter RAM contents are preserved beyond completion of a figure or graphic character drawing instruction, so you can make repeated use of the parameter RAM without having to reload it.

GDC Status Information

Information regarding the busy or otherwise status of the GDC can be read in at port A0. The eight bits thus read by the processor have the following significance.

Bit 0: when set (1), indicates that a byte of data from the GDC RAM is available for reading. The bit is automatically reset as soon as the data transfer from the GDC begins.

Bit 1: when set, this bit indicates that the FIFO buffer is full. Therefore, programs should check that this flag is not set before transmitting a command or parameters to the GDC.

Bit 2: when set, this bit indicates that the FIFO buffer is empty. It is not necessary, nor desireable, to make output to the GDC dependent upon this bit being set, as this would mean dispensing with the advantages offered by buffering. Bit 2 is, however, useful, in that you know that your last command or parameter to the GDC has been accepted from the buffer, if this bit is set.

Bit 3: set while a graphic figure is being drawn.

Bit 4: set while a DMA transfer with the GDC is in progress.

Bit 5: set while vertical retracing on the CRT is in progress.

Bit 6: set while horizontal retracing is in progress. The GDC is set during initialization not to draw during active display time, in order to eliminate display disturbances.

Bit 7: set indicates that the light pen address register contains a deglitched value for the processor.

Commands and their Parameters

The graphics display controller accepts via its FIFO buffer certain commands and parameters which affect the display on the CRT. The following presents a summary of these commands, with special emphasis on those which are of importance to the setting up of user graphics. The first byte issued to the GDC in each case is the command byte. The bytes (if any) which follow the command byte are the obligatory, or sometimes optional, parameters belonging to that command. The command byte in your NCR DECISION MATE V must always be transmitted via port A1, the parameters via A0. The GDC regards the parameters for the old command as concluded, as soon as a new command is issued. This is true even if the parameter list for the old command is incomplete.

Reset – This command blanks the display, resets the FIFO buffer and the command processor, and sets idle mode.

Command byte: 0.

This command can be issued at any time for the above mentioned purpose. It does not destroy the contents of graphic display memory. RESET can be followed by eight parameters to set mode of display, type of video framing, type of graphic display RAM, number of active display words per line, horizontal and vertical sync, front porch and back porch widths, and the number of active display lines per video field. The tasks are all carried out at power-up initialization so these parameters do not have to be accessed for the purpose of user graphics. The precise initialization procedure is contained in the firmware listings included in the Hardware Description of the System Technical Manual (Volume 1).

Sync: – Command byte: 0FH (display enabled) or 0EH (display blanked).

The output parameters are the same as those for the reset command. However, Sync does not reset the GDC or activate idle mode.

Vertical Sync – Command byte: 6EH (slave) or 6FH (master).

This command is meaningful only when more than one GDC is being used to create one image.

Cursor and Character – Command byte: 4BH.

This is normally used to set up the cursor by means of 3 parameter bytes.

Byte 1	CD	0	0	Lines				
Byte 2	BL (L)		CB	Top				
Byte 3	Bottom					BL (H)		

Lines refers to the number of display lines to be used for each character row, minus 1. If the CD bit is reset, the cursor is not displayed. Top contains the top line number in the row defined by Lines. If CB is reset, the cursor will blink in accordance with the speed set in BL low and high. For graphics this command is significant inasmuch as the cursor must be set to non-display mode and the number of display lines must be set to zero. In this case, there is no need to transmit bytes 2 and 3.

Start Display – Command byte: 6BH, no parameters.

The GDC leaves the idle mode and enters the display mode.

Display On/Off – Command byte: 0CH (display blanked) or 0DH (display active), no parameters.

Zoom – Command byte: 46H.

The single parameter byte which follows this command indicates in its four most significant bits a zoom factor for the entire display, or in its least significant bits, a zoom factor for the graphics character which is about to be transmitted to the GDC. In each case the value 0 indicates no magnification. Magnification, if set, takes place in both x and y directions. A zoom factor specified for a graphic character determines the actual bit-mapping in graphic display memory, so that the enlarged image remains irrespective of subsequent use of the zoom facility. A display zoom factor, on the other hand, does not alter the bit map of the graphic display memory.

Position Cursor – Command byte: 49H.

Byte 1	Word Address (L)		
Byte 2	Word Address (M)		
Byte 3	Dot	0 0	WA (H)

Word Address (upper 2 bits in byte 3) indicates a 16-pixel boundary, and Dot a pixel position offset to that boundary, where the cursor is to be situated. The character mode does not require parameter byte 3. Remember that the origin for counting word addresses is the top left corner of the CRT. As the GDC in your NCR DECISION MATE V addresses 640 x 400 pixels, a total of 18 bits address capacity is required. This means that WA (H) will be zero. The cursor position in a graphics application is an imaginary one, as it would not usually be desirable to display a cursor.

Load Parameter RAM – This command loads the parameter RAM from a position in that RAM (1 to 15) with the ensuing parameter bytes.

Command byte: bit 7 zero; bits 4, 5, and 6 are set. The four least significant bits contain a value between 0 and 15, according to where in the parameter RAM loading should start.

Example: The command byte 78H tells the GDC that the parameters at port A0 should be loaded into the parameter RAM starting at byte 8, and working towards byte 15.

Pitch – Command byte: 47H.

The single byte parameter contains the number of word addresses in a horizontal line of display. The GDC drawing instructions require this information for calculating the word above or below the current word. This value is set at power-up initialization in your NCR DECISION MATE V. The pitch value is also set by the Reset and Sync commands.

Write Data – This command is an instruction to the GDC to write one word or byte of data into display memory. Following this, the cursor position is advanced in the last specified direction (see Figure) to the next word address. It is possible to specify a word or byte write. In the latter case, only one, not two, parameters are accepted. In the case of bit-map graphics, only parameter byte 1 is significant, and only then when all bits are set or all bits are reset. In a coded character situation, the bits of the parameter byte(s) set the drawing pattern.

The command byte differs according to the type of transfer and the logical operation which is to govern the write operation.

Command	0	0	1	Type	0	Logic

A zero value in two bits for Type indicates write Word (Low), then Word (High); the value 2 determines that Word (Low), the value 3 that Word (High) should be transmitted; value 1 is invalid. A zero value in two bits for Logic determines that the word or byte addressed by the cursor is to be replaced by the pattern contained in the one or two byte parameters; value 1 means that the individual pixel is to be complemented if the corresponding bit in the pattern is set; analogously, value 2 means reset to zero; and value 3 means set to 1. As already stated, the parameters consist of one or two bytes:

Byte 1	Word (L) or Byte
Byte 2	Word (H)

It is admissable to supply further parameter bytes without repeating the command. These will be applied to the automatically advanced cursor position.

The Write Data command must be preceded by a Figure command (only the first three bytes are required, see Figure).

Mask – Command byte: 4AH, followed by two parameter bytes, namely Mask (Low), then Mask (High).

This command sets a 16-bit mask for subsequent figure drawing (the same mask is set by parameter byte 3 of the Position Cursor command). Mask is usually used for clearing or filling large areas of memory, with all the mask bits set. For pixel by pixel drawing there is no need to use the Mask command, as the Cursor Position command can specify the pixel position.

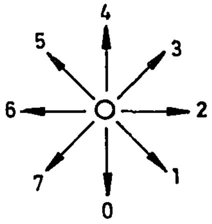
Figure – This command, using as many as 11 parameter bytes, is used for specifying whether individual dot or figure drawing is to take place, and in the latter case, it specifies the figure to be drawn. Beyond this, it is also used for determining the direction of activity for any screen writing. DMA activity also requires certain Figure parameters.

Command byte: 4CH.

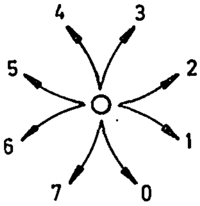
Byte 1	SL	R	A,C	G	L	Direction

The significance of the individual bits of byte 1 is as follows. SL = slanted graphics character, R = rectangle drawing, A,C = arc or circle drawing, G = graphics character, L = line drawing. None of these bits set denotes individual pixel drawing, character screen writing or reading, or a DMA transfer.

Direction refers to a 3-bit value for the direction of drawing, emanating from the last pixel drawn.



In terms of arc drawing from a point, the following diagram applies:



The remaining parameters are distributed over the remaining ten bytes as follows:

Byte 2	D 1 (L)		
Byte 3	0	MG	D 1 (H)
Byte 4	D 2 (L)		
Byte 5	0	0	D 2 (H)
Byte 6	D 3 (L)		
Byte 7	0	0	D 3 (H)
Byte 8	D 4 (L)		
Byte 9	0	0	D 4 (H)
Byte 10	D 5 (L)		
Byte 11	0	0	D 5 (H)

Bit MG in byte 2 must be set to denote graphics drawing.

The values required for the parameters D1 to D5:

Initial values
D1 = 0; D2 = 8; D3 = 8; D4 = all bits set; D5 = all bits set.

Pixel plotting
As initial values.

Line drawing
D1 = the distance covered on the x or y axis, whichever is the greater; D2 = 2 * the distance on the other axis, then subtract D1; D3 = 2 * the shorter minus the longer distance;

D4 = 2 * the shorter of the two distances; D5 = initial setting. D2 and D3 require two's complement notation, other values are absolute. The Direction value for the Figure command must contain the octant in which line drawing is to take place.

Arc drawing

D1 = radius of curvature * sine of angle between major axis and end of arc (max. 45°); D2 = one pixel less than the radius of curvature; D3 = 2 * D2; D4 = all bits set; D5 = radius of curvature * sine of angle between major axis and beginning of arc (max. 45°), then rounded down to next integer.

Rectangle drawing

D1 = 3; D2 = number of pixels in direction specified in command byte, minus one; D3 = number of pixels in direction at right angle to direction specified in command byte, minus one; D4 = all bits set; D5 = D2.

Filling an area

D1 = one less than the number of pixels at right angle to direction specified in command byte; D2 = number of pixels in direction specified in command byte; D3 = D2.

Graphic Character

This process is really a case of area filling, where the number of pixels in each direction is < = 8. If that number in the direction specified in the command byte is 8, there is no need to load D2 and D3.

Writing data

D1 = number of display words required, minus 1. All other parameters are of no significance.

Write via DMA

D1 = number of words to be accessed in direction at right angle to direction specified in command byte, minus one; D2 = number of bytes to be transferred in the other direction, minus one; other parameters are not significant.

Read via DMA

D1 = number of words to be accessed in direction at right angle to direction specified in command byte; D2 = number of bytes to be transferred in the initially specified direction, minus two; D3 = D2/2 (required only for word read); D4 and D5 are not significant.

Read data via CPU

D1 = number of words to be accessed; other parameters are not significant.

Draw – Command byte: 6CH, no parameters.

Drawing is started at the pixel indicated by the current cursor position, and in accordance with bytes 8 and 9 in the parameter RAM and the drawing parameters set by Figure.

Draw Graphics Character – Command byte: 68H, no parameters.

As in Draw, except that the 8 x 8 pixel pattern in parameter RAM bytes 8-15 is drawn.

Read Data from Graphic Display Memory – This command reverses the direction of the FIFO buffer if it has so far been used for transferring data to the GDC. This means the loss of any commands or parameters in the buffer which follow the Read Data command. The structure of the command byte is:

1	0	1	Type	0	Logic

A zero value for Type denotes a word read (low then high). Value 2 indicates low byte of word only, value 3 high byte only. Value 1 is not valid. The Logic value (see Write Data) determines the state in which the graphic display memory will be after reading. Assuming that you wish only to read data and not modify them in any way, this value must be zero.

Reading data from graphic display memory requires that you state the number of words to be read by means of the Figure command. In addition you must set the Direction, and, if this is neither 0 nor 4, you should issue a Mask command with all the parameter bits set. Perhaps the most easily understandable Direction setting is 2, as this accesses the addresses in ascending order, i.e. left to right, then the next line down, and so on. Do not forget to ensure that the cursor is in the position where you wish reading to commence. It is also advisable to check the data ready status bit (bit 1) before each read.

Each byte of data can be read by the CPU at port A1, whereupon a further byte is loaded by the GDC into its FIFO buffer. A read sequence can be discontinued by transmitting a command to the GDC. Otherwise, reading is continued until D1 (see Figure command) decrements to zero.

Read Current Address of Cursor – Command byte: 0E0H.

The cursor address is returned via the FIFO in the following format:

Byte 1	Word Address (L)						
Byte 2	Word Address (M)						
Byte 3	0	0	0	0	0	0	WA (H)
Byte 4	Dot (L)						
Byte 5	Dot (H)						

Note that the dot position is not represented by a binary value in 4 bits, but as one set bit among 15 zero bits.

DMA Transfer – Command byte for read request:

1	0	1	Type	1	Logic

Command byte for write request:

0	0	1	Type	1	Logic

The significance of Type and Logic bits is the same as for the Read Data command.

Before the transfer can be executed, the Figure command must be issued with appropriate parameters (see Figure). The cursor must be positioned and the Mask register bits must be all set. As DMA transfers bypass the FIFO buffer, its contents are not affected.

GDC Status Considerations

When transmitting data to the GDC, it is important that the FIFO buffer does not overflow. Checking status bit 1 before transmitting ensures that there is space in the FIFO for at least one command or parameter byte. Alternatively, the processor could wait for the buffer to become empty (status bit 2), and then transmit up to

16 bytes. Whichever method you choose, you should not transmit data to the GDC merely on the assumption that the FIFO buffer will have passed on some of its contents for execution. Especially during figure drawing there are always delays, during which no bytes are taken from the buffer.

The GDC makes use of a separate data register to help eliminate delays in providing data at the read port. Nonetheless, it is advisable to check bit 0 (data ready) of the GDC status. If you are using status bit 1 (FIFO full) to synchronize GDC data output with processor data reading, your program should not make an early termination (i.e. termination before D1 has decremented to zero) of the read sequence dependent on the FIFO buffer not being full. The status bit will not be reset as long as the buffer is full of read data, so if your new command byte is waiting for this bit to reset, your program will loop.

SOME GDC PROGRAMMING EXAMPLES

The assembly language routines contained in this section are designed to provide you with a starting point for the development of your own graphics. They include examples of how to set your cursor position, draw rectangles, arcs and circles, and how to do pixel by pixel drawing under keyboard control. Instructions are also given about how to read the character generator of the firmware ROM in your NCR DECISION MATE V, and how to store and restore your graphic designs. A number of arithmetic routines for pixel calculation are also included.

These and similar graphic routines can be written with the symbolic assembler provided with your operating system software. Following assembly, you can test and adapt the routines using the debugging utility which is also present on your operating system flexible disk.

The stage by stage program construction in this section introduces each DB or DW at the time of discussion of the first routine which makes use of that particular storage definition. The most convenient way of organizing the Code and Data Segment Registers is to set CS and DS to the same value. Assuming that you have assembled one or more of these routines using the MS™-Macro Assembler (available as an option from NCR), you have to make use of the LINK utility to create an EXE file. Upon loading the EXE file into machine memory using the DEBUG utility, you will note that the paragraph value in DS is 10 (Hexadecimal) lower than that contained in CS. The first two lines of the program (PUSH CS POP DS) in conjunction with the ASSUME directives set DS to the higher paragraph value contained in CS. Therefore, it is imperative to run these first two lines as soon as the EXE file has been loaded into machine memory. To do so, you must use the G instruction within DEBUG with a breakpoint (G=0 2):

```
0000                    CSEG SEGMENT
                        ASSUME CS:CSEG,DS:CSEG
0000  0E                        PUSH CS     ;execute these two
0001  1F                        POP DS      ;instructions before
                                            ;anything else
                        ;
0002  0000              SPSTORE:  DW 0
```

The 16-bit area SPSTORE is included in order to remind you to consider setting up your own user stack. This might become necessary if you intend to extend the graphics examples. However, the LINK utility creates an EXE file even without an explicit stack segment.

OUTC is a routine for transmitting a command byte to the GDC. Upon entry, the command byte must be in register AL, Transmission takes place only when there is no drawing in progress and the FIFO buffer is capable of receiving at least one byte.

```
0004  50                OUTC:     PUSH AX
0005  E4 A0             OUTC1:    IN AL,0A0H
0007  24 0A                       AND AL,0AH
0009  75 FA                       JNZ OUTC1
000B  58                          POP AX
000C  E6 A1                       OUT 0A1H,AL
000E  C3                          RET
```

OUTP transmits a number of parameters. Upon entry, the number of parameters must be contained in register DL, the first parameter must be addressed by BX.

```
000F  E4 A0             OUTP:     IN AL,0A0H
0011  24 0A                       AND AL,0AH
0013  75 FA                       JNZ OUTP
0015  8A 07             OUTP1:    MOV AL,BYTE PTR [BX]
0017  E6 A0                       OUT 0A0H,AL
0019  43                          INC BX
001A  FE CA                       DEC DL
001C  75 F7                       JNZ OUTP1
001E  C3                          RET
```

Therefore, you could arrange parameters for graphics initialization as follows:

```
001F  00                 PRAMS     DB 0
0020  08                           DB 8
0021  00 00 00 59 00 00  PRAMS1    DB 0,0,0,59H,0,0,0,59H,0FFH,0FFH,
                                      0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
      00 59 FF FF FF FF
      FF FF FF FF
0031  00 00 00           PRAMS2    DB 0,0,0
0034  FF FF              PRAMS3    DB 0FFH,0FFH
0036  02 FF 7F 08 00 08  PRAMS4    DB 2,0FFH,7FH,8,0,8,0,0FFH,3FH,0FFH,3FH
      00 FF 3F FF 3F
0041  FF FF              PRAMS5    DB 0FFH,0FFH
0043  21                 WRLOGIC   DB 21H          ;complement
```

GINIT is the routine which transmits these parameters:

```
0044  8D 1E 001F R      GINIT:    LEA BX,PRAMS
0048  B0 0C                        MOV AL,0CH      ;bit 0 blanks screen
004A  E8 0004 R                    CALL OUTC
004D  B0 46                        MOV AL,46H      ;set zoom to zero
004F  E8 0004 R                    CALL OUTC
0052  B2 01                        MOV DL,1
0054  E8 000F R                    CALL OUTP
0057  B0 4B                        MOV AL,4BH      ;cursor/char
                                                   ;characteristics.
0059  E8 0004 R                    CALL OUTC
005C  B2 01                        MOV DL,1        ;parameter sets lines
                                                   ;per row to zero.
005E  E8 000F R                    CALL OUTP
0061  B0 70                        MOV AL,70H      ;load entire
                                                   ;parameter RAM.
0063  E8 0004 R                    CALL OUTC
0066  B2 10                        MOV DL,10H
0068  8D 1E 0021 R                 LEA BX,PRAMS1
006C  E8 000F R                    CALL OUTP       ;sets graphics and
                                                   ;400 pixels vertical.
006F  B0 49                        MOV AL,49H      ;set cursor pos
0071  E8 0004 R                    CALL OUTC
0074  B2 03                        MOV DL,3
0076  8D 1E 0031 R                 LEA BX,PRAMS2
007A  E8 000F R                    CALL OUTP       ;first pixel addressed
007D  B0 4A                        MOV AL,4AH      ;set mask
```

```
007F  E8 0004 R                   CALL OUTC
0082  B2 02                       MOV DL,2
0084  8D 1E 0034 R                LEA BX,PRAMS3
0088  E8 000F R                   CALL OUTP
008B  B0 4C                       MOV AL,4CH    ;figure parameters
008D  E8 0004 R                   CALL OUTC
0090  B2 0B                       MOV DL,0BH
0092  8D 1E 0036 R                LEA BX,PRAMS4
0096  E8 000F R                   CALL OUTP     ;no geom. figs,
                                                ;direction east.
0099  B0 22                       MOV AL,22H    ;write data word high
                                                ;then low, reset to 0.
009B  E8 0004 R                   CALL OUTC
009E  B2 02                       MOV DL,2
00A0  8D 1E 0041 R                LEA BX,PRAMS5
00A4  E8 000F R                   CALL OUTP
00A7  B0 21                       MOV AL,21H    ;write data,
                                                ;this time complement.
00A9  A2 0043 R                   MOV WRLOGIC,AL
00AC  E8 0004 R                   CALL OUTC
00AF  B0 0D                       MOV AL,0DH    ;re-enable screen
00B1  E8 0004 R                   CALL OUTC
00B4  E8 00BC R         WAIT:     CALL GETKEY
00B7  3C 24                       CMP AL,'$'
00B9  75 F9                       JNE WAIT
00BB  C3                          RET
```

Command 0CH blanks the screen. The first parameter at PRAMS is used for setting zoom to zero, the second sets the number of display lines per character row to zero. Command 70H means start loading the parameter RAM at the first byte. The parameters used (PRAMS1) set up one display partition, starting at the address zero in graphic display memory with length 400 (display lines). The remaining parameters are initialized to all bits set. This is of significance in the case of parameter RAM bytes 8 and 9, as this will ensure that figure drawing is carried out with unbroken lines. Command 49H sets the cursor to the beginning of the display area. Remember that this corresponds to the top left corner on the CRT. If you wish to use Cartesian coordinates, your programs will require additional calculations. Command 4AH uses PRAMS3 to set the mask register with all bits set. PRAMS4 contains the initial values for figure drawing (dot drawing, direction East). Command 22H uses PRAMS5 and the Logic setting 2 (reset to zero) to set the entire bit-map to zero. Command 21H sets the complement Logic for future drawing and writing. This state of Logic is also recorded in the byte WRLOGIC. Finally, the screen is re-enabled.

Further processing is now dependent on entering $ at the keyboard. The GETKEY routine for reading the keyboard must be careful not to attempt to output a character to the CRT, once the GDC is in graphics mode. In order to suppress this screen echo, the direct I/O function of the operating system is used. This routine will be invaluable in the keyboard-controlled drawing described later. GETKEY returns the key pressed in register AL.

```
00BC  52                 GETKEY:    PUSH DX
00BD  B4 06                         MOV AH,6
00BF  B2 FF                         MOV DL,0FFH
00C1  CD 21                         INT 21H
00C3  5A                            POP DX
00C4  C3                            RET
```

Assuming that you wish to return to normal character writing after completion of your graphics routines, you require an exit routine to restore the status prior to graphic processing. This routine is at any rate to be recommended when using the debugging tool, so that you can inspect registers and memory afterwards. The parameters starting at EXPRAMS are used by the exit routine GEXIT.

```
00C5  8F 00              EXPRAMS    DB 8FH,0
00C7  00 90 00 01 00 FF  EXPRAMS1   DB 0,90H,0,1,0,0FFH,0FFH,0FFH,0FFH,
                                       0FFH,0FFH,0FFH,0FFH
      FF FF FF FF FF FF
      FF
                         ;
00D4  8D 1E 00C5 R       GEXIT:     LEA BX,EXPRAMS
00D8  B0 4B                         MOV AL,4BH
00DA  E8 0004 R                     CALL OUTC
00DD  B2 01                         MOV DL,1
00DF  E8 000F R                     CALL OUTP
00E2  B0 46                         MOV AL,46H
00E4  E8 0004 R                     CALL OUTC
00E7  B2 01                         MOV DL,1
00E9  E8 000F R                     CALL OUTP
00EC  B0 70                         MOV AL,70H
00EE  E8 0004 R                     CALL OUTC
00F1  B2 0D                         MOV DL,0DH
```

```
00F3  E8 000F R                        CALL OUTP
00F6  B2 1A               CLSCRN:      MOV DL,1AH
00F8  B4 02                            MOV AH,2
00FA  CD 21                            INT 21H
00FC  C3                               RET
```

Command 4BH resets the number of display lines per character row to 16. 46H ensures that zoom is set to zero. Following this, the parameter RAM bytes are set. The IM bit is now reset, so that graphics display memory is no longer to be regarded as bit-mapped. Finally, the screen is cleared and the cursor set top left.

As the next stage, we can reserve an area for cursor position (CURPRAMS) and create a routine, CURSET, for transmitting that position to the GDC. CURPRAMS contains in 2 bytes (lower location = less significant byte) the word position, the third byte (highest location) must contain in its four uppermost bits the dot address within that word (see Position Cursor). The values used here in the DB directives will place the cursor 131,584 pixels from the beginning of display memory (no special significance to this value), that is, approximately halfway along the 206th line of the 400 line display.

```
00FD  20                  CURPRAMS     DB 20H
00FE  20                               DB 20H
00FF  00                               DB 0
                          ;
0100  B0 49               CURSET:      MOV AL,49H
0102  E8 0004 R                        CALL OUTC
0105  8D 1E 00FD R                     LEA BX,CURPRAMS
0109  B2 03                            MOV DL,3
010B  E8 000F R                        CALL OUTP
010E  C3                               RET
```

Now reserve an area for storing figure drawing parameters:

```
010F  00 00 00 00 00 00 FIGPRAMS     DB 0,0,0,0,0,0,0,0,0,0,0
      00 00 00 00 00
```

Enter the routine for transmitting these parameters to the GDC

```
011A  B0 4C               FIGSET:      MOV AL,4CH
011C  E8 0004 R                        CALL OUTC
011F  8D 1E 010F R                     LEA BX,FIGPRAMS
```

```
0123  B2 0B                              MOV DL,0BH
0125  E8 000F R                          CALL OUTP
0128  C3                                 RET
```

and the command which sets drawing in progress.

```
0129  B0 6C                 FIGDRAW:     MOV AL,6CH
012B  E8 0004 R                          CALL OUTC
012E  C3                                 RET
```

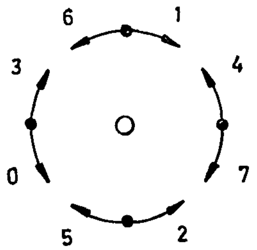
All that is now required are actual parameters for figure drawing. The following can be used for drawing a square:

```
012F  40 03 40 30 00 30 FIGPRAM1   DB 40H,3,40H,30H,0,30H,0,0FFH,3FH,30H,0
      00 FF 3F 30 00
```

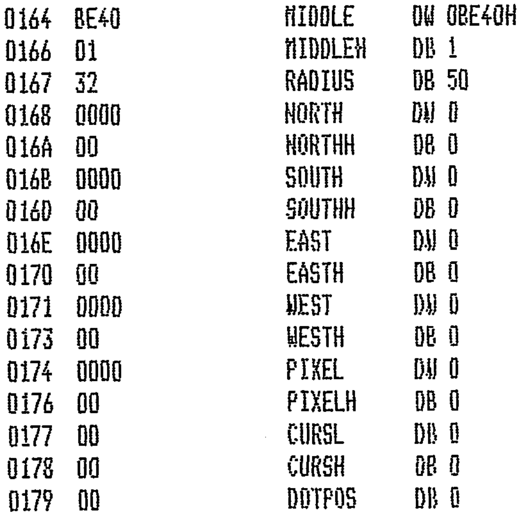
The routines described hitherto can now be used in a program to draw a square. First, the actual parameters in FIGPRAM1 are copied to the 11-byte FIGPRAMS area, as this is where the FIGSET routine expects to find them. Then the GDC is set up for graphics. Enter $, whereupon the cursor is set and the figure drawn. The figure will remain on the screen until you enter x. After the initial run, you may wish to experiment with the values in CURPRAMS and FIGPRAM1.

```
013A  8D 1E 012F R                       LEA BX,FIGPRAM1
013E  8D 3E 010F R                       LEA DI,FIGPRAMS
0142  B1 0B                              MOV CL,0BH
0144  8A 07                 NEXTPR1:     MOV AL,BYTE PTR [BX]
0146  88 05                              MOV BYTE PTR [DI],AL
0148  43                                 INC BX
0149  47                                 INC DI
014A  FE C9                              DEC CL
014C  75 F6                              JNZ NEXTPR1
014E  E8 0044 R                          CALL GINIT
0151  E8 0100 R                          CALL CURSET
0154  E8 011A R                          CALL FIGSET
0157  E8 0129 R                          CALL FIGDRAW
015A  E8 00BC R             WAIT2:       CALL GETKEY
015D  3C 78                              CMP AL,'x'
015F  75 F9                              JNE WAIT2
0161  E8 0004 R                          CALL GEXIT
```

To draw a circle, it is necessary to draw 8 arcs each turning through 45°. The arcs are drawn from four points around the centre of the circle, using the following Direction values:



Begin by setting up the data storage areas as follows:



```
0164  BE40        MIDDLE    DW 0BE40H
0166  01          MIDDLEH   DB 1
0167  32          RADIUS    DB 50
0168  0000        NORTH     DW 0
016A  00          NORTHH    DB 0
016B  0000        SOUTH     DW 0
016D  00          SOUTHH    DB 0
016E  0000        EAST      DW 0
0170  00          EASTH     DB 0
0171  0000        WEST      DW 0
0173  00          WESTH     DB 0
0174  0000        PIXEL     DW 0
0176  00          PIXELH    DB 0
0177  00          CURSL     DB 0
0178  00          CURSH     DB 0
0179  00          DOTPOS    DB 0
```

The first three bytes contain the pixel position in up to 18 bits (MIDDLEH = most significant byte, upper 6 bits reset) of the centre of the circle. The initial values used here place this point approximately halfway along the 179th display line. Using this position and RADIUS, the North, South, East, and West points on the circumference of the circle can be calculated. These pixel values are returned in NORTH, NORTHH, etc. as 3-byte values, the third byte in each case being the most significant byte. Do not, for the moment, alter the value in RADIUS.

```
017A  53               COMPASS:  PUSH BX
017B  51                         PUSH CX
017C  52                         PUSH DX
017D  BA 0280                    MOV DX,280H                ;pitch
0180  A1 0164 R        CNORTH:   MOV AX,WORD PTR MIDDLE
0183  8A 0E 0167 R               MOV CL,RADIUS
0187  32 ED                      XOR CH,CH
0189  8A 1E 0166 R               MOV BL,MIDDLEH
018D  F8               NDCR:     CLC
018E  1B C2                      SBB AX,DX
0190  80 DB 00                   SBB BL,0
0193  E2 F8                      LOOP NDCR
0195  A3 0168 R                  MOV WORD PTR NORTH,AX
0198  88 1E 016A R               MOV NORTHH,BL
019C  A1 0164 R        CSOUTH:   MOV AX,WORD PTR MIDDLE
019F  8A 0E 0167 R               MOV CL,RADIUS
01A3  32 ED                      XOR CH,CH
01A5  8A 1E 0166 R               MOV BL,MIDDLEH
01A9  F8               SDCR:     CLC
01AA  13 C2                      ADC AX,DX
01AC  80 D3 00                   ADC BL,0
01AF  E2 F8                      LOOP SDCR
01B1  A3 016B R                  MOV WORD PTR SOUTH,AX
01B4  88 1E 016D R               MOV SOUTHH,BL
01B8  A1 0164 R        CEAST:    MOV AX,WORD PTR MIDDLE
01BB  8A 0E 0167 R               MOV CL,RADIUS
01BF  32 ED                      XOR CH,CH
01C1  8A 1E 0166 R               MOV BL,MIDDLEH
01C5  F8               EDCR:     CLC
01C6  15 0001                    ADC AX,1
01C9  80 D3 00                   ADC BL,0
01CC  E2 F7                      LOOP EDCR
01CE  A3 016E R                  MOV WORD PTR EAST,AX
01D1  88 1E 0170 R               MOV EASTH,BL
01D5  A1 0164 R        CWEST:    MOV AX,WORD PTR MIDDLE
01D8  8A 0E 0167 R               MOV CL,RADIUS
01DC  32 ED                      XOR CH,CH
01DE  8A 1E 0166 R               MOV BL,MIDDLEH
01E2  F8               WDCR:     CLC
01E3  1D 0001                    SBB AX,1
01E6  80 DB 00                   SBB BL,0
01E9  E2 F7                      LOOP WDCR
01EB  A3 0171 R                  MOV WORD PTR WEST,AX
01EE  88 1E 0173 R               MOV WESTH,BL
```

```
01F2  5A                         POP DX
01F3  59                         POP CX
01F4  58                         POP AX
01F5  C3                         RET
```

The following routine is useful for converting a 3-byte pixel value into a format appropriate to the Position Cursor command, that is, as a 16-bit word address and one additional byte with a 4-bit dot-position value in bits 4-7. Upon entry to WORDAD, the pixel value must be available in PIXEL and (most significant) PIXELH. The word address and dot position will be returned in CURSL (least significant) and CURSH, with the dot position in DOTPOS.

```
01F6  53               WORDAD:    PUSH BX
01F7  51                          PUSH CX
01F8  52                          PUSH DX
01F9  A1 0174 R                   MOV AX,WORD PTR PIXEL
01FC  8A D0                       MOV DL,AL
01FE  B1 04                       MOV CL,4
0200  D3 E8                       SHR AX,CL
0202  8A 36 0176 R                MOV DH,PIXELH
0206  D2 E2                       SHL DL,CL
0208  02 E6                       SHL DH,CL
020A  0A E6                       OR AH,DH
020C  88 26 0178 R                MOV CURSH,AH
0210  A2 0177 R                   MOV CURSL,AL
0213  88 16 0179 R                MOV DOTPOS,DL
0217  5A                          POP DX
0218  59                          POP CX
0219  5B                          POP BX
021A  C3                          RET
```

The next routine, CURTRANSF, does no more than copy at CURPRAMS the cursor position in CURSL, CURSH, and DOTPOS. This means that the cursor position calculated by WORDAD can be used by the CURSET routine.

```
021B  8D 1E 00FD R     CURTRANSF: LEA BX,CURPRAMS
021F  A1 0177 R                   MOV AX,WORD PTR CURSL
0222  89 07                       MOV WORD PTR [BX],AX
0224  43                          INC BX
0225  43                          INC BX
0226  A0 0179 R                   MOV AL,DOTPOS
```

```
0229  88 07                        MOV BYTE PTR [BX],AL
022B  C3                           RET
```

The program to draw two 45° arcs, one on each side of the northmost point of the circumference, can now be put together. The initialization of the graphics mode is the same procedure as when drawing the rectangle. Following this, COMPASS calculates pixel values for the North, South, East, and West positions. The word address is calculated for North and placed at CURPRAMS so that the cursor can be set:

```
022C  E8 0044 R                    CALL GINIT
022F  E8 017A R                    CALL COMPASS
0232  A1 016B R                    MOV AX,WORD PTR NORTH
0235  A3 0174 R                    MOV WORD PTR PIXEL,AX
0238  A0 016A R                    MOV AL,BYTE PTR NORTHH
023B  A2 0176 R                    MOV BYTE PTR PIXELH,AL
023E  E8 01F6 R                    CALL WORDAD
0241  E8 021B R                    CALL CURTRANSF
0244  E8 0100 R                    CALL CURSET
```

The next step is to set up FIGPRAMS with the parameter for figure drawing. Note that drawing parameters D1, D2, D3, and D5 contain values which apply specifically to the chosen radius of 50 pixels. Therefore, if you change the radius, you will have to adjust these parameters or write a routine to do this for you. The most interesting parameter in FIGPRAMS is the first. The bit for arc drawing remains set throughout the program but the three Direction bits require different values between 0 and 7, depending on the arc to be drawn (see figure immediately following the rectangle program). The values for drawing the two arcs from the North point are 1 and 6. This program draws the Direction 1 arc first.

```
0247  8D 1E 010F R                 LEA BX,FIGPRAMS
024B  C6 07 21                     MOV BYTE PTR [BX],21H
                                            ;type of drawing = arc,
                                            ;direction = 1.
024E  43                           INC BX
024F  C6 07 23                     MOV BYTE PTR [BX],23H
                                            ;rsin 45 for radius
                                            ;50 pixels
0252  43                           INC BX
0253  C6 07 40                     MOV BYTE PTR [BX],40H
```

```
                                                  ;graphics drawing flag
0256  43                               INC BX
0257  C6 07 31                         MOV BYTE PTR [BX],31H
                                                  ;one less than radius
025A  43                               INC BX
025B  C6 07 00                         MOV BYTE PTR [BX],0
                                                  ;upper bits zero
025E  43                               INC BX
025F  C6 07 62                         MOV BYTE PTR [BX],62H
                                                  ;2 * (radius-1)
0262  43                               INC BX
0263  C6 07 00                         MOV BYTE PTR [BX],0
                                                  ;upper bits zero
0266  43                               INC BX
0267  C6 07 FF                         MOV BYTE PTR [BX],0FFH   ;D4
026A  43                               INC BX
026B  C6 07 3F                         MOV BYTE PTR [BX],3FH
026E  43                               INC BX
026F  C6 07 00                         MOV BYTE PTR [BX],0      ;D5
0272  43                               INC BX
0273  C6 07 00                         MOV BYTE PTR [BX],0
                        ;
0276  E8 011A R                        CALL FIGSET
0279  E8 0129 R                        CALL FIGDRAW
```

Then follows the Direction 6 arc:

```
027C  E8 0100 R                        CALL CURSET
027F  C6 06 010F R 26                  MOV BYTE PTR FIGPRAMS,26H
0284  E8 011A R                        CALL FIGSET
0287  E8 0129 R                        CALL FIGDRAW
```

Once the arcs at the point North on the circumference have been drawn, the program can proceed to convert the pixel value for South into a cursor position, set the cursor position, and draw the southern arcs. The two arcs at East and the two arcs at West are drawn in the same way.

```
028A  A1 016B R                        MOV AX,WORD PTR SOUTH
028D  A3 0174 R                        MOV WORD PTR PIXEL,AX
0290  A0 016D R                        MOV AL,BYTE PTR SOUTHH
0293  A2 0176 R                        MOV BYTE PTR PIXELH,AL
0296  E8 01F6 R                        CALL WORDAD
0299  E8 021B R                        CALL CURTRANSF
```

```
029C  E8 0100 R                        CALL CURSET
029F  C6 06 010F R 22                  MOV BYTE PTR FIGPRAMS,22H
02A4  E8 011A R                        CALL FIGSET
02A7  E8 0129 R                        CALL FIGDRAW
02AA  E8 0100 R                        CALL CURSET
02AD  C6 06 010F R 25                  MOV BYTE PTR FIGPRAMS,25H
02B2  E8 011A R                        CALL FIGSET
02B5  E8 0129 R                        CALL FIGDRAW
                             ;
02B8  A1 016E R                        MOV AX,WORD PTR EAST
02BB  A3 0174 R                        MOV WORD PTR PIXEL,AX
02BE  A0 0170 R                        MOV AL,BYTE PTR EASTH
02C1  A2 0176 R                        MOV BYTE PTR PIXELH,AL
02C4  E8 01F6 R                        CALL WORDAD
02C7  E8 021B R                        CALL CURTRANSF
02CA  E8 0100 R                        CALL CURSET
02CD  C6 06 010F R 24                  MOV BYTE PTR FIGPRAMS,24H
02D2  E8 011A R                        CALL FIGSET
02D5  E8 0129 R                        CALL FIGDRAW
02D8  E8 0100 R                        CALL CURSET
02DB  C6 06 010F R 27                  MOV BYTE PTR FIGPRAMS,27H
02E0  E8 011A R                        CALL FIGSET
02E3  E8 0129 R                        CALL FIGDRAW
                             ;
02E6  A1 0171 R                        MOV AX,WORD PTR WEST
02E9  A3 0174 R                        MOV WORD PTR PIXEL,AX
02EC  A0 0173 R                        MOV AL,BYTE PTR WESTH
02EF  A2 0176 R                        MOV BYTE PTR PIXELH,AL
02F2  E8 01F6 R                        CALL WORDAD
02F5  E8 021B R                        CALL CURTRANSF
02F8  E8 0100 R                        CALL CURSET
02FB  C6 06 010F R 20                  MOV BYTE PTR FIGPRAMS,20H
0300  E8 011A R                        CALL FIGSET
0303  E8 0129 R                        CALL FIGDRAW
0306  E8 0100 R                        CALL CURSET
0309  C6 06 010F R 23                  MOV BYTE PTR FIGPRAMS,23H
030E  E8 011A R                        CALL FIGSET
0311  E8 0129 R                        CALL FIGDRAW
```

The circle will remain on the screen until you press x:

```
0314  E8 00BC R              WAIT3:    CALL GETKEY
0317  3C 78                            CMP AL,'x'
0319  75 F9                            JNE WAIT3
031B  E8 00D4 R                        CALL GEXIT
```

The next example of programming the GDC in your NCR DECISION MATE V gives you the possibility of doing pixel by pixel drawing, by using the keys around the 5 key on the calculator pad situated on the right of the keyboard. Depressing the 8 key will plot one pixel north of the last pixel plotted; depressing the 9 key will plot a pixel north-east of the last pixel plotted, and so on. Pressing the 5 key will effect unplot instead of plot. In this way, you can move the plot position without actually plotting. To see where you are on the screen, press 5 and plot a point. If this is not where you want to be, press 5 again and retrace the last movement to erase the pixel plotted. Enter 0 and then x to leave the program.

The following routine reads the keyboard, and, upon receiving a valid entry 1-9, sets the Direction bits in the first byte of FIGPRAMS accordingly. Note that the numbers on the calculator pad require translation before they can be used as Direction values. The part of the routine at ONOFF (executed if 5 is pressed) executes a GDC Write Data command using the byte stored at WRLOGIC (defined at the beginning of the programming session) as a toggle: if the set Logic is active, then it is replaced by reset Logic, and vice-versa.

```
031E  E8 00BC R        CALCUL:   CALL GETKEY
0321  32 D2                      XOR DL,DL
0323  3C 30                      CMP AL,'0'
0325  74 26                      JE OVER
0327  3C 35                      CMP AL,'5'
0329  74 3D                      JE ONOFF
032B  3C 31                      CMP AL,'1'
032D  74 1F                      JE DIR7
032F  3C 32                      CMP AL,'2'
0331  74 29                      JE DIR0
0333  3C 33                      CMP AL,'3'
0335  74 23                      JE DIR1
0337  3C 34                      CMP AL,'4'
0339  74 15                      JE DIR6
033B  3C 36                      CMP AL,'6'
033D  74 19                      JE DIR2
033F  3C 37                      CMP AL,'7'
0341  74 0F                      JE DIR5
0343  3C 38                      CMP AL,'8'
0345  74 0D                      JE DIR4
0347  3C 39                      CMP AL,'9'
0349  74 0B                      JE DIR3
034B  EB D1                      JMP CALCUL
```

```
0340  C3                  OVER:     RET
034E  FE C2               DIR7:     INC DL
0350  FE C2               DIR6:     INC DL
0352  FE C2               DIR5:     INC DL
0354  FE C2               DIR4:     INC DL
0356  FE C2               DIR3:     INC DL
0358  FE C2               DIR2:     INC DL
035A  FE C2               DIR1:     INC DL
035C  88 16 010F R        DIR0:     MOV BYTE PTR FIGPRAMS,DL
                                             ;Direction in FIGPRAMS
0360  E8 011A R                     CALL FIGSET
0363  E8 0129 R                     CALL FIGDRAW
0366  EB B6                         JMP CALCUL
                          ;
0368  A0 0043 R           ONOFF:    MOV AL,BYTE PTR WRLOGIC
036B  34 01                         XOR AL,1
036D  A2 0043 R                     MOV BYTE PTR WRLOGIC,AL
0370  E8 0004 R                     CALL OUTC
0373  EB A9                         JMP CALCUL
```

For pixel by pixel drawing, the "initial values" stated in the description of the GDC Figure command should be set:

```
0375  00 00 40 08 00 08 FIGPRAM2  DB 0,0,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
      00 FF 3F FF 3F
```

To do this, the program first copies FIGPRAM2 to FIGPRAMS. Set the cursor at CURPRAMS (this time the program does not do this for you) before CURSET is called. The GDC command byte 23H changes the drawing Logic from its initialization setting of "complement" to "set to 1." This means that if lines cross during drawing, pixel erasure will not occur. If this GDC command is omitted, ONOFF will not work properly. The instruction pointer will not leave CALCUL until you press 0. The "complement" setting of the drawing Logic is then restored. The JMP SAVEIT instruction applies to a program extension described later. For the moment, this instruction should read JMP SAVED.

```
0380  8D 1E 0375 R                 LEA BX,FIGPRAM2
0384  8D 3E 010F R                  LEA DI,FIGPRAMS
0388  B1 0B                         MOV CL,0BH
038A  8A 07               NEXTPR2:  MOV AL,BYTE PTR [BX]
038C  88 05                         MOV BYTE PTR [DI],AL
```

```
038E  43                           INC BX
038F  47                           INC DI
0390  FE C9                        DEC CL
0392  75 F6                        JNZ NEXTPR2
0394  E8 0044 R                    CALL GINIT
0397  E8 0100 R                    CALL CURSET
039A  C6 06 0043 R 23              MOV BYTE PTR WRLOGIC,23H
039F  B0 23                        MOV AL,23H
03A1  E8 0004 R                    CALL OUTC
03A4  E8 031E R                    CALL CALCUL
03A7  E8 00BC R          WAIT4:    CALL GETKEY
03AA  3C 78                        CMP AL,'x'
03AC  75 F9                        JNE WAIT4
03AE  C6 06 0043 R 21              MOV BYTE PTR WRLOGIC,21H
03B3  B0 21                        MOV AL,21H
03B5  E8 0004 R                    CALL OUTC
                                                ;resets to complement
                                                ;from any setting.
03B8  E9 439B R                    JMP SAVEIT   ;JMP SAVED
03BB  E8 0004 R          SAVED:    CALL GEXIT
```

The character set of your NCR DECISION MATE V is stored in the ROM which executes power-up initialization. The characters are stored in ascending ASCII sequence from location 1000H onwards. Each character is stored in 16 bytes, representing 16 horizontal line scans. In order to read a portion of the ROM, you must activate Port 11 (Hex), which acts as a ROM-select switch. To switch back to user RAM, Port 10 (Hex) must be activated. While the ROM is selected, the RAM below location 2000H is de-selected. This means that the part of your program which reads the ROM must be located at or above that address. This presents no problem inasmuch as the operating system loads transient programs well above that address. However, you should bear in mind that the 8086 interrupt vector is not accessible while the ROM is selected. This means that INT 21H would cause loss of program control. Therefore, you must de-select the ROM before using MS-DOS function requests. If you are using your own interfaces with peripheral devices and these interfaces make use of interrupts, it is advisable to issue a disable interrupts instruction (CLI) prior to ROM selection.

CHSTORE is to be used for storing the 16-byte character pattern immediately upon being read from the ROM:

```
03BE      10 [            CHSTORE    DB 16 DUP(0)
               00
                  ]
```

The following routine, ASCII, fetches a 16 x 8 bit pattern from the ROM and deposits it in the 16-byte storage area CH-STORE. Upon entry, register AL must contain the ASCII character for which the bit pattern is required. The binary value of the ASCII character is multiplied by 16, the result residing in AX. The start address of the character area in the ROM is added to this, thus BX addresses the first of the 16 bytes containing the bit pattern. These bytes are then copied via register AL to CHSTORE. Note the segment override prefix in the program line containing the ROMBYTE label. This must be included, otherwise the 1000H offset would relate to the beginning of the program area set up by the operating system, and not to the beginning of machine memory.

```
03CE  53                  ASCII:     PUSH BX
03CF  51                             PUSH CX
03D0  52                             PUSH DX
03D1  B2 10                          MOV DL,10H
03D3  F6 E2                          MUL DL          ;code already in AL
                                                     ;at calling.
03D5  05 1000                        ADD AX,1000H     ;address of char
                                                      ;in ROM now in AX.
03D8  8B D8                          MOV BX,AX
03DA  8D 3E 03BE R                   LEA DI,CHSTORE
03DE  B9 0010                        MOV CX,10H
03E1  BA 0000                        MOV DX,0
03E4  8E C2                          MOV ES,DX
03E6  E6 11                          OUT 11H,AL
03E8  26: 8A 07           ROMBYTE:   MOV AL,ES:BYTE PTR [BX]
03EB  88 05                          MOV BYTE PTR [DI],AL
03ED  43                             INC BX
03EE  47                             INC DI
03EF  E2 F7                          LOOP ROMBYTE
03F1  E6 10                          OUT 10H,AL
03F3  5A                             POP DX
03F4  59                             POP CX
03F5  5B                             POP BX
03F6  C3                             RET
```

The following two program lines make a copy of the bit pattern of the number 7:

```
03F7  B0 37                       MOV AL,'7'
03F9  E8 03CE R                   CALL ASCII
```

If you write out the bit pattern contained in CHSTORE, you will see that the least significant bit of each byte contains the leftmost pixel of the line scan for that byte.

The GDC parameter RAM provides a comfortable means of creating your own user-defined graphic symbols. An 8 x 8 pixel design stored in bytes 8-15 of the parameter RAM can be output as often as you wish.

You may find the two following routines useful. The first sets a zoom factor for the CRT representation of the graphic symbol contained in the parameter RAM. This zoom factor (0-15) must be available in the lower four bits of a single byte area, ZOOMFACT.

```
03FC  04                ZOOMFACT  DB 4
                        ;
03FD  B0 46             ZOOM:     MOV AL,46H
03FF  E8 0004 R                   CALL OUTC
0402  8D 1E 03FC R                LEA BX,ZOOMFACT
0406  B2 01                       MOV DL,1
0408  E8 000F R                   CALL OUTP
040B  C3                          RET
```

The second routine, SKEW, produces in CHARMIR a mirror image of each byte of an 8 x 8 design stored in CHARPATT. This design is thus copied "back to front." Furthermore, the byte sequence is inverted.

```
040C     08 [           CHARMIR   DB 8 DUP(0)
               00
                    ]
0414  00 5A 42 7E 3C 24 CHARPATT  DB 0,5AH,42H,7EH,3CH,24H,24H,42H
      24 42
                                                      ;random example
                        ;
041C  8D 1E 0414 R      SKEW:     LEA BX,CHARPATT
0420  83 C3 07                    ADD BX,7         ;BX points to 8th byte
0423  8D 3E 040C R                LEA DI,CHARMIR   ;DI points to first byte
0427  B9 0008                     MOV CX,8
042A  8A 07             NEXTCH:   MOV AL,BYTE PTR [BX]
```

```
042C  E8 0436 R                     CALL MIRROR
                                                  ;to cancel mirror,
                                                  ;replace CALL instruc-
                                                  ;tion by three NOPs.
042F  88 05                         MOV BYTE PTR [DI],AL
0431  4B                            DEC BX
0432  47                            INC DI
0433  E2 F5                         LOOP NEXTCH   ;until all 8 bytes
0435  C3                            RET           ;exchanged and mirrored
                           ;
0436  53                   MIRROR:  PUSH BX       ;the bits of the AL
0437  51                            PUSH CX       ;register are mirrored
0438  52                            PUSH DX       ;around an imaginary
0439  32 F6                         XOR DH,DH     ;axis between bits 3
043B  B2 01                         MOV DL,1      ;and 4. Thus bits 0
043D  B1 01                         MOV CL,1      ;and 7 exchange posit-
043F  8A D8                         MOV BL,AL     ;ions, as do bits 1
0441  32 E4                NEXTSHFT: XOR AH,AH    ;and 6, and so on.
0443  8A C3                         MOV AL,BL
0445  D3 E0                         SHL AX,CL
0447  22 E2                         AND AH,DL
0449  0A F4                         OR DH,AH
044B  D0 E2                         SHL DL,1
044D  80 C1 02                      ADD CL,2
0450  80 F9 11                      CMP CL,11H
0453  75 EC                         JNE NEXTSHFT
0455  8A C6                         MOV AL,DH
0457  5A                            POP DX
0458  59                            POP CX
0459  5B                            POP BX
045A  C3                            RET
```

The CHAROUT routine loads the 8 x 8 pattern contained in CHARMIR into bytes 8-15 of the GDC parameter RAM. Following this, the parameters for the GDC Figure command and the zoom factor are set. The Figure parameters

```
045B  16 07 40 07 00     CHFGPRAM   DB 16H,7,40H,7,0
                                       ;set slant with bit 7 in byte 1
```

indicate in byte 1 that a non-slanting graphics character with initial drawing direction 6 is to be created. Byte 2 contains the number of pixels, minus 1. The only significance to byte 3 is that the graphics bit is set. Bytes 4 and 5 conclude the setting of the graphics character window as 8 x 8 pixels. Command byte 68H finally draws the character, using the magnification factor placed by CHAROUT in ZOOMFACT.

```
0460  B0 78               CHAROUT:   MOV AL,78H     ;starter in pRAM
                                                    ;at parm 8.
0462  E8 0004 R                      CALL OUTC
0465  8D 1E 040C R                   LEA BX,CHARMIR
0469  B2 08                          MOV DL,8
046B  E8 000F R                      CALL OUTP
046E  B0 4C                          MOV AL,4CH     ;figset
0470  E8 0004 R                      CALL OUTC
0473  8D 1E 045B R                   LEA BX,CHFGPRAM
0477  B2 05                          MOV DL,5
0479  E8 000F R                      CALL OUTP
047C  C6 06 03FC R 04                MOV BYTE PTR ZOOMFACT,4
0481  E8 03FD R                      CALL ZOOM
0484  B0 68                          MOV AL,68H     ;draw graphic char
0486  E8 0004 R                      CALL OUTC
0489  C3                             RET
```

You can put these routines together in the following program. The number 7 is copied from the ROM into CHSTORE. The first three and the last four bytes of CHSTORE contain zero, representing line scans for that character in which no pixels are drawn. The number 7, like many characters in the character set, is nine pixels high, so it will not fit into the GDC parameter RAM. In fact, the bottom of the 7 is truncated during the 8-byte transfer from CHSTORE to CHARPATT in this example. You can get around this problem in graphics mode character writing by transmitting the entire 16-byte in two stages to the GDC parameter RAM (this is how your NCR DECISION MATE V uses the GDC for screen writing in the non-graphics mode), or by simply plotting the character pixel by pixel. For user-defined graphics, this additional programming is not necessary, provided that you can fit all the dots (set bits) into the 8 x 8 format. This program writes copies of the character below one another, if you press the r key. The reason for the position of the next copy becomes apparent if you consider the order in which the bits of the parameter RAM are transmitted (see "The Parameter RAM") and the direction set by CHFGPRAM. By way of extending this program, you may wish to include a cursor positioning facility.

```
048A  E8 0044 R                CALL GINIT
048D  E8 0100 R                CALL CURSET
0490  B0 37                    MOV AL,'7'
0492  E8 03CE R                CALL ASCII
0495  8D 1E 03BE R             LEA BX,CHSTORE
0499  43                       INC BX       ;does not copy
049A  43                       INC BX       ;entire character
049B  43                       INC BX
049C  8D 3E 0414 R             LEA DI,CHARPATT
04A0  B9 0008                  MOV CX,8
04A3  8A 07          NEXTCOP:  MOV AL,BYTE PTR [BX]
04A5  88 05                    MOV BYTE PTR [DI],AL
04A7  43                       INC BX
04A8  47                       INC DI
04A9  E2 F8                    LOOP NEXTCOP
04AB  E8 041C R                CALL SKEW
04AE  E8 0460 R      REPEAT:   CALL CHAROUT
04B1  E8 00BC R      WAIT5:    CALL GETKEY
04B4  3C 72                    CMP AL,'r'
04B6  74 F6                    JE REPEAT
04B8  3C 78                    CMP AL,'x'
04BA  75 F5                    JNE WAIT5
04BC  E8 0004 R                CALL GEXIT
```

By altering the parameters for the GDC Figure command and blanking out the CALL SKEW and CALL MIRROR instructions, you can create some interesting effects..

Finally, let us look at an example of reading the graphic display memory. This facility of the GDC enables you to store graphic designs in such a way that they can be reproduced on the screen at a later time. The following routines enable you to copy graphics display memory contents into user memory. Once they are in user memory, you can easily adjust the graphic image, and then re-write to graphic display memory or store on disk. In everyday practice you will probably read and store blocks of GDC memory in multiples of the disk record size. The routines described here read one half of the graphic display memory for a monochrome CRT into user memory. This is to facilitate manipulation of the graphic image. If your NCR DECISION MATE V has a memory greater than 64KB, you can read the entire graphic bit map (32000 bytes). This is impracticable in the 64KB memory if the operating system and the debugging utility are to be retained.

The data areas required:

```
04BF  FF FF              PRAMSR     DB 0FFH,0FFH
04C1  FF FF              RMASK      DB 0FFH,0FFH
04C3  02 08 40 08 00 08  FIGSR      DB 2,8,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
      00 FF 3F FF 3F
04CE  02                 MASKFIG    DB 2
04CF  3E80 [             SCREEN     DB 16000 DUP(0)
                 00
                     ]

434F  FF FF              DUMBYTES   DB 0FFH,0FFH
```

When you have completed a screen drawing using the pixel by pixel drawing facility described earlier in these GDC programming examples, you probably want to save your graphic design. This must be done before your program leaves the graphic mode, as the GEXIT routine sets the graphics display memory to zero. Therefore, you should insert an instruction before or in place of the CALL GEXIT instruction at the end of the pixel by pixel drawing program, in order to jump first to the program which saves your graphic design: JMP SAVEIT.

Before looking at the SAVEIT program, let us consider three routines which govern the GDC commands and parameters required for reading graphic display memory. The READSCRN routine reads eight 16-bit words of graphic display memory (the size of the FIFO buffer) into user memory via the port A1. Before reading each byte, bit zero of the GDC status register is read, in order to check whether a data byte is available. As soon as a byte is read, this bit resets to zero and remains zero until the next data byte is available from the FIFO buffer. The speed of this resetting to zero is sufficiently high to prevent an unwanted second reading of the same data byte. As each byte is read, it is stored at a memory address pointed to by the DI register, and that register is then incremented.

```
4351  51                 READSCRN:  PUSH CX
4352  B9 0008                       MOV CX,8
4355  B2 02              NEXTWORD:  MOV DL,2
4357  E4 A0              READYCHK:  IN AL,0A0H
4359  24 01                         AND AL,1
435B  74 FA                         JZ READYCHK
435D  E4 A1                         IN AL,0A1H
435F  88 05                         MOV BYTE PTR [DI],AL
4361  47                            INC DI
```

```
4362  FE CA                         DEC DL
4364  75 F1                         JNZ READYCHK
4366  E2 ED                         LOOP NEXTWORD
4368  59                            POP CX
4369  C3                            RET
```

FIFOCLR issues the Read Data command to the GDC, thus effecting the FIFO buffer turn-around. You do not have to check whether the FIFO buffer is empty before issuing this command, as any commands and parameters already in the buffer will be dealt with before the Read Data command is actually executed.

```
436A  B0 A0              FIFOCLR:   MOV AL,0A0H
436C  E8 0004 R                     CALL OUTC
436F  C3                            RET
```

Before the Read Data command is issued, you must set up the parameter RAM, and Mask and Figure parameters: bytes 8 and 9 of the parameter RAM and the Mask register must contain FF values to ensure that all bits in the graphic display memory are read; the two significant parameters in FIGSR for the Read Data command are the Direction in the first byte, and the number of words to be read (8, as also specified in READSCRN) in the second byte. The Direction specified is 2 (East), as this enables graphic display memory words to be accessed sequentially without the program overhead of cursor positioning. This means that the first 80 bytes read from the GDC correspond to the top pixel row on the CRT, the next 80 bytes refer to the next pixel row (also reading from left to right), and so on. If you write a program to send screen contents to a printer, you will find it more convenient to set a vertical Direction, thus reading a rectangular area of the screen with each Read Data command.

```
4370  B0 78              SETREAD:   MOV AL,78H
4372  E8 0004 R                     CALL OUTC      ;set pRAM
4375  8D 1E 04BF R                  LEA BX,PRAMSR
4379  B2 02                         MOV DL,2
437B  E8 000F R                     CALL OUTP
437E  B0 4A                         MOV AL,4AH
4380  E8 0004 R                     CALL OUTC      ;set mask
4383  8D 1E 04C1 R                  LEA BX,RMASK
4387  B2 02                         MOV DL,2
4389  E8 000F R                     CALL OUTP
438C  B0 4C                         MOV AL,4CH
```

```
438E  E8 0004 R                   CALL OUTC     ;set fig
4391  8D 1E 04C3 R                LEA BX,FIGSR
4395  B2 0B                       MOV DL,0BH
4397  E8 000F R                   CALL OUTP
439A  C3                          RET
```

You can now put together these routines to read the lower half of the (monochrome) graphics display memory into the 16,000 byte area SCREEN. This corresponds to the top half of the screen.

```
439B  8D 1E 00FD R      SAVEIT:   LEA BX,CURPRAMS
                                              ;reading to start
                                              ;at top left corner
                                              ;of the screen
439F  C7 07 0000                  MOV WORD PTR [BX],0
43A3  43                          INC BX
43A4  C7 07 0000                  MOV WORD PTR [BX],0
43A8  E8 0100 R                   CALL CURSET
43AB  8D 3E 04CF R                LEA DI,SCREEN
43AF  B9 03E8                     MOV CX,03E8H
43B2  E8 4370 R         NEXTSCRN: CALL SETREAD
43B5  E8 436A R                   CALL FIFOCLR
43B8  E8 4351 R                   CALL READSCRN
43BB  E2 F5                       LOOP NEXTSCRN
43BD  E8 00D4 R                   CALL GEXIT
```

Before re-writing your display data to graphics display memory, you might wish to change the data in some way:

```
43C0  E8 4428 R                   CALL ADJUST
```

Leaving such changes aside for the moment, let us first examine a method of writing the 16,000 byte graphic design, now held in main memory, back into the graphics display memory. You have already practised one way of doing this, namely, in the program example of pixel by pixel drawing under keyboard control. The difference is that the keyboard control is replaced by the permanently set Direction 2 (East). In this way, the screen is built up in the sequence in which it was read. This is accomplished by reading SCREEN byte by byte, shifting each bit of each byte through the Carry flag, and setting the drawing Logic to "set to one" or "reset to zero" in accordance with that CPU flag. The NOP instruction is included to facilitate breakpoint setting when you are testing the program with the debugging utility.

```
43C3  8D 1E 0375 R      PAINT:     LEA BX,FIGPRAM2
43C7  8D 3E 010F R                 LEA DI,FIGPRAMS
43CB  B9 000B                      MOV CX,0BH
43CE  8A 07             NEXTPR3:   MOV AL,BYTE PTR [BX]
43D0  88 05                        MOV BYTE PTR [DI],AL
43D2  43                           INC BX
43D3  47                           INC DI
43D4  E2 F8                        LOOP NEXTPR3
43D6  C6 06 010F R 02              MOV BYTE PTR FIGPRAMS,2
                                               ;Direction East
43DB  E8 0044 R                    CALL GINIT
43DE  E8 0100 R                    CALL CURSET
43E1  E8 011A R                    CALL FIGSET
43E4  8D 3E 04CF R                 LEA DI,SCREEN
43E8  B9 3E80                      MOV CX,3E80H
43EB  51                NEWBYTE:   PUSH CX
43EC  B9 0008                      MOV CX,8
43EF  8A 25                        MOV AH,BYTE PTR [DI]
43F1  D0 EC             CHECKBIT:  SHR AH,1
43F3  72 05                        JC PLOT       ;check Carry,
43F5  B0 22                        MOV AL,22H    ;set drawing Logic
43F7  EB 03 90                     JMP LOGICSET  ;accordingly
43FA  B0 23             PLOT:      MOV AL,23H
43FC  E8 0004 R         LOGICSET:  CALL OUTC
43FF  B0 4C                        MOV AL,4CH    ;command to set
4401  E8 0004 R                    CALL OUTC     ;Figure parameters
4404  8D 1E 010F R                 LEA BX,FIGPRAMS
4408  B2 03                        MOV DL,3      ;set first three
440A  E8 000F R                    CALL OUTP     ;Figure parameters.
440D  B0 6C                        MOV AL,6CH    ;Draw command
440F  E8 0004 R                    CALL OUTC
4412  E2 DD                        LOOP CHECKBIT
4414  47                           INC DI
4415  59                           POP CX
4416  E2 D3                        LOOP NEWBYTE
4418  B0 21                        MOV AL,21H    ;restore
441A  E8 0004 R                    CALL OUTC     ;Complement Logic
441D  E8 00BC R         WAIT6:     CALL GETKEY
4420  3C 78                        CMP AL,'x'
4422  75 F9                        JNE WAIT6
4424  E8 0004 R                    CALL GEXIT
4427  90                           NOP
```

The following routine shows just two of many possibilities of altering the graphic image while it is stored in main memory. You can construct a vector from which one of a number of alteration routines can be activated, according to keyboard input.

```
4428  E8 00BC R         ADJUST:    CALL GETKEY
442B  3C 00                         CMP AL,0
442D  74 F9                         JE ADJUST
442F  3C 69                         CMP AL,'i'    ;pixel inversion
4431  74 05                         JE ADJUST1
4433  3C 6D                         CMP AL,'m'    ;mirror image
4435  74 13                         JE ADJUST2
4437  C3                            RET
```

The two possibilities envisaged here are the inversion (bit complementing) of the screen image, and the production of a mirror image. The inversion routine simply uses the 8086 instruction to produce the one's complement of a register. The effect is the same as writing all ones with complement Logic into the graphics display memory.

```
4438  8D 3E 04CF R      ADJUST1:   LEA DI,SCREEN
443C  B9 1F40                      MOV CX,1F40H
443F  8B 05             ADJUST11:  MOV AX,WORD PTR [DI]
4441  F7 D0                        NOT AX
4443  89 05                        MOV WORD PTR [DI],AX
4445  47                           INC DI
4446  47                           INC DI
4447  E2 F6                        LOOP ADJUST11
4449  C3                           RET
```

The mirror routine (ADJUST2) regards SCREEN as 200 "lines," each containing 80 bytes (= 640 bits for one display line). Each line is turned "back to front." Following this, the same is done within each byte, using the MIRROR routine described earlier. Thus, an arrow which previously pointed left, will now point to the right when the contents of SCREEN are re-written to graphics display memory.

```
444A  8D 1E 04CF R      ADJUST2:   LEA BX,SCREEN
444E  4B                           DEC BX
444F  B9 00C8                      MOV CX,200
4452  51                NEXTLINE:  PUSH CX
4453  B9 0028                      MOV CX,40
```

```
4456  BF 0028                    MOV DI,40
4459  BE 0029                    MOV SI,41
                                           ;starting from center of
                                           ;line working outwards,
                                           ;exchange byte pairs
                                           ;within that line
445C  8A 01             LINESWOP: MOV AL,BYTE PTR [BX+DI]
445E  8A 20                      MOV AH,BYTE PTR [BX+SI]
4460  88 21                      MOV BYTE PTR [BX+DI],AH
4462  88 00                      MOV BYTE PTR [BX+SI],AL
4464  4F                         DEC DI
4465  46                         INC SI
4466  E2 F4                      LOOP LINESWOP
4468  59                         POP CX
4469  83 C3 50                   ADD BX,80
446C  E2 E4                      LOOP NEXTLINE
                                         ;point to next line of
                                         ;stored screen
446E  8D 3E 04CF R               LEA DI,SCREEN
4472  B9 3E80                    MOV CX,3E80H
4475  8A 05             ADJUST21: MOV AL,BYTE PTR [DI]
4477  E8 0436 R                  CALL MIRROR
                                         ;mirror byte within itself
447A  88 05                      MOV BYTE PTR [DI],AL
447C  47                         INC DI
447D  E2 F6                      LOOP ADJUST21
447F  C3                         RET
```

You are probably asking yourself why the screen writing takes so much time. There are two factors to be considered. First, the program described above does a complete write operation, in the sense that each pixel is addressed, irrespective of whether it is to be turned on or not. The fast method of drawing a figure on the screen is to store and output the coordinates and other parameters which relate solely to the pixels to be plotted, and to make use of the GDC's figure drawing capabilities (line, arc, etc.). This is how the square and circle were drawn in the earlier examples. In fact, you can draw many more figures, and the drawing process will still appear to be instantaneous. The second factor regarding the speed of the screen write is that the Figure parameters have to be restored for each pixel.

There are two other methods of screen writing in the graphics mode, both of which give improved performance. One method is to load the parameter RAM with one 8 x 8 pixel pattern after another. This creates some additional program overhead for cursor positioning. For this reason, the following method is worth considering:

```
4480  E8 4428 R                  CALL ADJUST
4483  E8 0044 R                  CALL GINIT
4486  B0 4C                      MOV AL,4CH  ;command to set
4488  E8 0004 R                  CALL OUTC   ;Figure parameters
448B  8D 1E 04CE R               LEA BX,MASKFIG
448F  B2 01                      MOV DL,1    ;only one required
4491  E8 000F R                  CALL OUTP
4494  E8 0100 R                  CALL CURSET
4497  8D 1E 04CF R               LEA BX,SCREEN
449B  8D 3E 434F R               LEA DI,DUMBYTES
449F  B9 1F40                    MOV CX,1F40H
44A2  B0 4A           NEXTMASK:  MOV AL,4AH  ;command to load
44A4  E8 0004 R                  CALL OUTC   ;Mask with word of
44A7  B2 02                      MOV DL,2    ;stored screen contents
44A9  E8 000F R                  CALL OUTP
44AC  B0 23                      MOV AL,23H  ;write to screen with
44AE  E8 0004 R                  CALL OUTC   ;"set" Logic ...
44B1  87 DF                      XCHG BX,DI  ;(BX now points to
                                             ; DUMBYTES)
44B3  B2 02                      MOV DL,2    ;... setting all bits
44B5  E8 000F R                  CALL OUTP   ;that are set in Mask
44B8  4B                         DEC BX
44B9  4B                         DEC BX
44BA  87 DF                      XCHG BX,DI  ;BX now points to next
                                             ;screen word,
                                             ;DI to DUMBYTES
44BC  E2 E4                      LOOP NEXTMASK
44BE  E8 00BC R       WAIT7:     CALL GETKEY
44C1  3C 78                      CMP AL,'x'
44C3  75 F9                      JNE WAIT7
44C5  E8 0004 R                  CALL GEXIT
44C8  90                         NOP
                      ;
44C9                  CSEG ENDS
```

As before, the writing Direction should be set to 2 (East), thus enabling sequential writing without the need to position the cursor, beyond initially specifying the top left corner (check CURPRAMS). This program loads the Mask register word by word with the contents of SCREEN. The Write Data command is transmitted to the GDC with all its parameter bits set. This means that the 16-bit pattern contained in the Mask register appears as a horizontal pattern of data on the screen in one write cycle. There is no need to repeat the Figure parameter setting. By altering the initial cursor position, you can address different parts of the screen.

COLOR GRAPHICS

The discussion of the GDC and the programming examples so far have dealt with graphics on a monochrome CRT. If your NCR DECISION MATE V has a color CRT, you can make full use of color in the graphics as well as the non-graphics mode. For this purpose, the graphic display RAM has a capacity of 96 KB, instead of the 32 KB RAM used by monochrome CRTs. Even the larger RAM area lies well within the addressing capability of the GDC.

Whereas color in the non-graphic mode is stored in the video attribute byte belonging to each 16 x 8 character area of the graphic display RAM, the graphic mode requires the use of three separate areas corresponding to the green, red, and blue guns of the color CRT. Therefore, your graphics programs must influence not just one, but three bit maps, if you wish to make full use of the color range. The bit maps start at 32 KB boundaries in the 96 KB graphic display memory. Even if you wish to confine pixel writing and drawing to green on black (the first 32 KB govern the green gun, the next 32 KB the red gun, and the last 32 KB the blue gun), you must adapt your graphics initialization routine to reset bits in all three maps. This ensures that the screen is black. Failure to do so may produce intermittent splashes of red and blue. To address the red and blue bit maps, simply increase the cursor address (as used in the graphics initialization routine) by 32 KB and 64 KB respectively.

Apart from this, all you have to remember is that each Draw and Draw Graphics Character command must be repeated once or twice, or not at all, according to the color effect desired.

APPENDIX C

THE MS-DOS I/O PROGRAM (Version 1)

LOADER

```
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;
;               BOOT LOADER FOR MS-DOS VER. 2.0
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;
;
;
;      THIS MODULE MUST BE RECORDED BY "FORMAT" ON EACH MS-DOS
;      DISKETTE IN THE FIRST SECTOR (512 BYTES).
;
;      IT CONSISTS OF A BIOS PARAMETER BLOCK (BPB) DESCRIBING
;      THE MEDIA, AND CODE REQUIRED TO LOAD THE MS-DOS OPERATING
;      SYSTEM MODULES IO.SYS AND MSDOS.SYS.
;
;      IO.SYS AND MSDOS.SYS HAVE TO BE THE FIRST TWO FILES ON
;      EACH SYSTEM DISKETTE.
;
;      THIS MODULE IS ENTERED FROM FIRMWARE AFTER THE FIRST THREE
;      TRACKS HAVE BEEN READ TO MEMORY STARTING AT LOCATION
;      0000:2000.
;
;      MEMORY ALLOCATION AFTER FIRMWARE BOOTING:
;      ------------------------------------
;
;      address         contents
;
;      0000:2000       LOADER
;      0000:2200       FAT
;      0000:2600       ROOT DIRECTORY     (1st sector) FORMATS FE AND FF
;      0000:2A00       ROOT DIRECTORY     (1st sector) FORMATS FC AND FD
;
;
;      FIRMWARE TRANSFERS CONTROL TO THIS MODULE VIA A JUMP FAR
;      0000:2000.
;      CS, DS, ES, SS ARE ALL SET TO ZERO.
;
;
;      ADDRESS 0000:FE06 CONTAINS A MEMORY SIZE ID BYTE
;      ------------------------------------------
;
;      id byte         memory size
;
;      00 hex          64   kbytes
```

```
LOADER.

;       01 hex          128  kbytes
;       02 hex          192  kbytes
;       03 hex          256  kbytes
;       04 hex          320  kbytes
;       05 hex          384  kbytes
;       06 hex          448  kbytes
;       07 hex          512  kbytes
;               etc.
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
```

```
                    ;
                    ;***********************
                    ;***  DISK CONSTANTS  ***
                    ;***********************
                    ;
                    ;
0040                DENSITY EQU     40H             ; DOUBLE DENSITY BIT (MFM)
0080                MULTTRK EQU     80H             ; MULTY TRACK READING
0002                BYTSEC  EQU     2               ; BYTES PER SECTOR (N)
001B                GPL     EQU     1BH             ; GAP LENGTH
                                                    ;
00F6                PATTERN EQU     0F6H            ; FORMAT PATTERN
                                                    ;
0005                INNER   EQU     5               ; Number of inner retries
                    ;
                    ;
                    ;
                    ;******************
                    ;***  I/O PORTS  ***
                    ;******************
                    ;
                    ;
                    ; FDC
                    ; ---
                    ;
0051                DCOMD   EQU     51H             ; DISK COMMAND PORT
0050                DSTAT   EQU     50H             ; DISK STATUS PORT
0051                FDCRA   EQU     51H             ; READ DATA FROM FDC PORT
                    ;
                    ;
                    ;
                    ; DMA
                    ; ---
                    ;
002A                DMAMB   EQU     2AH             ; WRITE SINGLE MASK REGISTER BIT
002B                DMAMO   EQU     2BH             ; DMA MODE PORT
0026                COAD    EQU     26H             ; DMA ADDR PORT
0027                COTC    EQU     27H             ; DMA LENGTH PORT
                    ;
                    ;
                    ;
                    ; SYSTEM STATUS
                    ; -------------
                    ;
0013                SYSSTA  EQU     13H             ; SYSTEM STATUS PORT
                    ;
                    ;
                    LOADER.



                    ;
                    ;*********************
                    ;***  DMA COMMANDS  ***
                    ;*********************
                    ;
                    ;
0047                DMAWRT  EQU    47H          ; WRITE DMA COMMAND
                    ;
                    ;
                    ;*********************
                    ;***  FDC COMMANDS  ***
                    ;*********************
                    ;
                    ;
0002                READTRK EQU    02H          ; READ TRACK COMMAND
0005                WRITDAT EQU    05H          ; WRITE DATA COMMAND
0006                READDAT EQU    06H          ; READ DATA COMMAND
0007                RESTORE EQU    07H          ; RESTORE COMMAND
0008                FDCSIS  EQU    08H          ; SENSE INTERRUPT STATUS
000A                IDREAD  EQU    0AH          ; READ ID COMMAND
000D                WRITFMT EQU    0DH          ; FORMAT A TRACK
000F                SEEKTRK EQU    0FH          ; SEEK A TRACK
                    ;
                    ;*********************
                    ;*** LOADER EQUATES ***
                    ;*********************
                    ;
0000                BIOSOFF        EQU   0000H  ;IP AT BIOS ENTRY
0040                BIOSSEG        EQU   400H/16 ;CS AT BIOS ENTRY
                    ;
0000                Z80            EQU   00OH   ;PROCESSOR SWITCH
0010                RAM            EQU   10H    ;SWITCH IN RAM
0011                ROM            EQU   11H    ;SWITCH TO ROM
0041                KEYST          EQU   41H    ;KEYBOARD STATUS PORT
0001                KEYSTROKE      EQU   01H    ;IF TRUE ANY KEY HAS BEEN DEPRESSED
0001                INVDISP        EQU   01H    ;INVERSE VIDEO
0000                NORMAL         EQU   00H
                    ;
E000                REAL_ADDRESS   EQU   0E000H ;ADDRESS OF RELOCATED LOADER
                    ;
```

```
                        LOADER.


                        ;
                                CSEG
                                ORG     2000H
                        ;

2000 E91B00     201E            JMP     START           ;SKIP BPB
                        ;
2003 313642495420       OS_ID   DB      '16BIT          ;THIS IDENTIFIER IS NEEDED BY FIRMWARE
     2020
                                                        ;TO INITIALIZE 16-BIT PROCESSOR
                        ;
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;--------------------------- B P B -------------------------------
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;
                        ;       FORMAT........................FD     FF     FC     FE
                        ;
200B 0002               BYTES_PER_SECTOR                DW      512    ;512   512    512
200D 02                 SECTORS_PER_ALLOCATION_UNIT     DB      2      ;2     1      1
200E 0100               RESERVED_SECTORS                DW      1      ;1     1      1
2010 02                 NUMBER_OF_FATS                  DB      2      ;2     2      2
2011 7000               NUMBER_OF_ROOT_DIR_ENTRIES      DW      112    ;112   64     64
2013 D002               NUMBER_OF_SECTORS_IN_LOG_IMAGE  DW      720    ;640   360    320
2015 FD                 MEDIA_DESCRIPTOR                DB      0FDH   ;0FFH  0FCH   0FEH
2016 0200               NUMBER_OF_FAT_SECTORS           DW      2      ;1     2      1
                        ;
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;
                        ;                                       FD     FF     FC     FE
                        ;
2018 0900               NUMBER_OF_SECTORS_PER_TRACK     DW      9      ;8     9      8
201A 0200               NUMBER_OF_HEADS                 DW      2      ;2     1      1
201C 0000               NUMBER_OF_HIDDEN_SECTORS        DW      0      ;0     0      0
                        ;
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;------------------------------------------------------------------
                        ;
                        ; DEFAULT FORMAT PRODUCED BY FORMAT UTILITY IS FD, OTHER FORMATS ARE
                        ; PRODUCED ACCORDING TO SWITCH SETTING.
                        ; WHEN THE LOADER IS TRANSFERRED TO A NEWLY FORMATTED DISK BY FORMAT
                        ; THE BPB ENTRIES ARE UPDATED ACCORDING TO SELECTED FORMAT.
                        ;
```

```
                              LOADER.



                              ;
                              ;       RELOCATE LOADER TO FREE LOW MEMORY FOR SYSTEM
                              ;
                              START:
201E FC                                 CLD                                ;SET INCREMENT
201F B9C801                             MOV     CX,LOADER_LENGTH           ;LENGTH TO MOVE
2022 BE3320                             MOV     SI,OFFSET LOADER           ;START AT LOADER:
2025 BF33E0                             MOV     DI,REAL_ADDRESS + (OFFSET LOADER - 2000H) ;DESTINATION
2028 F3A4                          REP  MOVSB                              ;MOVE BYTE STRING
202A B8000C                             MOV     AX,(REAL_ADDRESS - 2000H)/16
202D 1E                                 PUSH    DS                         ;SAVE DATA SEGMENT
202E 8ED8                               MOV     DS,AX                      ;ADJUST DS FOR RELOCATED DSEG
2030 E900C0             E033            JMP     LOADER + (REAL_ADDRESS - 2000H) ;START LOADER
                              ;
                              LOADER:
2033 26A00022                           MOV     AL,ES:FAT                  ;GET FAT ID
2037 8AD8                               MOV     BL,AL                      ; COPY TO BL
2039 2401                               AND     AL,01H                     ;TEST DOUBLE DIDEDNESS
203B 7505               2042            JNZ     F0                         ; JUMP IF DOUBLE SIDED
203D C606F02146                         MOV     CONSTR9,46H                ;MFM BUT NOT MT
                              F0:
2042 A20022                             MOV     HEAD,AL                    ;SET HEAD FOR START READING
2045 3401                               XOR     AL,01H                     ;FLIP HEAD BIT
2047 A2FF21                             MOV     CYLMODE,AL                 ;HEAD 1 = CYLMODE 0
204A FEC3                               INC     BL                         ;TEST FOR FAT-ID FF
204C 7509               2057            JNZ     F1                         ; JUMP IF NOT
204E C606FA2103                         MOV     SECTOR,3                   ;FF..START AT SIDE 1 TO S3
2053 FE0EF621                           DEC     SECTRK                     ;SET 8 SECTORS PER TRACK
                              F1:
2057 BF0026                             MOV     DI,ES:OFFSET IO            ;BEGIN OF ROOT DIRECTORY
205A FEC3                               INC     BL                         ;TEST FOR FAT-ID FE
205C 7509               2067            JNZ     F2                         ; JUMP IF NOT FE
205E C606FA2108                         MOV     SECTOR,8                   ;FE..START AT SIDE 0 TO S8
2063 FE0EF621                           DEC     SECTRK                     ;SET 8 SECTORS PER TRACK
                              F2:
2067 FEC3                               INC     BL                         ;TEST FOR FAT-ID FD
2069 740A               2075            JZ      F3                         ; JUMP IF FD (DEFAULT)
206B FEC3                               INC     BL                         ;TEST FOR FAT-ID FC
206D 7509               2078            JNZ     F4                         ; JUMP IF NOT
206F C706F9210101                       MOV     WORD PTR TRACK,0101H       ;START AT SIDE 0 T1 S1
                              F3:
2075 BF002A                             MOV     DI,ES:OFFSET IOFCFD        ;ROOT DIRECTORY FOR FD AND FC
                              F4:
2078 B90B00                             MOV     CX,11                      ;LENGTH OF FILENAME IN DIR
207B BEE021                             MOV     SI,OFFSET IOSYS            ;STRING "IO      SYS"
207E F3A6                          REPE CMPSB                              ;COMPARE WHILE EQUAL
2080 750F               2091            JNZ     BOOT_ERROR                 ;JUMP IF NO IO.SYS FILE ON DISK
                              ;
                              ;
                              ;AT THIS POINT WE KNOW BOTH, IO.SYS AND MSDOS.SYS EXISTS ON THIS DISK
                              ;
```

```
                         LOADER.

                         ; WE'VE SETUP ALL PARAMETERS REQUIRED BY THE FD PIM TO LOAD
                         ; THE SYSTEM.
                         ;
                         ;START LOADING NOW
                         ;
2082 E83800       20BD          CALL    DREAD                   ;MULTIPLE SECTOR READ ROUTINE
                         ;
                         ;CHECK IF SYSTEM LOADED SUCCESSFULLY
                         ;
2085 F606FE21FF                 TEST    CNTR,0FFH               ;RETRY COUNTER )0 INDICATES SUCCESS
208A 7405         2091          JZ      BOOT_ERROR              ;JUMP IF NOT OK
                         ;
                         ;THE OPERATING SYSTEM HAS BEEN LOADED SUCCESSFULLY
                         ;EXIT VIA JUMP FAR TO BIOS ROUTINE (BEGINNING OF IO.SYS)
                         ;
208C EA                         DB      0EAH    ;JUMP FAR
208D 0000                       DW      BIOSOFF ;OFFSET BIOS START
208F 4000                       DW      BIOSSEG ;BIOS SEGMENT
                         ;
```

```
                       LOADER.



                       ;
                       ;***********************************************
                       ;***********************************************
                       ;
                       ; DISPLAY ERROR MESSAGE USING ROM ROUTINE AND
                       ; WAIT FOR KEYSTROKE TO START OVER AGAIN
                       ;
                       ;***********************************************
                       ;***********************************************
                       ;
                       ;
                       ;
                       BOOT_ERROR:
                       ;
2091 BBCBE1                    MOV     BX,OFFSET NONSYS + (REAL_ADDRESS - 2000H)
2094 1F                        POP     DS                      ;RESTORE DATA SEGMENT
2095 C6060BF900                MOV     INVERS,NORMAL           ;INVERS VIDEO
209A C70600FB0017              MOV     CURSXY,1700H            ;LINE 24 1st COLUMN
                       ;
                       ;CALL DISPLAY ROUTINE VIA INTERSEGMENT CALL
                       ;
20A0 9A                CALLFAR         DB  9AH                 ;CODE FOR INTERSEGMENT CALL
20A1 0000              DISPOFF         DW  0000H               ;OFFSET OF DISPLAY ROUTINE
20A3 00FF              DISPSEG         DW  OFF00H              ;CS MUST BE SET TO FF00(0)HEX
                       ;
                       ;DISPLAY ROUTINE RETURNS HERE VIA INTERSEGMENT RETURN
                       ;
                       WAITX:
20A5 E441                      IN      AL,KEYST                ;TEST FOR KEYSTROKE
20A7 DDE8                      SHR     AL,1            ;KEYSTROKE SETS BIT 0
20A9 73FA         20A5         JNB     WAITX           ;LOOP UNTIL KEY DEPRESSED
20AB'C60611F1C3                MOV     JMP_CODE,0C3H   ;SET Z80 JUMP CODE
20B0 C70612F10C01              MOV     JMP_ADD,010CH   ;ADDRESS JUMP (BOOT_LOADER IN FW)
20B6 E611                      OUT     ROM,AL                  ;SWITCH TO ROM
20B8 E6D0                      OUT     Z80,AL          ;ACTIVATE Z80...Z80 INSTRUCTION POINTER
                                                       ; SET TO F111H
20BA E9613F       601E         JMP     START - REAL_ADDRESS +2000H     ;ABSOLUTE 0000:2018
                       ;
                       ;16-BIT MSDOS LOADER STARTED OVER AGAIN
                       ;
```

LOADER.

```
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;    ROUTINE NAME:       DREAD
;
;
;    FUNCTION:           DREAD - low level READ DATA
;
;
;    ENTRY VIA:          CALL
;
;
;    ENTRY CONDITIONS:   Following variables are set:
;                        CYLMODE, HEAD, TRACK, SECTOR,
;                        SECCNT (Number of sectors),
;
;    EXIT VIA:           RETURN
;
;    EXIT CONDITIONS:    STATUS (returned in ERRBUF)
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;
;
```

```
                      LOADER.


                      DREAD:
                                                    ; Check track conflict
                                                    ; ------------------
20BD 8A1EF621                 MOV    BL,SECTRK      ; BL <-- SECTORS PER TRACK +1
20C1 FEC3                     INC    BL
20C3 2A1EFA21                 SUB    BL,SECTOR      ; BL = remaining sectors in track
                                                    ;
20C7 A0FF21                   MOV    AL,CYLMODE     ; If CYLINDER MODE
20CA 0A060022                 OR     AL,HEAD        ; and HEAD 0
20CE 7504        20D4         JNZ    I02            ;
20D0 021EF621                 ADD    BL,SECTRK      ; then add sectors of corresponding track
                      I02:                          ;
20D4 3A1EFB21                 CMP    BL,SECCNT      ; Check if sectors fit in track
20D8 7204        20DE         JB     I03            ; Jump if sectors don't fit in track
                                                    ;
20DA 8A1EFB21                 MOV    BL,SECCNT      ;
                      I03:                          ; BL - number of sectors for I/O
20DE 281EFB21                 SUB    SECCNT,BL      ; SECCNT <-- remaining sectors for next I/O
                                                    ; (0 if sectors fit in track)
20E2 B700                     MOV    BH,0
20E4 A1EB21                   MOV    AX,SECSIZ      ; SECTOR SIZE
20E7 F7E3                     MUL    BX             ;
20E9 8BD0                     MOV    DX,AX          ; DMA LENGTH
20EB C606FE2105               MOV    CNTR,INNER     ; CNTR <-- Inner retry counter
                      I04:                          ;
20F0 E85800      214B         CALL   DSEEK          ; First do low level SEEK A TRACK
                                                    ;
20F3 A00022                   MOV    AL,HEAD        ;
20F6 D0E0                     SHL    AL,1           ;
20F8 D0E0                     SHL    AL,1           ;
20FA A2F121                   MOV    COMSTR9+1,AL   ;                  <-- DRIVE & HEAD
20FD A0F921                   MOV    AL,TRACK       ;
2100 A2F221                   MOV    COMSTR9+2,AL   ;                  <-- TRACK
2103 A00022                   MOV    AL,HEAD        ;
2106 A2F321                   MOV    COMSTR9+3,AL   ;                  <-- HEAD
2109 A0FA21                   MOV    AL,SECTOR      ;
210C A2F421                   MOV    COMSTR9+4,AL   ;                  <-- SECTOR
210F E89E00      21B0         CALL   DMA            ; Initialize DMA
2112 B90900                   MOV    CX,9           ; Length of command string
2115 BBF021                   MOV    BX,OFFSET COMSTR9 ; BX points to command string
2118 E86100      217C         CALL   XWAIT          ; Send COMMAND STRING to FDC
211B E86900      2187         CALL   GETBYT         ; Get STATUS BYTES
211E 7407        2127         JZ     ERROR
2120 F6060122C0               TEST   ERRBUF,0C0H    ; Test for normal termination
2125 7407        212E         JZ     EXIT           ; Exit loop if normal termination
                                                    ;
2127 FE0EFE21         ERROR:  DEC    CNTR           ; Decrement retry counter
212B 75C3        20F0         JNZ    I04            ; Do retries
212D C3                       RET
                                                    ;
                      EXIT:                         ;
```

```
                                               ; Set variables for next I/O dep. on CYL MODE
                                               ; ------------------------------------------
212E F606FB21FF                TEST   SECCNT,0FFH  ;check if another I/O is necessary
2133 7501         2136         JNZ    I06          ; Jump if necessary
2135 C3                        RET                 ; Return if I/O complete
                        I06:                       ;
2136 0116FC21                  ADD    DMAADDR,DX   ; SET DMA ADDRESS FOR NEXT TRACK/CYLINDER
213A FE06F921                  INC    TRACK        ; NEXT TRACK OR CYLINDER
213E C606FA2101                MOV    SECTOR,1     ; Set SECTOR to begin of track
2143 C606002200                MOV    HEAD,0       ; set HEAD 0
2148 E972FF       20BD         JMP    DREAD        ; READ NEXT TRACK OR CYLINDER
```

```
                  LOADER.


                  ;*******************************************************************
                  ;*******************************************************************
                  ;*******************************************************************
                  ;
                  ;    ROUTINE NAME:      DSEEK
                  ;
                  ;    FUNCTION:          Low level SEEK A TRACK
                  ;
                  ;    ENTRY VIA:         CALL
                  ;
                  ;    ENTRY CONDITIONS:  Following variables are set:
                  ;
                  ;                       DRIVE, HEAD, and TRACK
                  ;    EXIT VIA:          RETURN
                  ;
                  ;    EXIT CONDITIONS:   SEEK complete
                  ;
                  ;*******************************************************************
                  ;*******************************************************************
                  ;*******************************************************************
                  ;
                  ;
                  ;
                  DSEEK:                          ; Set up COMMAND STRING
                                                  ; -------------------
2148 A00022                 MOV    AL,HEAD        ;
214E D0E0                   SHL    AL,1           ;
2150 D0E0                   SHL    AL,1           ;
2152 A2EE21                 MOV    COMSTR3+1,AL   ;             <-- DRIVE & HEAD
2155 A0F921                 MOV    AL,TRACK       ;
2158 A2EF21                 MOV    COMSTR3+2,AL   ;             <-- TRACK
215B B90300                 MOV    CX,3           ; Length of command string
215E 8BE021                 MOV    BX,OFFSET COMSTR3
2161 E81800        217C     CALL   XWAIT          ; Send COMMAND STRING to FDC
                  DSEEK1:                         ;
2164 E413                   IN     AL,SYSSTA      ; Wait for interrupt
2166 A808                   TEST   AL,08          ; Test DISK INTERRUPT BIT
2168 74FA          2164     JZ     DSEEK1         ; jump if no interrupt
216A E83C00        21A9     CALL   FDCRDY
216D 8008                   MOV    AL,FDCSIS      ;Sense interrupt status command
216F E651                   OUT    DCOMD,AL
2171 E83500        21A9     CALL   FDCRDY
2174 E451                   IN     AL,FDCRA
2176 E83000        21A9     CALL   FDCRDY
2179 E451                   IN     AL,FDCRA
217B C3                     RET
                  ;
```

LOADER.

```
                        ;*************************************************************************
                        ;*************************************************************************
                        ;*************************************************************************
                        ;
                        ;    ROUTINE NAME:      XWAIT
                        ;
                        ;    FUNCTION:          Send COMMAND STRING to FDC
                        ;
                        ;    ENTRY VIA:         CALL
                        ;
                        ;    ENTRY CONDITIONS:  CX - LENGTH OF STRING
                        ;                       BX - OFFSET OF COMMAND STRING
                        ;
                        ;    EXIT VIA:          RETURN
                        ;
                        ;    EXIT CONDITIONS:   NONE
                        ;
                        ;
                        ;*************************************************************************
                        ;*************************************************************************
                        ;*************************************************************************
                        ;
                        ;
                        ;
                        XWAIT:
217C E82A00     21A9            CALL    FDCRDY          ; Wait until FDC is ready
217F 8A07                       MOV     AL,BYTE PTR [BX]; AL <-- next COMMAND STRING byte
2181 E651                       OUT     DCOMD,AL        ; Send byte to FDC
2183 43                         INC     BX
2184 E2F6       217C            LOOP    XWAIT           ; Loop until last byte
2186 C3                         RET                     ;
                                                        ;
```

```
                       LOADER.


                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;
                       ;    ROUTINE NAME:       GETBYT
                       ;
                       ;    FUNCTION:           Get STATUS BYTES into ERRBUF
                       ;
                       ;    ENTRY VIA:          CALL
                       ;
                       ;    ENTRY CONDITIONS:   NONE
                       ;
                       ;    EXIT VIA:           RETURN
                       ;
                       ;    EXIT CONDITIONS:    Z-FLAG SET IF NOT READY
                       ;
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;
                       ;
                       ;
                       GETBYT:                         ;
2187 E413                      IN      AL,SYSSTA
2189 A804                      TEST    AL,04
218B 741B         21A8         JZ      XE              ; EXIT IF NOT READY
218D E450                      IN      AL,DSTAT
218F A880                      TEST    AL,80H
2191 74F4         2187         JZ      GETBYT          ; LOOP IF NO MASTER REQUEST
2193 B007                      MOV     AL,07H          ; DISABLE FDC CHANNEL
2195 E62A                      OUT     DMAMB,AL
2197 B90700                    MOV     CX,07           ; CX <-- Length of ERROR BUFFER
219A BB0122                    MOV     BX,OFFSET ERRBUF; BX <-- Addr of ERROR BUFFER
                       GETBYT1:                        ;
219D E451                      IN      AL,FDCRA        ; Read STATUS BYTE from FDC
219F 8807                      MOV     BYTE PTR [BX],AL; into ERROR BUFFER
21A1 43                        INC     BX              ;
21A2 E80400       21A9         CALL    FDCRDY          ; Wait until FDC is ready
21A5 E2F6         219D         LOOP    GETBYT1         ; Loop until last byte (7 times)
21A7 41                        INC     CX              ; SET NON ZERO
21A8 C3                XE:     RET
```

LOADER.

```
                                 ;*****************************************************************************
                                 ;*****************************************************************************
                                 ;*****************************************************************************
                                 ;
                                 ;    ROUTINE NAME:       FDCRDY
                                 ;
                                 ;    FUNCTION:           Wait until FDC is ready
                                 ;
                                 ;    ENTRY VIA:          CALL
                                 ;
                                 ;    ENTRY CONDITIONS:   NONE
                                 ;
                                 ;    EXIT VIA:           RETURN
                                 ;
                                 ;    EXIT CONDITIONS:    NONE
                                 ;
                                 ;*****************************************************************************
                                 ;*****************************************************************************
                                 ;*****************************************************************************
                                 ;
                                 ;
                                 ;
                                 FDCRDY:                                ;
21A9 E450                                IN      AL,DSTAT               ; AL <-- MAIN STATUS REGISTER
21AB A880                                TEST    AL,80H                 ; Test MASTER REQUEST BIT
21AD 74FA           21A9                 JZ      FDCRDY                 ; Jump if no MASTER REQUEST (means: in execution)
                                                                        ;
21AF C3                                  RET                            ; Return if FDC is ready
```

```
                       LOADER.


                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;
                       ;    ROUTINE NAME:       DMA    (FDC TO MEMORY)
                       ;
                       ;    FUNCTION:           DMA write
                       ;
                       ;    ENTRY VIA:          CALL
                       ;
                       ;    ENTRY CONDITIONS:   DX - DMA LENGTH
                       ;
                       ;    EXIT VIA:           RETURN
                       ;
                       ;    EXIT CONDITIONS:    none
                       ;
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;*******************************************************************************
                       ;
                       ;
                       ;
                       DMA:                             ;
21B0 B047                      MOV     AL,DMAWRT
21B2 E62B                      OUT     DMAMO,AL         ; OUT MODE
                                                        ;
21B4 A1FC21                    MOV     AX,DMAADDR       ; ABSOLUTE ADDRESS
21B7 E626                      OUT     COAD,AL          ; OUT DMA ADDR LOW
21B9 8AC4                      MOV     AL,AH            ;
21BB E626                      OUT     COAD,AL          ; OUT DMA ADDR high
21BD 8BC2                      MOV     AX,DX            ; AX <-- DMA LENGTH
21BF 48                        DEC     AX               ; DMA LENGTH-1
21C0 E627                      OUT     COTC,AL          ; OUT DMA LENGTH low
21C2 8AC4                      MOV     AL,AH            ;
21C4 E627                      OUT     COTC,AL          ; OUT DMA LENGTH high
                                                        ;
21C6 B003                      MOV     AL,03            ;
21C8 E62A                      OUT     DMAMB,AL         ; Enable FDC CHANNEL
                                                        ;
21CA C3                        RET                      ;
                       ;
```

```
                        LOADER.


                        ;
                        ;************************************
                        ;************************************
                        ;
                        ;               DATA AREA
                        ;
                        ;************************************
                        ;************************************
                        ;
  21CB                  DATA    EQU     OFFSET $
                                DSEG
                                ORG     DATA
                        ;
21CB 144E6F6E2053       NONSYS  DB      20,'Non-System disk (CR)'
     797374656D20
     6469736B203C
     43523E
                        ;
21E0 494F20202020       IOSYS   DB      'IO      SYS'
     2020535953
21EB 0002               SECSIZ  DW      512             ;512 BYTES/SECTOR
21ED 0F0000             COMSTR3 DB      SEEKTRK,0,0     ;DRIVE & HEAD, TRACK
21F0 C6                 COMSTR9 DB      0C6H            ;READDAT & DENSITY & MULTTRK
21F1 00000000                   DB      0,0,0,0
21F5 02                         DB      2               ;512 BYTES/SECTOR
21F6 09                 SECTRK  DB      9               ;SECTORS/TRACK
21F7 1B                         DB      GPL
21F8 FF                         DB      0FFH
                        ;
21F9 00                 TRACK   DB      0               ;PRESET TRACK ZERO
21FA 04                 SECTOR  DB      4               ;DEFAULT DOUBLE SIDED
21FB 50                 SECCNT  DB      80              ;LOAD 80 SECTORS (40 kbytes)
21FC 0004               DMAADDR DW      400H            ;BIOS TO START AT ABSOLUTE 400H
                        ;
  01CB                  LOADER_LENGTH   EQU     OFFSET $ - OFFSET LOADER
                        ;
21FE                    CNTR    RB      1               ;RETRY COUNTER
21FF                    CYLMODE RB      1               ; 0 = CYLINDER MODE, 1 = non CYLINDER MODE
2200                    HEAD    RB      1               ; HEAD NUMBER
                        ;
2201                    ERRBUF  RB      7               ; STATUS BYTES STRING (for errors)
                        ;
                        ;
                        ;
                                ORG     2200H
2200                    FAT     RB      1
                                ORG     2600H
2600                    IO      RB      32
                                ORG     2A00H
2A00                    IOFCFD  RB      32
                        ;
```

```
                    LOADER


                           ORG      0F111H
F111                JMP_CODE RB     1
F112                JMP_ADD  RW     1
                           ORG      0F800H
F800                CURSXY   RW     1
                           ORG      0F908H
F908                INVERS   RB     1
                           ORG      0FE05H
                    ;
                    END
```

```
                              ;
                              ; I/O system for Version 2.x of MSDOS.

                              ;This MODULE designed to be linked with the SYSINIT module provided by
                              ;Microsoft

 = 2C00                       BIOSIZ  EQU     11264           ;SIZE OF BIOS IN BYTES
 = 02C0                       BIOSIZS EQU     (BIOSIZ+15)/16  ;Size of BIOS in Paragraphs.
 = 0011                       ROM     EQU     11H             ;PORT SWITCH TO ROM
 = 0010                       RAM     EQU     10H             ;PORT SWITCH TO RAM
 = 000D                       CR      EQU     0DH
 = 000A                       LF      EQU     0AH
 = 0040                       BIOSSEG EQU     040H            ;I/O system segment.
 = 1000                       BANK_PAR EQU    1000H           ;Paragraphs per bank.
 = FE06                       MEM_SIZ_ID EQU  0FE06H          ;Absolute address where firmware passes
                                                              ; memory size ID.
 = 0FF7                       FW_VER  EQU     0FF7H           ;LOCATION OF FIRMWARE VERSION MESSAGE
                                                              ; 1st BYTE IS LENGTH


                              ;Things needed to communicate with SYSINIT

                              EXTRN   SYSINIT:FAR                   ;The entry point of SYSINIT
                              EXTRN   CURRENT_DOS_LOCATION:WORD     ;Where the DOS is when SYSINIT called
                              EXTRN   FINAL_DOS_LOCATION:WORD       ;Where I want SYSINIT to put the DOS
                              EXTRN   DEVICE_LIST:DWORD             ;Pointer to the DEVICE list.
                              EXTRN   MEMORY_SIZE:WORD              ;Size in paragraphs of Physical memory.
                              EXTRN   DEFAULT_DRIVE:BYTE            ;Default Drive to use when system booted
                              EXTRN   BUFFERS:BYTE                  ;Number of default buffers.
                                                                    ; Leave as is and SYSINIT uses only 2.
                              EXTRN   DEVSTART:NEAR
                              EXTRN   DREND:NEAR
                              EXTRN   OUTCTR:NEAR
                              EXTRN   ED:NEAR
                              EXTRN   FLOPPY_DRIVES:BYTE
                              EXTRN   DRVMAX:BYTE
                              EXTRN   READ_FW_VER:FAR
                              EXTRN   MONO_COLOR:BYTE
                              EXTRN   I29_HANDLER:NEAR
                              EXTRN   INT_TRAP:NEAR
                              EXTRN   CHTRANS:WORD
                              EXTRN   CHRTRN:NEAR

                              PUBLIC  HWINIT
                              PUBLIC  FIRM_MESS
```

```
BASINIT


                                  ;
                                  ;
0000                              CSEG    SEGMENT PUBLIC  'CODE'
                                  ASSUME  CS:CSEG,DS:CSEG,SS:CSEG,ES:CSEG

                                  ;
                                  ; Overlayed by MSDOS by SYSINIT.
                                  ;
0000  01 07 00                    RELEASE_ID      DB      01H,07H,00H
                                  ;
0003                              WRKSTK  LABEL   WORD
0003      64 [                            DB      100 DUP (?)          ;STACK FOR INITIALIZATION
                ??
                   ]


                                  ;
                                  ;  NCR SIGNON MESSAGE
                                  ;
0067  0073                        MESS_LEN        DW      SO_END - OFFSET SIGNON_MESSAGE
                                  ;
                                  ;
0069  4D 53 20 44 4F 53           SIGNON_MESSAGE  DB       'MS-DOS for NCR DECISION MATE V',CR,LF
      20 66 6F 72 20 4E
      43 52 20 44 45 43
      49 53 49 4F 4E 20
      4D 41 54 45 20 56
      0D 0A
0089  30 30 30 30 20 6B           SIZ_MESS        DB       '0000 kbytes memory',CR,LF
      62 79 74 65 73 20
      6D 65 6D 6F 72 79
      0D 0A
009D  44 30 30 36 20 30                           DB       'D006-0052-0300',CR,LF
      30 35 32 2D 30 33
      30 30 0D 0A
00AD  53 65 72 69 61 6C                           DB       'Serial number '
      20 6E 75 6D 62 65
      72 20
00BB  30 30 30 30 30 30           SERIAL_NR       DB       '000000',CR,LF
      0D 0A
00C3  46 69 72 6D 77 61                           DB       'Firmware version:'
      72 65 20 76 65 72
      73 69 6F 6E 3A
00D4  20 4D 2E 30 30 2E           FIRM_MESS       DB       ' M.00.00'
      30 30
= 00DC                            SO_END          EQU     OFFSET $
                                  ;
                                  ; MEMORY SIZE TABLE
                                  ;
00DC  20 20 36 34                 SIZ_TBL         DB      '  64'
00E0  20 31 32 38                                 DB      ' 128'
00E4  20 31 39 32                                 DB      ' 192'
00E8  20 32 35 36                                 DB      ' 256'
00EC  20 33 32 30                                 DB      ' 320'
00F0  20 33 38 34                                 DB      ' 384'
00F4  20 34 34 38                                 DB      ' 448'
00F8  20 35 31 32                                 DB      ' 512'
00FC  20 35 37 36                                 DB      ' 576'
0100  20 36 34 30                                 DB      ' 640'
0104  20 37 30 34                                 DB      ' 704'

0108  20 37 38 36                                 DB      ' 786'
010C  20 38 33 32                                 DB      ' 832'
0110  20 38 39 36                                 DB      ' 896'
0114  20 39 36 30                                 DB      ' 960'
0118  31 30 32 34                                 DB      '1024'
```

```
011C                      HWINIT:
011C  33 ED                       XOR    BP,BP
011E  8B DD                       MOV    BX,BP                  ;BX = ZERO
0120  8E DD                       MOV    DS,BP                  ;DS = ZERO
0122  8E C5                       MOV    ES,BP                  ;ES = ZERO
                          ;
                          ; INITIALIZE ALL INTERRUPTS TO ADDRESS TRAP
                          ;
0124  FC                          CLD
0125  C7 07 0000 E                MOV    WORD PTR [BX],OFFSET INT_TRAP
0129  43                          INC    BX
012A  43                          INC    BX
012B  8C 0F                       MOV    WORD PTR [BX],CS
012D  BF 0004                     MOV    DI,4
0130  BE 0000                     MOV    SI,0
0133  B9 01FE                     MOV    CX,510
0136  F3/ A5                REP   MOVSW                         ;ALL 256 INTERRUPT VECTORS SET TO
                                                                ; INT_TRAP NOW
0138  BB 00A4                     MOV    BX,00A4H               ;INT 29H VECTOR ADDRESS
013B  C7 07 0000 E                MOV    WORD PTR [BX],OFFSET I29_HANDLER ;INITIALIZE INT 29H
013F  8C C8                       MOV    AX,CS
0141  8E D0                       MOV    SS,AX                  ;SS = CS
0143  8E D8                       MOV    DS,AX                  ;DS = CS
0145  BC 0065 R                   MOV    SP,OFFSET WRKSTK+98    ;INITIALIZE STACK POINTER
                          ;
                          ; Calculate memory size in paragraphs from ID passed by firmware
                          ; at absolute address FE06 hex.
                          ;
0148  BB FE06                     MOV    BX,MEM_SIZ_ID
014B  26: 8A 07                   MOV    AL,BYTE PTR ES:[BX]    ;AL = MEMORY SIZE ID
                                                                ;00 = 64k, 01 = 128k, 02 = 192k
                                                                ;03 = 256k.
014E  50                          PUSH   AX                     ;SAVE MEMORY SIZE ID
014F  FE C0                       INC    AL                     ;Increment for calculation.
0151  98                          CBW                           ;Byte AL --> Word AX
0152  BB 1000                     MOV    BX,BANK_PAR            ;BX = Paragraphs per bank.
0155  F7 E3                       MUL    BX                     ;AX = Total memory size in paragraphs.
0157  8B D8                       MOV    BX,AX                  ; needed in BX.
0159  53                          PUSH   BX                     ;SAVE MEMORY SIZE (PARAGRAPHS)
015A  A0 0000 E                   MOV    AL,BYTE PTR FLOPPY_DRIVES
015D  A2 0000 E                   MOV    BYTE PTR DRVMAX,AL     ;NUMBER OF FLOPPY DRIVES
0160  E8 0000 E                   CALL   ED                     ;CLEAR SCREEN
0163  9A 0000 ---- E              CALL   READ_FW_VER            ;READ FIRMWARE VERSION MESSAGE
                                  ;
                                  ; BUILD SIGNON MESSAGE
                                  ;
0168  FC                          CLD
0169  BB FE06                     MOV    BX,MEM_SIZ_ID
016C  26: 8A 07                   MOV    AL,BYTE PTR ES:[BX]
016F  B4 00                       MOV    AH,0
0171  D0 E0                       SHL    AL,1
0173  D0 E0                       SHL    AL,1
0175  0E                          PUSH   CS
0176  07                          POP    ES                     ;ES = CS = DS
0177  BE 00DC R                   MOV    SI,OFFSET SIZ_TBL
017A  03 F0                       ADD    SI,AX
017C  BF 0089 R                   MOV    DI,OFFSET SIZ_MESS
017F  B9 0004                     MOV    CX,4
```

```
BASINIT

0182  F3/ A4                   REP  MOVSB
                            ;
                            ;
                            ;
0184  BE 2D00                       MOV     SI,BIOSIZ + 100H
0187  BB 00BB R                     MOV     BX,OFFSET SERIAL_NR
018A  B9 0003                       MOV     CX,3
018D                        USER_NXT:
018D  8A 04                         MOV     AL,BYTE PTR [SI]
018F  50                            PUSH    AX
0190  24 F0                         AND     AL,0F0H
0192  D0 E8                         SHR     AL,1
0194  D0 E8                         SHR     AL,1
0196  D0 E8                         SHR     AL,1
0198  D0 E8                         SHR     AL,1
019A  08 07                         OR      BYTE PTR [BX],AL
019C  58                            POP     AX
019D  24 0F                         AND     AL,0FH
019F  43                            INC     BX
01A0  08 07                         OR      BYTE PTR [BX],AL
01A2  46                            INC     SI
01A3  43                            INC     BX
01A4  E2 E7                         LOOP    USER_NXT
01A6  BE 0069 R                     MOV     SI,OFFSET SIGNON_MESSAGE
01A9  8B 0E 0067 R                  MOV     CX,WORD PTR MESS_LEN
01AD  AC                    NEXTCH: LODS    BYTE PTR [SI]
01AE  51                            PUSH    CX
01AF  E8 0000 E                     CALL    OUTCTR
01B2  59                            POP     CX
01B3  E2 F8                         LOOP    NEXTCH
01B5  C7 06 0000 E 0000 E           MOV     CHTRANS,OFFSET CHRTRN   ;                  *1*
                            ;
                            ; SIGNON MESSAGE DISPLAYED NOW .. WILL BE FOLLOWED BY MICROSOFT'S MESSAGE
                            ;
                            ;
                            ; SET CRT PARAMETERS
                            ;
01BB  A0 0005 R                     MOV     AL,BYTE PTR FIRM_MESS+1
01BE  A2 0000 E                     MOV     BYTE PTR MONO_COLOR,AL
01C1  B8 ---- E                     MOV     AX,SEG SYSINIT
01C4  8E D8                         MOV     DS,AX

                                    ASSUME  DS:SEG SYSINIT

01C6  8C C8                         MOV     AX,CS
01C8  05 02C0                       ADD     AX,BIOSIZS
01CB  A3 0000 E                     MOV     DS:[CURRENT_DOS_LOCATION],AX
01CE  8C CB                         MOV     BX,CS
01D0  B8 0000 E                     MOV     AX,OFFSET DREND
01D3  05 000F                       ADD     AX,15
01D6  B1 04                         MOV     CL,4
01D8  D3 E8                         SHR     AX,CL           ;(OFFSET DREND+15)/16
01DA  03 C3                         ADD     AX,BX
01DC  A3 0000 E                     MOV     DS:[FINAL_DOS_LOCATION],AX
01DF  5B                            POP     BX                      ;RESTORE MEMORY SIZE (PARAGRAPHS)
01E0  89 1E 0000 E                  MOV     DS:[MEMORY_SIZE],BX
01E4  8C C8                         MOV     AX,CS
01E6  A3 0002 E                     MOV     WORD PTR DS:[DEVICE_LIST+2],AX
01E9  C7 06 0000 E 0000 E           MOV     WORD PTR DS:[DEVICE_LIST],OFFSET DEVSTART
01EF  EA 0000 ---- E                JMP     SYSINIT
01F4                        CSEG    ENDS
                                    END
```

```
BASINIT


Segments and groups:

                N a m e             Size    align   combine class

CSEG . . . . . . . . . . . . . .    01F4    PARA    PUBLIC  'CODE'

Symbols:

                N a m e             Type    Value   Attr

BANK_PAR . . . . . . . . . . . .    Number  1000
BIOSIZ . . . . . . . . . . . . .    Number  2C00
BIOSIZS. . . . . . . . . . . . .    Number  02C0
BIOSSEG. . . . . . . . . . . . .    Number  0040
BUFFERS. . . . . . . . . . . . .    V BYTE  0000            External
CHRTRN . . . . . . . . . . . . .    L NEAR  0000            External
CHTRANS. . . . . . . . . . . . .    V WORD  0000            External
CR . . . . . . . . . . . . . . .    Number  000D
CURRENT_DOS_LOCATION . . . . . .    V WORD  0000            External
DEFAULT_DRIVE. . . . . . . . . .    V BYTE  0000            External
DEVICE_LIST. . . . . . . . . . .    V DWORD 0000            External
DEVSTART . . . . . . . . . . . .    L NEAR  0000            External
DREND. . . . . . . . . . . . . .    L NEAR  0000            External
DRVMAX . . . . . . . . . . . . .    V BYTE  0000            External
ED . . . . . . . . . . . . . . .    L NEAR  0000            External
FINAL_DOS_LOCATION . . . . . . .    V WORD  0000            External
FIRM_MESS. . . . . . . . . . . .    L BYTE  00D4    CSEG    Global
FLOPPY_DRIVES. . . . . . . . . .    V BYTE  0000            External
FW_VER . . . . . . . . . . . . .    Number  0FF7
HWINIT . . . . . . . . . . . . .    L NEAR  011C    CSEG    Global
I29_HANDLER. . . . . . . . . . .    L NEAR  0000            External
INT_TRAP . . . . . . . . . . . .    L NEAR  0000            External
LF . . . . . . . . . . . . . . .    Number  000A
MEMORY_SIZE. . . . . . . . . . .    V WORD  0000            External
MEM_SIZ_ID . . . . . . . . . . .    Number  FE06
MESS_LEN . . . . . . . . . . . .    L WORD  0067    CSEG
MONO_COLOR . . . . . . . . . . .    V BYTE  0000            External
NEXTCH . . . . . . . . . . . . .    L NEAR  01AD    CSEG
OUTCTR . . . . . . . . . . . . .    L NEAR  0000            External
RAM. . . . . . . . . . . . . . .    Number  0010
READ_FW_VER. . . . . . . . . . .    L FAR   0000            External
RELEASE_ID . . . . . . . . . . .    L BYTE  0000    CSEG
ROM. . . . . . . . . . . . . . .    Number  0011
SERIAL_NR. . . . . . . . . . . .    L BYTE  00BB    CSEG
SIGNON_MESSAGE . . . . . . . . .    L BYTE  0069    CSEG
SIZ_MESS . . . . . . . . . . . .    L BYTE  0089    CSEG
SIZ_TBL. . . . . . . . . . . . .    L BYTE  00DC    CSEG
SO_END . . . . . . . . . . . . .    Number  000C    CSEG
SYSINIT. . . . . . . . . . . . .    L FAR   0000            External
USER_NXT . . . . . . . . . . . .    L NEAR  018D    CSEG
WRKSTK . . . . . . . . . . . . .    L WORD  0003    CSEG
```

PROCEDURE TO READ FIRMWARE VERSION

```
 = 0011                          ROM    EQU    11H
 = 0010                          RAM    EQU    10H
 = 0FF7                          FW_VER EQU    0FF7H

                                 EXTRN  FIRM_MESS:BYTE

                                 PUBLIC READ_FW_VER

 0000                            FWSEG  SEGMENT 'FRCODE'
                                 ASSUME CS:FWSEG,DS:FWSEG,ES:FWSEG,SS:FWSEG

                                 ; ENTER WITH DS:40H, ES:00H

 0000                            READ_FW_VER PROC FAR

 0000                            RD_FW_VER:
 0000  E6 11                            OUT     ROM,AL                  ;SWITCH IN ROM
 0002  BB 0FF7                          MOV     BX,FW_VER               ;POINTER TO ROM LOCATION WHERE
                                                                        ;FIRMWARE VERSION IS DEFINED
 0005  26: 8A 0F                        MOV     CL,BYTE PTR ES:[BX]     ;FIRST BYTE IS LENGTH OF STRING
 0008  B5 00                            MOV     CH,0                    ;CX = LENGTH IN BYTES
 000A  80 F9 0B                         CMP     CL,0BH                  ;TEST FOR NO FIRMWARE MESSAGE
 000D  75 0C                            JNZ     NO_TRANS                ; JUMP IF NO MESSAGE
 000F  BE 0000 E                        MOV     SI,OFFSET FIRM_MESS     ;FIELD IN SIGNON MESSAGE
 0012  43                        F1:    INC     BX                      ;MOVE SOURCE POINTER
 0013  26: 8A 07                        MOV     AL,BYTE PTR ES:[BX]
 0016  88 04                            MOV     BYTE PTR [SI],AL
 0018  46                               INC     SI                      ;MOVE DESTINATION POINTER
 0019  E2 F7                            LOOP    F1                      ;LOOP TILL DONE (CX = 00)
 001B                            NO_TRANS:
 001B  E6 10                            OUT     RAM,AL                  ;SWITCH BACK TO RAM
 001D  CB                               RET

 001E                            READ_FW_VER ENDP
 001E                            FWSEG  ENDS
                                 END
```

```
PROCEDURE TO READ FIRMWARE VERSION


Segments and groups:

                N a m e             Size    align   combine class

FWSEG. . . . . . . . . . . . . .    001E    PARA    NONE    'FRCODE'

Symbols:

                N a m e             Type    Value   Attr

F1 . . . . . . . . . . . . . . .    L NEAR  0012    FWSEG
FIRM_MESS. . . . . . . . . . . .    V BYTE  0000            External
FW_VER . . . . . . . . . . . . .    Number  0FF7
NO_TRANS . . . . . . . . . . . .    L NEAR  001B    FWSEG
RAM. . . . . . . . . . . . . . .    Number  0010
RD_FW_VER. . . . . . . . . . . .    L NEAR  0000    FWSEG
READ_FW_VER. . . . . . . . . . .    F PROC  0000    FWSEG   Global  Length =001E
ROM. . . . . . . . . . . . . . .    Number  0011
```

```
                             ; I/O system for Version 2.x of MSDOS.

                             ;This module designed to be linked with KBD-DRV, DSKDRV, BASINIT, FWVERRO
                             ;and the modules SYSINIT and SYSIMES provided by Microsoft

 = 0040                      BIOSSEG EQU    40H               ;SEGMENT OF BIOS

                             PUBLIC  RE_INIT                  ;SYSINIT CALLS THIS ENTRY AFTER DEVICE
                                                              ; INSTALLATION
                             PUBLIC  ERROR_0,ERROR_1,ERROR_2,ERROR_3,ERROR_4,ERROR_5
                             PUBLIC  ERROR_6,ERROR_7,ERROR_8,ERROR_9,ERROR_10,ERROR_11
                             PUBLIC  ERROR_12,BUS_EXIT,ERR_EXIT,EXIT,EXIT1,
                             PUBLIC  OUTCTR,DEVSTART,PTRSAV,DRVMAX,FLOPPY_DRIVES,FL_OUT_RETRIES
                             PUBLIC  FL_IN_RETRIES,ED,MONO_COLOR
                             PUBLIC  kbd_tt,functbl,clear_1,clear_2,dec_sign_1,dec_sign_2
                             PUBLIC  I29_HANDLER,INT_TRAP,CHTRANS,CHRTRN,FLTAB

                             EXTRN HWINIT:NEAR
                             EXTRN key_init:NEAR
                             EXTRN key_in:NEAR
                             EXTRN key_nd_in:NEAR
                             EXTRN key_st:NEAR
                             EXTRN key_in_fl:NEAR
                             EXTRN kbd_out:NEAR
                             EXTRN language:BYTE
                             EXTRN def_fun:NEAR
                             EXTRN DSK_INT:NEAR
                             EXTRN flag_buf:BYTE
                             EXTRN remnum:WORD
                             EXTRN funcoff:WORD


 0000                        CSEG    SEGMENT PUBLIC  'CODE'
                             ASSUME  CS:CSEG,DS:CSEG,ES:CSEG,SS:CSEG

 0000                                ORG    0              ;Starts at an offset of zero.

 0000  E9 0000 E             INIT:   JMP    HWINIT         ;PASS PARAMETERS TO SYSINIT AND ACTIVATE
                                                           ; SYSTEM INITIALIZATION (HWINIT)
```

```
                        ;------------------------------------------------------------------
                        ;
                        ;          SYSTEM DEFINITION AREA AT ABSOLUTE ADDRESS 403H
                        ;
                        ;------------------------------------------------------------------

0003  0265 R            CRT_SB_FUNCT    DW    OFFSET CTABLEN  ;TABLE OF SINGLE BYTE CRT CONTROL
                                                              ;FUNCTIONS
0005  023D R            ANSI_ESC_SEQ    DW    OFFSET CMDTABL  ;TABLE OF ANSI ESCAPE SEQUENCES

0007  0284 R            CRT_TR_TABLE    DW    OFFSET CRTTBL   ;CRT TRANSLATION TABLE

0009  0285 R            NON_ANSI_ESC    DW    OFFSET ETBLENT  ;TABLE OF NON ANSI ESCAPE SEQUENCES

000B  021D R            KEYBOARD_TBL    DW    OFFSET kbd_tt   ;KEYBOARD TRANSLATION TABLE

000D  0000              KEYBFUNC_TBL    DW    OFFSET functbl  ;TABLE OF FUNCTION KEY ASSIGNMENTS


000F  0000                              DW    0               ;SPARE

0011  0000              FLTAB           DW    0               ;FLEX MEDIA CHANGE BYTES    *7*
                                                              ;1st BYTE - UNIT 0
                                                              ;2nd BYTE - UNIT 1

                        ;

0013  02                FLOPPY_DRIVES   DB    2               ;DEFAULTS TO 2

0014  05                FL_OUT_RETRIES  DB    5               ;DEFAULTS TO 5

0015  05                FL_IN_RETRIES   DB    5               ;DEFAULTS TO 5

                        ;

0016  00                WINCH_DRIVES    DB    0               ;DEFAULTS TO 0

0017  05                WI_OUT_RETRIES  DB    5               ;DEFAULTS TO 5

0018  05                WI_IN_RETRIES   DB    5               ;DEFAULTS TO 5

                        ;

0019  50                PRINTER_IF_TYPE DB    'P'             ;DEFAULTS TO PARALLEL ('P')
                                                              ;FOR SERIAL ENTER ('S')
                        ;

001A  79                M1RS232         DB    79H             ;DEFAULT:
                                                              ;       1 STOP BIT
                                                              ;       EVEN PARITY AND ENABLED
                                                              ;       7 BITS PER CHARACTER
                                                              ;       ASYNCHRONOUS

001B  3E                M2RS232         DB    3EH             ;DEFAULT:
                                                              ;       INTERNAL CLOCKS
                                                              ;       9600 BAUD

001C  00                PVRS232         DB    0               ;ZERO....ONE PROTOCOL ONLY
```

```
                                        ;
                                        ;       ****************************************************
                                        ;       ****************************************************
                                        ;       ***                                              ***
                                        ;       ***          F U N C T I O N     K E Y           ***
                                        ;       ***                                              ***
                                        ;       ***            D E F I N I T I O N S             ***
                                        ;       ***                                              ***
                                        ;       ****************************************************
                                        ;       ****************************************************
                                        ;
                                        ;
= 0200                                  funclen equ     512                     ;max. length of function table
= 001D                                  functbl =       fun1
001D  03                                        fun1    db      len1                    ; first function length
001E  46 31                                             db      "F1"                    ; first function content
= 0003                                          len1    =       $-fun1                  ; length set
                                        ;
0020  03                                        fun2    db      len2
0021  46 32                                             db      "F2"
= 0003                                          len2    =       $-fun2
                                        ;
0023  03                                        fun3    db      len3
0024  46 33                                             db      "F3"
= 0003                                          len3    =       $-fun3
                                        ;
0026  03                                        fun4    db      len4
0027  46 34                                             db      "F4"
= 0003                                          len4    =       $-fun4
                                        ;
0029  03                                        fun5    db      len5
002A  46 35                                             db      "F5"
= 0003                                          len5    =       $-fun5
                                        ;
002C  03                                        fun6    db      len6
002D  46 36                                             db      "F6"
= 0003                                          len6    =       $-fun6
                                        ;
002F  03                                        fun7    db      len7
0030  46 37                                             db      "F7"
= 0003                                          len7    =       $-fun7
                                        ;
0032  03                                        fun8    db      len8
0033  46 38                                             db      "F8"
= 0003                                          len8    =       $-fun8
                                        ;
0035  03                                        fun9    db      len9
0036  46 39                                             db      "F9"
= 0003                                          len9    =       $-fun9
                                        ;
0038  04                                        fun10   db      len10
0039  46 31 30                                          db      "F10"
= 0004                                          len10   =       $-fun10
                                        ;
003C  04                                        fun11   db      len11
003D  46 31 31                                          db      "F11"
= 0004                                          len11   =       $-fun11
                                        ;
0040  04                                        fun12   db      len12
0041  46 31 32                                          db      "F12"
```

```
= 0004                          len12   =       $-fun12
                        ;
0044  04                        fun13   db      len13
0045  46 31 33                          db      "F13"
= 0004                          len13   =       $-fun13
                        ;
0048  04                        fun14   db      len14
0049  46 31 34                          db      "F14"
= 0004                          len14   =       $-fun14
                        ;
004C  04                        fun15   db      len15
004D  46 31 35                          db      "F15"
= 0004                          len15   =       $-fun15
                        ;
0050  04                        fun16   db      len16
0051  46 31 36                          db      "F16"
= 0004                          len16   =       $-fun16
                        ;
0054  04                        fun17   db      len17
0055  46 31 37                          db      "F17"
= 0004                          len17   =       $-fun17
                        ;
0058  04                        fun18   db      len18
0059  46 31 38                          db      "F18"
= 0004                          len18   =       $-fun18
                        ;
005C  04                        fun19   db      len19
005D  46 31 39                          db      "F19"
= 0004                          len19   =       $-fun19
                        ;
0060  04                        fun20   db      len20
0061  46 32 30                          db      "F20"
= 0004                          len20   =       $-fun20
                        ;
= 01B9                          funfill =       funclen-($-functbl)
0064  01B9 [                            db      1b9h    dup (00)
            00          ; fill rest of function table with zero
               ]

                        ;       *************************************
                        ;       ***   !!! funfill = 1b9h  !!!!!    ***
                        ;       *************************************
                        ;
                        ;
                        ; functbl       end
```

```
                            .
                            ;       ****************************************************
                            ;       ****************************************************
                            ;       ***                                              ***
                            ;       ***        F U N C T I O N      K E Y            ***
                            ;       ***                                              ***
                            ;       ***           D E F I N I T I O N S              ***
                            ;       ***                                              ***
                            ;       ****************************************************
                            ;       ****************************************************
                            ;
                            ;
= 0200                      funclen equ     512                     ;max. length of function table
= 001D                      functbl =       fun1
001D  03                            fun1    db      len1                    ; first function length
001E  46 31                                 db      "F1"                    ; first function content
= 0003                              len1    =       $-fun1                  ; length set
                            ;
0020  03                            fun2    db      len2
0021  46 32                                 db      "F2"
= 0003                              len2    =       $-fun2
                            ;
0023  03                            fun3    db      len3
0024  46 33                                 db      "F3"
= 0003                              len3    =       $-fun3
                            ;
0026  03                            fun4    db      len4
0027  46 34                                 db      "F4"
= 0003                              len4    =       $-fun4
                            ;
0029  03                            fun5    db      len5
002A  46 35                                 db      "F5"
= 0003                              len5    =       $-fun5
                            ;
002C  03                            fun6    db      len6
002D  46 36                                 db      "F6"
= 0003                              len6    =       $-fun6
                            ;
002F  03                            fun7    db      len7
0030  46 37                                 db      "F7"
= 0003                              len7    =       $-fun7
                            ;
0032  03                            fun8    db      len8
0033  46 38                                 db      "F8"
= 0003                              len8    =       $-fun8
                            ;
0035  03                            fun9    db      len9
0036  46 39                                 db      "F9"
= 0003                              len9    =       $-fun9
                            ;
0038  04                            fun10   db      len10
0039  46 31 30                              db      "F10"
= 0004                              len10   =       $-fun10
                            ;
003C  04                            fun11   db      len11
003D  46 31 31                              db      "F11"
= 0004                              len11   =       $-fun11
                            ;
0040  04                            fun12   db      len12
0041  46 31 32                              db      "F12"
```

```
= 0004                            len12    =       $-fun12
                              ;
0044  04                          fun13    db      len13
0045  46 31 33                             db      "F13"
= 0004                            len13    =       $-fun13
                              ;
0048  04                          fun14    db      len14
0049  46 31 34                             db      "F14"
= 0004                            len14    =       $-fun14
                              ;
004C  04                          fun15    db      len15
004D  46 31 35                             db      "F15"
= 0004                            len15    =       $-fun15
                              ;
0050  04                          fun16    db      len16
0051  46 31 36                             db      "F16"
= 0004                            len16    =       $-fun16
                              ;
0054  04                          fun17    db      len17
0055  46 31 37                             db      "F17"
= 0004                            len17    =       $-fun17
                              ;
0058  04                          fun18    db      len18
0059  46 31 38                             db      "F18"
= 0004                            len18    =       $-fun18
                              ;
005C  04                          fun19    db      len19
005D  46 31 39                             db      "F19"
= 0004                            len19    =       $-fun19
                              ;
0060  04                          fun20    db      len20
0061  46 32 30                             db      "F20"
= 0004                            len20    =       $-fun20
                              ;
= 01B9                            funfill =        funclen-($-functbl)
0064  01B9 [                               db      1b9h    dup (00)
              00              ; fill rest of function table with zero
                 ]

                              ;        **************************************
                              ;        ***   !!! funfill = 1b9h  !!!!!     ***
                              ;        **************************************
                              ;
                              ;
                              ; functbl        end
```

IO.SYS

```
                                  ;
                                  ;
021D  00                          kbd_tt  db      00h                  ; 80 H
021E  17                                  db      17h                  ; 81 H
021F  13                                  db      13h                  ; 82 H  cursor left
0220  18                                  db      18h                  ; 83 H  cursor down
0221  05                                  db      05h                  ; 84 H  cursor up
0222  04                                  db      04h                  ; 85 H  cursor right
0223  18                          clear_1  db     18h                  ; 86 H  clear line (rubout)     *6*
0224  07                                  db      07h                  ; 87 H
0225  0D                                  db      0dh                  ; 88 H  carrige return
0226  09                                  db      09h                  ; 89 H
0227  2C                          dec_sign_1  db  2ch                  ; 8a H  comma (may be changed by KBD_INIT routine)
0228  08                                  db      08h                  ; 8b H  backspace
0229  0C                                  db      0ch                  ; 8c H
022A  0D                                  db      0dh                  ; 8d H
022B  0E                                  db      0eh                  ; 8e H
022C  0F                                  db      0fh                  ; 8f H
                                  ;
022D  10                                  db      10h                  ; 90 H
022E  17                                  db      17h                  ; 91 H
022F  13                                  db      13h                  ; 92 H  cursor left
0230  18                                  db      18h                  ; 93 H  cursor down
0231  05                                  db      05h                  ; 94 H  cursor up
0232  04                                  db      04h                  ; 95 H  cursor right
0233  18                          clear_2  db     18h                  ; 96 H  clear line (rubout)     *6*
0234  17                                  db      17h                  ; 97 H
0235  0D                                  db      0dh                  ; 98 H  carrige return
0236  19                                  db      19h                  ; 99 H
0237  2C                          dec_sign_2  db  2ch                  ; 9a H  comma (may be changed by KBD_INIT routine)
0238  08                                  db      08h                  ; 9b H  backspace
0239  1C                                  db      1ch                  ; 9c H
023A  1D                                  db      1dh                  ; 9d H
023B  1E                                  db      1eh                  ; 9e H
023C  1F                                  db      1fh                  ; 9f H
```

```
                              ;
                              ; ANSI ESCAPE SEQUENCES
                              ;
0230  41                      CMDTABL DB      'A'           ;Cursor up.  "esc","[",÷,"A"
023E  094D R                          DW      CUU
0240  42                              DB      'B'           ;Cursor down. "esc","[",÷,"B"
0241  0951 R                          DW      CUD
0243  43                              DB      'C'           ;Cursor forward. "esc","[",÷,"C"
0244  0955 R                          DW      CUF
0246  44                              DB      'D'           ;Cursor back. "esc","[",÷,"D"
0247  0959 R                          DW      CUB
0249  48                              DB      'H'           ;Direct cursor posit. "esc","[",x,y,"H"
024A  0B68 R                          DW      CUP
024C  4A                              DB      'J'           ;Erase. "esc","[",code,"J"
024D  0B79 R                          DW      ED
024F  4B                              DB      'K'           ;Erase in line. "esc","[",code,"K"
0250  0B8A R                          DW      EL
0252  66                              DB      'f'           ;Direct cursor posit. "esc","[",x,y,"f"
0253  0B68 R                          DW      CUP
0255  6D                              DB      'm'           ;Special video mode. "esc","[",code,"m"
0256  0B91 R                          DW      SGR
0258  73                              DB      's'           ;Save cursor posit. "esc","[","s"
0259  0C50 R                          DW      PSCP
025B  75                              DB      'u'           ;Move cursor to saved. "esc","[","u"
025C  0C57 R                          DW      PRCP
025E  70                              DB      'p'           ;Define Function Key
025F  0C5D R                          DW      DEFFK         ;"esc","[",0,FN,"string",cr,"string".."p"
                                                            ;Disable Function Key extension
                                                            ;"esc","[",0,0"p"
                                                            ;Enable Function Key extension
                                                            ;"esc","[",0,99"p"

0261  6E                              DB      'n'           ;Cursor position report
0262  0C88 R                          DW      XDSR          ;"esc","[","6","n"
0264  00                              DB      00            ;End of table.
```

IO.SYS

```
                               ;
                               ;
                               ; COMTBL SINGLE BYTE CRT CONTROL FUNCTIONS
                               ;
0265  0009                     CTABLEN DW      9       ;NUMBER OF TABLE ENTRIES
0267  07                       COMTBL  DB      07H
0268  09A2 R                           DW      BELL    ;RING THE BELL
026A  08                               DB      08H
026B  09A8 R                           DW      BACKSP  ;NON DESTRUCTIVE BACKWARD SPACE
026D  0A                               DB      0AH
026E  09BB R                           DW      LINEFD  ;LINE FEED
0270  0B                               DB      0BH
0271  09C2 R                           DW      RLF     ;REVERSE LINE FEED
0273  0C                               DB      0CH
0274  09CB R                           DW      NDFS    ;NON DESTRUCTIVE FORWARD SPACE
0276  0D                               DB      0DH
0277  09D8 R                           DW      CARRET  ;CARRIAGE RETURN
0279  17                               DB      17H
027A  0B8A R                           DW      EL      ;ERASE TO END OF LINE
027C  1A                               DB      1AH
027D  0B79 R                           DW      ED      ;CLEAR SCREEN
027F  1E                               DB      1EH
0280  09DF R                           DW      VHOME   ;HOME CURSOR
                               ;
                               ; E&M INTERNAL RELEASE ID                                         *3*
                               ;
0282  01                               DB      01H     ;ISSUE
0283  06                               DB      06H     ;SUB-ISSUE
0284  00                               DB      00H     ;PATCH LEVEL
                               ;
                               ; PLACED HERE AS CONFIG COPIES UP TO ADDRESS 27FH
                               ; THIS RELEASE ID MUST NOT BE COPIED BY CONFIG
                               ;
```

```
                                ;
                                ;
                                ; ESCTBL TABLE OF NON ANSI ESCAPE FUNCTIONS
                                ;
                                ;
0285  000F                      ETBLENT DW      15      ;NUMBER OF TABLE ENTRIES
0287  3A                        ESCTBL  DB      3AH
0288  0B79 R                            DW      ED      ;CLEAR SCREEN
028A  2A                                DB      2AH
028B  0B79 R                            DW      ED      ;CLEAR SCREEN
028D  29                                DB      29H
028E  09E3 R                            DW      SHALF_INT       ;SET HALF INTENSITY
0290  28                                DB      28H
0291  0A32 R                            DW      RHALF_INT       ;RESET HALF INTENSITY
0293  59                                DB      'Y'
0294  0ADE R                            DW      BLEOS   ;ERASE TO END OF SCREEN
0296  79                                DB      'y'
0297  0ADE R                            DW      BLEOS
0299  54                                DB      'T'
029A  0B8A R                            DW      EL      ;ERASE TO END OF LINE
029C  74                                DB      't'
029D  0B8A R                            DW      EL
029F  45                                DB      'E'
02A0  0AE6 R                            DW      INSLIN  ;INSERT LINE
02A2  52                                DB      'R'
02A3  0AEE R                            DW      DELLIN  ;DELETE LINE
02A5  51                                DB      'Q'
02A6  0AF6 R                            DW      ICHR    ;INSERT CHARACTER
02A8  57                                DB      'W'
02A9  0AFE R                            DW      DCHR    ;DELETE CHARACTER
02AB  3D                                DB      '='
02AC  0B06 R                            DW      POSIT   ;POSITION CURSOR
02AE  4D                                DB      'M'
02AF  0CCB R                            DW      PLAY_MUSIC ;MUSIC
02B1  47                                DB      'G'
02B2  0B38 R                            DW      REVERSE ;INVERSE VIDEO
```

```
                        ;****************************************************************
                        ;*                                                              *
                        ;*              CRT TRANSLATE TABLE                             *
                        ;*                                                              *
                        ;****************************************************************


02B4                    CRTTBL:
                        ;
                        ;
                        ;
02B4  03                LVAR0:  DB      VAR0L
02B5  8A 2E             US:     DB      8AH,2EH
= 0003                  VAR0L   EQU     $-LVAR0

02B7  07                LVAR1:  DB      VAR1L
02B8  5E 0E 23 03 8A 2E UK:     DB      5EH,0EH,23H,03H,8AH,2EH
= 0007                  VAR1L   EQU     $-LVAR1

02BE  15                LVAR2:  DB      VAR2L
02BF  5B 0D 5C 08 5D 1C FRANCE: DB      5BH,0DH,5CH,08H,5DH,1CH,40H,0AH,7BH,14H,7CH,1AH
      40 0A 7B 14 7C 1A
02CB  7D 0B 7E 0F 23 03         DB      7DH,0BH,7EH,0FH.23H,03H,27H,0CH
      27 0C
= 0015                  VAR2L   EQU     $-LVAR2

02D3  13                LVAR3:  DB      VAR3L
02D4  5B 00 5C 06 5D 09 GERMANY:DB      5BH,00H,5CH,06H,5DH,09H,40H.1CH,7BH,10H,7CH,16H
      40 1C 7B 10 7C 16
02E0  7D 19 7E 1E 27 0C         DB      7DH,19H,7EH,1EH,27H,0CH
= 0013                  VAR3L   EQU     $-LVAR3

02E6  13                LVAR4:  DB      VAR4L
02E7  5B 00 5C 06 5D 02 SWEDEN: DB      5BH,00H,5CH,06H,5DH,02H,24H,13H,7BH,10H,7CH,16H
      24 13 7B 10 7C 16
02F3  7D 12 7E 0F 27 0C         DB      7DH,12H,7EH,0FH,27H,0CH
= 0013                  VAR4L   EQU     $-LVAR4

02F9  13                LVAR5:  DB      VAR5L
02FA  5B D1 5C 07 5D 02 DANSK:  DB      5BH,D1H,5CH,07H,5DH,02H,23H,03H,7BH,11H,7CH,17H
      23 03 7B 11 7C 17
0306  7D 12 7E 0F 27 0C         DB      7DH,12H,7EH,0FH,27H,0CH
= 0013                  VAR5L   EQU     $-LVAR5

030C  0D                LVAR6:  DB      VAR6L
030D  5B 1F 5C 05 5D 1D KSPAIN: DB      5BH,1FH,5CH,05H,5DH,1DH,27H,0CH,7CH,15H,23H,03H
      27 0C 7C 15 23 03
= 000D                  VAR6L   EQU     $-LVAR6

0319  17                LVAR7:  DB      VAR7L
031A  5B 0D 5C 08 5D 14 ITALY:  DB      5BH,0DH,5CH,08H,5DH,14H,23H,03H,40H,1CH,7BH,0AH
      23 03 40 1C 7B 0A
0326  7C 18 7D 0B 7E 1B         DB      7CH,18H,7DH,0BH,7EH,1BH,60H,1AH,27H,0CH
      60 1A 27 0C
= 0017                  VAR7L   EQU     $-LVAR7
```

```
0330  15                    LVAR8:  DB      VAR8L
0331  23 03 27 0C 40 08     SWISS12: DB     23H,03H,27H,0CH,40H,08H,5BH;0AH,5CH,14H,5DH,0BH
      5B 0A 5C 14 5D 0B
033D  7B 10 7C 16 7D 19             DB      7BH,10H,7CH,16H,7DH,19H,7EH,0FH
      7E 0F
= 0015                      VAR8L   EQU     $-LVAR8

0345  03                    LVAR9:  DB      VAR9L
0346  8A 2E                 AUSTRALIA: DB   8AH,2EH                    ;               *8*11*
= 0003                      VAR9L   EQU     $-LVAR9

0348  0F                    LVAR10: DB      VAR10L
0349  27 0C 40 0A 5C 08     CANADA2: DB     27H,0CH,40H,0AH,5CH,08H,7BH,14H,7CH,9FH,7DH,0BH,7EH,0FH ;*8*11
      7B 14 7C 9F 7D 0B
      7E 0F
= 000F                      VAR10L  EQU     $-LVAR10

0357  11                    LVAR11: DB      VAR11L
0358  27 0C 5B 83 5C 84     SAFRICA: DB     27H,0CH,5BH,83H,5CH,84H,5DH,82H,7BH,93H ;       *15*
      5D 82 7B 93
0362  7C 94 7D 92 7E 0F             DB      7CH,94H,7DH,92H,7EH,0FH
= 0011                      VAR11L  EQU     $-LVAR11

0368  11                    LVAR12: DB      VAR12L
0369  23 03 27 0C 5B 80     PORTUG: DB      23H,03H,27H,0CH,5BH,80H,5CH,81H,5DH.85H,7BH,90H
      5C 81 5D 85 7B 90
0375  7C 91 7D 08                   DB      7CH,91H,7DH,08H
= 0011                      VAR12L  EQU     $-LVAR12

0379  15                    LVAR13: DB      VAR13L
037A  40 8C 5B 8B 5C 88     YUGOSL: DB      40H,8CH,5BH,8BH,5CH,88H,5DH,89H,5EH,8AH,60H,9CH ;*8*
      5D 89 5E 8A 60 9C
0386  7B 9B 7C 98 7D 99             DB      7BH,9BH,7CH,98H,7DH,99H.7EH,9AH                  ;*8*
      7E 9A
= 0015                      VAR13L EQU      $-LVAR13
038E      08 [                      DB      8 DUP (' ')     ;SPACE FOR EXTENSION
                20
                   ]
```

```
                              ;
                              ; ACTIVE CRT TRANSLATION TABLE
                              ;
0396                          CRTACTTBL:
0396  20 21 22 23 24 25              DB    20H;21H,22H,23H,24H,25H,26H,27H,28H,29H,2AH,2BH,2CH,2DH,
      26 27 28 29 2A 2B                                                                  2EH,2FH
      2C 2D 2E 2F
03A6  30 31 32 33 34 35              DB    30H,31H,32H,33H,34H,35H,36H,37H,38H,39H,3AH,3BH,3CH,3DH,
      36 37 38 39 3A 3B                                                                  3EH,3FH
      3C 3D 3E 3F
03B6  40 41 42 43 44 45              DB    40H,41H,42H,43H,44H,45H,46H,47H,48H,49H,4AH,4BH,4CH,4DH,
      46 47 48 49 4A 4B                                                                  4EH,4FH
      4C 4D 4E 4F
03C6  50 51 52 53 54 55              DB    50H,51H,52H,53H,54H,55H,56H,57H,58H,59H,5AH,5BH,5CH,5DH.
      56 57 58 59 5A 5B                                                                  5EH,5FH
      5C 5D 5E 5F
03D6  60 61 62 63 64 65              DB    60H,61H,62H,63H,64H,65H,66H,67H,68H,69H,6AH,6BH,6CH,6DH,
      66 67 68 69 6A 6B                                                                  6EH,6FH
      6C 6D 6E 6F
03E6  70 71 72 73 74 75              DB    70H,71H,72H,73H,74H,75H,76H,77H,78H,79H,7AH,7BH,7CH,7DH,
      76 77 78 79 7A 7B                                                                  7EH,7FH
      7C 7D 7E 7F
03F6  80 81 82 83 84 85              DB    80H,81H,82H,83H,84H,85H,86H,87H,88H,89H,8AH,8BH,8CH,8DH,
      86 87 88 89 8A 8B                                                                  8EH,8FH
      8C 8D 8E 8F
0406  90 91 92 93 94 95              DB    90H,91H,92H,93H,94H,95H,96H,97H,98H,99H,9AH,9BH,9CH,9DH,
      96 97 98 99 9A 9B                                                                  9EH,9FH
      9C 9D 9E 9F
0416  A0 A1 A2 A3 A4 A5              DB    0A0H,0A1H,0A2H,0A3H,0A4H,0A5H,0A6H,0A7H,0A8H,0A9H,0AAH,
      A6 A7 A8 A9 AA AB                                              0ABH,0ACH,0ADH,0AEH,0AFH
      AC AD AE AF
0426  B0 B1 B2 B3 B4 B5              DB    0B0H,0B1H,0B2H,0B3H,0B4H,0B5H,0B6H,0B7H,0B8H,0B9H,0BAH,
      B6 B7 B8 B9 BA BB                                              0BBH,0BCH,0BDH,0BEH,0BFH
      BC BD BE BF
0436  C0 C1 C2 C3 C4 C5              DB    0C0H,0C1H,0C2H,0C3H,0C4H,0C5H,0C6H,0C7H,0C8H,0C9H,0CAH,
      C6 C7 C8 C9 CA CB                                              0CBH,0CCH,0CDH,0CEH,0CFH
      CC CD CE CF
0446  D0 D1 D2 D3 D4 D5              DB    0D0H,0D1H,0D2H,0D3H,0D4H,0D5H,0D6H,0D7H,0D8H,0D9H,0DAH,
      D6 D7 D8 D9 DA DB                                              0DBH,0DCH,0DDH,0DEH,0DFH
      DC DD DE DF
0456  E0 E1 E2 E3 E4 E5              DB    0E0H;0E1H,0E2H,0E3H,0E4H,0E5H,0E6H,0E7H,0E8H,0E9H,0EAH,
      E6 E7 E8 E9 EA EB                                              0EBH,0ECH,0EDH,0EEH,0EFH
      EC ED EE EF
0466  F0 F1 F2 F3 F4 F5              DB    0F0H,0F1H,0F2H,0F3H,0F4H,0F5H,0F6H,0F7H,0F8H,0F9H,0FAH.
      F6 F7 F8 F9 FA FB                                              0FBH,0FCH,0FDH,0FEH,0FFH
      FC FD FE FF
```

```
                          ;---------------------------------------------+
                          ;    DWORD pointer to next device             | 1 word offset.
                          ;       (-1,-1 if last device)                | 1 word segment.
                          ;---------------------------------------------+
                          ;    Device attribute WORD                    ; 1 word.
                          ;      Bit 15 = 1 for chacter devices.        ;
                          ;               0 for Block devices.          ;
                          ;                                             ;
                          ;     Charcter devices. (Bit 15=1)            ;
                          ;       Bit 0 = 1  current sti device.        ;
                          ;       Bit 1 = 1  current sto device.        ;
                          ;       Bit 2 = 1  current NUL device.        ;
                          ;       Bit 3 = 1  current Clock device.      ;
                          ;                                             ;
                          ;       Bit 13 = 1 for non IBM machines.      ;
                          ;                0 for IBM machines only.     ;
                          ;       Bit 14 = 1 IOCTL control bit.         ;
                          ;---------------------------------------------+
                          ;    Device strategy pointer.                 ; 1 word offset.
                          ;---------------------------------------------+
                          ;    Device interrupt pointer.                ; 1 word offset.
                          ;---------------------------------------------+
                          ;    Device name field.                       ; 8 bytes.
                          ;      Character devices are any valid name   ;
                          ;        left justified, in a space filled    ;
                          ;        field.                               ;
                          ;      Block devices contain # of units in    ;
                          ;        the first byte.                      ;
                          ;---------------------------------------------+

0476                      DEVSTART LABEL WORD
0476                      CONDEV:                     ;Header for device CON
0476  0488 R 0040                 DW      AUXDEV,BIOSSEG ;Link to next device
047A  8013                        DW      8013H       ;Attributes - console input, output device
                                                      ; INT 29H SUPPORT
047C  053C R                      DW      STRATEGY    ;Srategy entry point
047E  0547 R                      DW      CON_INT     ;Interrupt entry point
0480  43 4F 4E 20 20 20           DB      'CON        ;Device name
      20 20

0488                      AUXDEV:                     ;Header for device AUX
0488  049A R 0040                 DW      PRNDEV,BIOSSEG
048C  8000                        DW      8000H
048E  053C R                      DW      STRATEGY
0490  054D R                      DW      AUX_INT
0492  41 55 58 20 20 20           DB      'AUX
      20 20

049A                      PRNDEV:                     ;Header for device PRN
049A  04AC R 0040                 DW      TIMDEV,BIOSSEG
049E  8000                        DW      8000H
04A0  053C R                      DW      STRATEGY
04A2  0553 R                      DW      PRN_INT
04A4  50 52 4E 20 20 20           DB      'PRN
      20 20
```

IO.SYS

```
04AC                          TIMDEV:                          ;Header for device CLOCK
04AC  04BE R 0040                     DW      DSKDEV,BIOSSEG
04B0  8008                            DW      8008H
04B2  053C R                          DW      STRATEGY
04B4  0559 R                          DW      TIM_INT
04B6  43 4C 4F 43 4B 20               DB      'CLOCK
      20 20

04BE                          DSKDEV:                          ;Header for disk devices
04BE  FFFF FFFF                       DW      -1,-1            ;Last device
04C2  2000                            DW      2000H            ;Is a block device
04C4  053C R                          DW      STRATEGY
04C6  0000 E                          DW      DSK_INT
04C8  02                      DRVMAX  DB      2                ;Number of Units
04C9     07 [                         DB      7 DUP (?)
                ??
                   ]
```

```
04D0  0000 E            CONTBL: DW      key_init        ;0  - Init.
04D2  05C7 R                    DW      EXIT            ;1  - Media check (Not used)
04D4  05C7 R                    DW      EXIT            ;2  - Get Bios Parameter Block (Not used)
04D6  059C R                    DW      ERROR_3         ;3  - Reserved. (Currently returns error)
04D8  0000 E                    DW      key_in          ;4  - Character read. (Destructive)
04DA  0000 E                    DW      key_nd_in       ;5  - Character read. (Non-destructive)
04DC  0000 E                    DW      key_st          ;6  - Return status.
04DE  0000 E                    DW      key_in_fl       ;7  - Flush Input buffer.
04E0  05FC R                    DW      CON_WRIT        ;8  - Character write.
04E2  05FC R                    DW      CON_WRIT        ;9  - Character write with Verify.
04E4  05C7 R                    DW      EXIT            ;10 - Character write status. (Not used)
04E6  05C7 R                    DW      EXIT            ;11 - Flush output buffer. (Not used.)
04E8  05C7 R                    DW      EXIT            ;12 - IO Control.

04EA  05C7 R            AUXTBL: DW      EXIT            ;0  - Init. (Not used)
04EC  05C7 R                    DW      EXIT            ;1  - Media check (Not used)
04EE  05C7 R                    DW      EXIT            ;2  - Get Bios Parameter Block (Not used)
04F0  059C R                    DW      ERROR_3         ;3  - Reserved. (Returns an error)
04F2  0598 R                    DW      ERROR_2         ;4  - Character read. (Destructive) *14*
04F4  058C R                    DW      BUS_EXIT        ;5  - Character read. (Non-destructive)
04F6  0598 R                    DW      ERROR_2         ;6  - Return status. (Not used)  *14*
04F8  05C7 R                    DW      EXIT            ;7  - Flush Input buffer.
04FA  0598 R                    DW      ERROR_2         ;8  - Character write. *14*
04FC  0598 R                    DW      ERROR_2         ;9  - Character write with verify. *14*
04FE  0598 R                    DW      ERROR_2         ;10 - Character write status. *14*
0500  05C7 R                    DW      EXIT            ;11 - Flush output buffer. (Not used.)
0502  05C7 R                    DW      EXIT            ;12 - IO Control.

0504  05C7 R            TIMTBL: DW      EXIT            ;0  - Init. (Not used)
0506  05C7 R                    DW      EXIT            ;1  - Media check (Not used)
0508  05C7 R                    DW      EXIT            ;2  - Get Bios Parameter Block (Not used)
050A  059C R                    DW      ERROR_3         ;3  - Reserved. (Currently returns an error)
050C  0073 R                    DW      TIM_RED         ;4  - Character read. (Destructive)
050E  058C R                    DW      BUS_EXIT        ;5  - (Not used, returns busy flag.)
0510  05C7 R                    DW      EXIT            ;6  - Return status. (Not used)
0512  05C7 R                    DW      EXIT            ;7  - Flush Input buffer. (Not used)
0514  005E R                    DW      TIM_WRT         ;8  - Character write.
0516  005E R                    DW      TIM_WRT         ;9  - Character write with verify.
0518  05C7 R                    DW      EXIT            ;10 - Character write status. (Not used)
051A  05C7 R                    DW      EXIT            ;11 - Flush output buffer. (Not used)
051C  05C7 R                    DW      EXIT            ;12 - IO Control.

051E  0002 R            PRNTBL: DW      PRN_INIT        ;INIT                         *5*
0520  05C7 R                    DW      EXIT            ;1  - (Not used)
0522  05C7 R                    DW      EXIT            ;2  - Block (Not used)
0524  059C R                    DW      ERROR_3         ;3  - Reserved. (currently returns error)
0526  05C7 R                    DW      EXIT            ;4  - (Not used)
0528  058C R                    DW      BUS_EXIT        ;5  - (Not used, returns busy flag.)
052A  05C7 R                    DW      EXIT            ;6  - (Not used)
052C  05C7 R                    DW      EXIT            ;7  - (Not used)
052E  0031 R                    DW      PRN_WRT         ;8  - Character write.
0530  0031 R                    DW      PRN_WRT         ;9  - Character write with verify.
0532  0015 R                    DW      PRN_STA         ;10 - Character write status.
0534  05C7 R                    DW      EXIT            ;11 - (Not used.)
0536  05C7 R                    DW      EXIT            ;12 - IO Control.
```

IO.SYS

```
                             ;Define offsets for io data packet

                             IODAT   STRUC
0000  ??                     CMDLEN  DB      ?              ;LENGTH OF THIS COMMAND
0001  ??                     UNIT    DB      ?              ;SUB UNIT SPECIFIER
0002  ??                     CMD     DB      ?              ;COMMAND CODE
0003  ????                   STATUS  DW      ?              ;STATUS
0005    08 [                         DB      8 DUP (?)
             ??
                 ]

000D  ??                     MEDIA   DB      ?              ;MEDIA DESCRIPTOR
000E  ????????               TRANS   DD      ?              ;TRANSFER ADDRESS
0012  ????                   COUNT   DW      ?              ;COUNT OF BLOCKS OR CHARACTERS
0014  ????                   BEGIN   DW      ?              ;FIRST BLOCK TO TRANSFER
0016                         IODAT   ENDS

0538  00 00 00 00            PTRSAV  DD      0              ;Strategy pointer save.

                             ;
                             ; Simplistic Strategy routine for non-multi-Tasking system.
                             ;
                             ;   Currently just saves I/O packet pointers in PTRSAV for
                             ;   later processing by the individual interrupt routines.
                             ;

053C                         STRATP  PROC    FAR

053C                         STRATEGY:
053C  2E: 89 1E 0538 R               MOV     WORD PTR CS:[PTRSAV],BX
0541  2E: 8C 06 053A R               MOV     WORD PTR CS:[PTRSAV+2],ES
0546  CB                     RE_INIT: RET    ;FAR RETURN FOR SYSINIT CALL

0547                         STRATP  ENDP

                             ;
                             ; Console interrupt routine for processing I/O packets.
                             ;

0547                         CON_INT:
0547  56                             PUSH    SI
0548  BE 04D0 R                      MOV     SI,OFFSET CONTBL
054B  EB 10                          JMP     SHORT ENTRY

                             ;
                             ; Auxilary interrupt routine for processing I/O packets.
                             ;

054D                         AUX_INT:
054D  56                             PUSH    SI
054E  BE 04EA R                      MOV     SI,OFFSET AUXTBL
0551  EB 0A                          JMP     SHORT ENTRY

                             ;
                             ; Printer interrupt routine for processing I/O packets.
                             ;

0553                         PRN_INT:
```

```
0553  56                           PUSH    SI
0554  BE 051E R                    MOV     SI,OFFSET PRNTBL
0557  EB 04                        JMP     SHORT ENTRY

                            ;
                            ; Clock interrupt routine for processing I/O packets.
                            ;

0559                        TIM_INT:
0559  56                           PUSH    SI
055A  BE 0504 R                    MOV     SI,OFFSET TIMTBL
                            ;
                            ; Common program for handling the          I/O packet
                            ;   processing scheme in MSDOS 2.0
                            ;

055D  50                    ENTRY: PUSH    AX              ;Save all necessary registers.
055E  51                           PUSH    CX
055F  52                           PUSH    DX
0560  57                           PUSH    DI
0561  55                           PUSH    BP
0562  1E                           PUSH    DS
0563  06                           PUSH    ES
0564  53                           PUSH    BX

0565  2E: C5 1E 0538 R             LDS     BX,CS:[PTRSAV]  ;Retrieve pointer to I/O Packet.

056A  8A 47 01                     MOV     AL,[BX.UNIT]    ;AL = Unit code.
056D  8A 67 0D                     MOV     AH,[BX.MEDIA]   ;AH = Media descriptor.
0570  8B 4F 12                     MOV     CX,[BX.COUNT]   ;CX = Contains byte/sector count.
0573  8B 57 14                     MOV     DX,[BX.BEGIN]   ;DX = Starting Logical sector.

0576  97                           XCHG    DI,AX           ;Move Unit & Media into DI temporarily.
0577  8A 47 02                     MOV     AL,[BX.CMD]     ;Retrieve Command type. (1 => 11)
057A  32 E4                        XOR     AH,AH           ;Clear upper half of AX for calculation.
057C  03 F0                        ADD     SI,AX           ;Compute entry pointer in dispatch table.
057E  03 F0                        ADD     SI,AX
0580  3C 0B                        CMP     AL,11           ;Verify that not more than 11 commands.
0582  77 18                        JA      ERROR_3         ;Ah, well, error out.
0584  97                           XCHG    AX,DI           ;Move Unit & Media back where they belong.
0585  C4 7F 0E                     LES     DI,[BX.TRANS]   ;DI contains addess of Transfer address.
                                                           ;ES contains segment.
0588  0E                           PUSH    CS
0589  1F                           POP     DS              ;Data segment same as Code segment.
058A  FF 24                        JMP     [SI]            ;Perform I/O packet command.
```

IO.SYS

```
058C                    BUS_EXIT:                      ;Device busy exit.
058C  B4 03                     MOV     AH,00000011B   ;Set busy and done bits.
058E  EB 39                     JMP     SHORT EXIT1


                        ;
                        ;  Common error processing routine.
                        ;   AL contains actual error code.
                        ;
                        ;   Error # 0 = Write Protect violation.
                        ;           1 = Unkown unit.
                        ;           2 = Drive not ready.
                        ;           3 = Unknown command in I/O packet.
                        ;           4 = CRC error.
                        ;           5 = Bad drive request structure length.
                        ;           6 = Seek error.
                        ;           7 = Unknown media discovered
                        ;           8 = Sector not found.
                        ;           9 = Printer out of paper.
                        ;          10 = Write fault.
                        ;          11 = Read fault.
                        ;          12 = General failure.
                        ;

0590                    ERROR_0:
0590  32 C0                     XOR     AL,AL          ;Write protect violation.
0592  EB 2E                     JMP     SHORT ERR_EXIT
0594                    ERROR_1:
0594  B0 01                     MOV     AL,1           ;Unknown unit.
0596  EB 2A                     JMP     SHORT ERR_EXIT
0598                    ERROR_2:
0598  B0 02                     MOV     AL,2           ;Drive not ready.
059A  EB 26                     JMP     SHORT ERR_EXIT
059C                    ERROR_3:
059C  B0 03                     MOV     AL,3           ;Unknown command in I/O packet.
059E  EB 22                     JMP     SHORT ERR_EXIT
05A0                    ERROR_4:
05A0  B0 04                     MOV     AL,4           ;CRC error.
05A2  EB 1E                     JMP     SHORT ERR_EXIT
05A4                    ERROR_5:
05A4  B0 05                     MOV     AL,5           ;Bad drive request structure length.
05A6  EB 1A                     JMP     SHORT ERR_EXIT
05A8                    ERROR_6:
05A8  B0 06                     MOV     AL,6           ;Seek error.
05AA  EB 16                     JMP     SHORT ERR_EXIT
05AC                    ERROR_7:
05AC  B0 07                     MOV     AL,7           ;Unknown media discovered.
05AE  EB 12                     JMP     SHORT ERR_EXIT
05B0                    ERROR_8:
05B0  B0 08                     MOV     AL,8           ;Sector not found.
05B2  EB 0E                     JMP     SHORT ERR_EXIT
05B4                    ERROR_9:
05B4  B0 09                     MOV     AL,9           ;Printer out of paper.
05B6  EB 0A                     JMP     SHORT ERR_EXIT
05B8                    ERROR_10:
05B8  B0 0A                     MOV     AL,10          ;Write fault.
05BA  EB 06                     JMP     SHORT ERR_EXIT
05BC                    ERROR_11:
```

```
05BC  B0 0B                          MOV    AL,11          ;Read fault.
05BE  EB 02                          JMP    SHORT ERR_EXIT
05C0                         ERROR_12:
05C0  B0 0C                          MOV    AL,12          ;General failure.
                             ;
                             ;fall through to ERR_EXIT
                             ;

05C2                         ERR_EXIT:
05C2  B4 81                          MOV    AH,10000001B   ;Set error and done bits.
05C4  F9                             STC                   ;Set carry bit also.
05C5  EB 02                          JMP    SHORT EXIT1    ;Quick way out.

05C7                         EXITP   PROC   FAR            ;Normal exit for device drivers.

05C7  B4 01                  EXIT:   MOV    AH,00000001B   ;Set done bit for MSDOS.
05C9  2E: C5 1E 0538 R       EXIT1:  LDS    BX,CS:[PTRSAV]
05CE  89 47 03                       MOV    [BX.STATUS],AX ;Save operation complete and status.

05D1  5B                             POP    BX             ;Restore registers.
05D2  07                             POP    ES
05D3  1F                             POP    DS
05D4  5D                             POP    BP
05D5  5F                             POP    DI
05D6  5A                             POP    DX
05D7  59                             POP    CX
05D8  58                             POP    AX
05D9  5E                             POP    SI
05DA  CB                             RET                         ;RESTORE REGS AND RETURN
05DB                         EXITP   ENDP
```

```
                            ;
                            ; INTERRUPT 29 HANDLER
                            ;
                            ; ENTRY: CHARACTER TO BE OUTPUT TO CRT IN AL
                            ; EXIT:  ALL REGISTERS EXCEPT BX PRESERVED
                            ;
05DB                        I29_HANDLER:
05DB  56                           PUSH   SI
05DC  50                           PUSH   AX
05DD  51                           PUSH   CX
05DE  52                           PUSH   DX
05DF  57                           PUSH   DI
05E0  55                           PUSH   BP
05E1  1E                           PUSH   DS
05E2  06                           PUSH   ES
05E3  0E                           PUSH   CS
05E4  1F                           POP    DS
05E5  C6 06 0609 R 29              MOV    BYTE PTR INTTYPE,29H    ;FLAG INT 29H ENTRY
05EA  E8 07C0 R                    CALL   CONOUT                  ;CHARACTER IN AL TO CRT
05ED  C6 06 0609 R 00              MOV    BYTE PTR INTTYPE,0      ;CLEAR INT 29H FLAG
05F2  07                           POP    ES
05F3  1F                           POP    DS
05F4  5D                           POP    BP
05F5  5F                           POP    DI
05F6  5A                           POP    DX
05F7  59                           POP    CX
05F8  58                           POP    AX
05F9  5E                           POP    SI
05FA  CF                           IRET
                            ;
                            ; TRAP FOR UNDEFINED INTERRUPTS
                            ;
05FB                        INT_TRAP:
05FB  CF                           IRET                 ;RETURN ..NO ACTION
                            ;
                            ; Console output routine.
                            ;

05FC                        CON_WRIT:
05FC  8B F7                        MOV    SI,DI         ;Get destination to source.
05FE                        CON_WRI1:
05FE  26: AC                       LODS   BYTE PTR ES:[SI]
0600  51                           PUSH   CX
0601  E8 07C0 R                    CALL   CONOUT        ;Call ansi driver.
0604  59                           POP    CX
0605  E2 F7                        LOOP   CON_WRI1      ;Keep going until user buffer through.
0607  EB BE                        JMP    EXIT
```

```
                                        ;
                                        ;ANSI Info and routines. ANSI driver implemented as a finite state automata
                                        ;This ANSI driver translates the ANSI standard escape sequences into the
                                        ;NCR DECISION MATE 5 escape sequences. DM 5 sequences are also supported.
                                        ;This is not a full implementation of ANSI, but rather a minimal implementation
                                        ; which implements all of the necessary ANSI functions.
                                        ;

= 0008                                  POSCUR  EQU     8               ;CRTPIM ESC FOR CURSOR POSITIONING
= 0018                                  ROWS    EQU     24              ;SCREEN HIGHT
= 0050                                  SWIDTH  EQU     80              ;SCREEN WIDTH
= 001B                                  ESC     EQU     1BH             ;Escape character used in this implementation.
0609  00                                INTTYPE DB      0               ;INTERRUPT TYPE FLAG FOR INT 29H HANDLER
060A  0000                              RADDR   DW      0               ;ADDRESS OF ROUTINE REQUESTING DATA
060C  E8                                MON_ATT DB      0E8H            ;MONOCHROME CRT DEFAULT ATTRIBUTE         *1*
060D  4D                                MONO_COLOR DB   'M'             ;MONOCHROME OR COLOR FLAG -- DEFAULT MONO
                                                                        ; ACCORDINGLY                  *13*
060E  00                                BG_FG       DB  0               ;BIT 0 --) BACKGROUND DEFINED
                                                                        ; BIT 1 --) FOREGROUND DEFINED  *13*
060F  00                                DREQ    DB      0               ;DATA REQUEST FLAG
0610  0000                              SAVCRTPARB DW   0               ;SAVE / RESTORE CURSOR POSITION           *2*
0612  00                                STRINGF DB      0               ;String flag
0613  091E R                            CHTRANS DW      OFFSET SECREQ   ;FIRST SETUP CRTACTTBL                    *4*
0615  07C5 R                            STATE   DW      ST1             ;Current ANSI character state.
0617  0619 R                            PRMPNT  DW      PARMS           ;Current parameter pointer.
0619  0100 [                            PARMS   DB      256 DUP(0)      ;Allow for up to 257 parameters.
                00
                    ]

0719  00                                LASTPRM DB      0               ;With this being the last one.
071A     A0 [                           LINBUF  DB      160 DUP(' ')    ;BUFFER FOR CRTPIM
                20
                    ]


                                        ;
                                        ; CRT PARAMETER BLOCK
                                        ;
07BA  00                                CRTPARB DB      0       ;COLUMN
07BB  00                                        DB      0       ;ROW
07BC  E8                                        DB      0E8H    ;ATTRIBUTE
07BD  00                                        DB      0       ;ESCAPE CODE BYTE
07BE  00                                        DB      0       ;FREQUENCY
07BF  00                                        DB      0       ;TONE LENGTH
```

IO.SYS

```
                                ;
                                ;
                                ; ANSI console output driver.
                                ;

07C0  BF 0615 R                 CONOUT: MOV     DI,OFFSET STATE ;Retrieve current ansi state.
07C3  FF 25                             JMP     [DI]            ;Jump to it.

                                ;
                                ; State one (1).
                                ;   Looks for an Escape character.
                                ;

07C5  3C 1B                     ST1:    CMP     AL,ESC          ;See if this the first character is ESC.
07C7  74 03                             JZ      ST12            ;No, treat as regular character output.
07C9  E9 0897 R                         JMP     OUTCTR          ;OUTPUT THE CHARACTER
07CC  C7 05 07D1 R              ST12:   MOV     WORD PTR [DI],OFFSET ST2        ;Yes, setup state two.
07D0  C3                                RET

                                ;
                                ; State two (2).
                                ;   Looks for the "[" character.
                                ;

07D1  3C 5B                     ST2:    CMP     AL,'['          ;See if a valide state two.
07D3  74 03                             JZ      ST21            ;SKIP IF '[`
07D5  E9 086B R                         JMP     ESCNONA         ;No, test non_ANSI ESC sequence
07D8  BB 0619 R                 ST21:   MOV     BX,OFFSET PARMS ;Yes, get parameter pointer.
07DB  89 1E 0617 R                      MOV     WORD PTR [PRMPNT],BX    ;Setup in pointer index.
07DF  C7 07 0000                        MOV     WORD PTR [BX],0 ;Clear first entry.
07E3  C7 05 07E8 R                      MOV     WORD PTR [DI],OFFSET ST3;Setup for state three.
07E7  C3                                RET
```

```
                                ;
                                ; State three (3).
                                ;   Entered one or more times for parameter passing.
                                ;

07E8  F6 06 0612 R 22           ST3:    TEST    STRINGF,'"'             ;TEST IF STRING PROCESSING
07ED  75 04                             JNZ     ST31                    ; JUMP IF STRING FLAG SET
07EF  3C 3B                             CMP     AL,';'                  ;Look for decimal # seperator.
07F1  74 13                             JZ      ST3C                    ;Fall through if not ";"
07F3  3C 22                     ST31:   CMP     AL,'"'                  ;Look for string separator.
07F5  75 27                             JNZ     ST3A                    ;No check phase A
07F7  30 06 0612 R                      XOR     STRINGF,AL              ;Toggle string flag
07FB  84 06 0612 R                      TEST    STRINGF,AL              ;
07FF  75 04                             JNZ     ST3RET
0801  FF 0E 0617 R                      DEC     WORD PTR [PRMPNT]       ;Adjust pointer at end of string.
0805  C3                        ST3RET: RET
0806  FF 06 0617 R              ST3C:   INC     WORD PTR [PRMPNT]       ;Yes, incr. pointer to next param.
080A  B8 0719 R                         MOV     AX,OFFSET LASTPRM       ;Check for outside parameter list.
080D  39 06 0617 R                      CMP     [PRMPNT],AX
0811  76 03                             JBE     RETST3          ;Yes, proceed with next parameter.
0813  A3 0617 R                         MOV     [PRMPNT],AX     ;No, treat as extentsion to old.
0816  8B 3E 0617 R              RETST3: MOV     DI,[PRMPNT]     ;Setup for next parameter.
081A  C6 05 00                          MOV     BYTE PTR [DI],0 ;Pre-Initialize it to zero.
081D  C3                                RET

                                ;
                                ; State three A (3A).
                                ;   Check for a ascii digit.
                                ;

081E  F6 06 0612 R 22           ST3A:   TEST    STRINGF,'"'     ;TEST IF STRING PROCESSING
0823  75 17                             JNZ     ST3B            ; JUMP IF THRU
0825  3C 30                             CMP     AL,'0'          ;Check for ASCII digit.
0827  72 1E                             JB      ST3D            ;No, check for seconday command character.
0829  3C 39                             CMP     AL,'9'          ;Still checking for ASCII digit.
082B  77 1A                             JA      ST3D            ;No, it must be a secondary.
082D  2C 30                             SUB     AL,'0'          ;Convert to binary.
082F  8B 3E 0617 R                      MOV     DI,[PRMPNT]     ;Get the current parameter pointer.
0833  86 05                             XCHG    [DI],AL         ;Get existing #.
0835  B4 0A                             MOV     AH,10           ;Scale by 10.
0837  F6 E4                             MUL     AH
0839  00 05                             ADD     [DI],AL         ;Add to new digit.
083B  C3                                RET

                                ;
                                ; State three B (3B).
                                ;   Wasn't a ascii digit, so check for string flag and secondary command.
                                ;

083C  8B 3E 0617 R              ST3B:   MOV     DI,[PRMPNT]             ;GET POINTER TO PARMS
0840  88 05                             MOV     BYTE PTR [DI],AL        ;Store character in PARMS
0842  FF 06 0617 R                      INC     WORD PTR [PRMPNT]       ;and move pointer.
0846  C3                                RET
0847  C7 05 07C5 R              ST3D:   MOV     [DI],OFFSET ST1         ;Preset STATE to state 1 just in case.
084B  8B 0E 0617 R                      MOV     CX,[PRMPNT]             ;CX = CURRENT POINTER VALUE
084F  BF 0618 R                         MOV     DI,OFFSET PARMS-1       ;Get pointer to start of parameters.
0852  89 3E 0617 R                      MOV     [PRMPNT],DI             ;Save it in Parameter pointer.
0856  BF 023A R                         MOV     DI,OFFSET CMDTABL-3     ;Get start of Secondary command table.
```

IO.SYS

```
0859  83 C7 03                  ST3B1:  ADD     DI,3             ;Update Command table pointer.
085C  80 3D 00                          CMP     BYTE PTR [DI],0 ;Check for end of table.
085F  75 03                             JNZ     ST3B2            ;No, continue processing.
0861  EB 34 90                          JMP     OUTCTR           ;Yes, treat as regular character.
0864  3A 05                     ST3B2:  CMP     AL,[DI]          ;Check for valid. command.
0866  75 F1                             JNZ     ST3B1            ;No, keep checking.
0868  FF 65 01                          JMP     [DI+1]           ;Yes, transfer to that secondary command.
                                ;
                                ; NON ANSI ESCAPE SEQUENCE
                                ;

086B                            ESCNONA:        ;AL=CHARACTER
086B  F6 06 060F R FF                   TEST    DREQ,0FFH        ;SEE IF ANY SEQUENCE PENDING
0870  75 20                             JNZ     PASSCH           ;PASS BYTE TO REQUESTING ROUTINE
0872  8B 0E 0285 R                      MOV     CX,ETBLENT       ;CX=NUMBER OF ESCAPE TABLE ENTRIES
0876  BB 0287 R                         MOV     BX,OFFSET ESCTBL ;BX POINTS TO ESC TABLE
0879  3A 07                     ESCNON: CMP     AL,[BX]          ;SEARCH FOR CORRESPONDING ENTRY
087B  75 09                             JNZ     ESCFAL            ;NO MATCH THIS TIME
087D  C7 06 0615 R 086B R               MOV     STATE,OFFSET ESCNONA ;NON ANSII ESCAPE SEQUENCE
0883  FF 67 01                          JMP     [BX+1]            ;MATCH FOUND EXIT TO CORR. ROUTINE
0886  83 C3 03                  ESCFAL: ADD     BX,3             ;BUMP POINTER TO NEXT ENTRY
0889  E2 EE                             LOOP    ESCNON           ;LOOP UNTIL TABLE ENDS
088B  C7 06 0615 R 07C5 R               MOV     STATE,OFFSET ST1 ;RETURN TO STATE 1
0891  C3                                RET                      ;ESCAPE FUNCTION NOT IMPLEMENTED
                                ;
0892  BF 060A R                 PASSCH: MOV     DI,OFFSET RADDR ;RADDR CONTAINS REQUESTERS ADDRESS
0895  FF 25                             JMP     [DI]                   ;PASS THE CHARACTER IN AL
                                ;
                                ; CHARACTER OUTPUT ROUTINE
                                ;
0897                            OUTCTR:                          ;                            *1* *2*
0897  3C 20                             CMP     AL,' '
0899  72 0C                             JB      CCNTCH           ;CHARACTERS < 20H ARE CONTROL CHARACTERS
089B  FF 16 0613 R                      CALL    [CHTRANS]        ;TRANSLATE CHARACTER >= 20H
089F  86 C8                             XCHG    CL,AL   ;CRTPIM NEEDS THE CHARACTER IN CL
08A1  BB 07BA R                         MOV     BX,OFFSET CRTPARB ;POINTER TO CRT PARAMETER BLOCK
08A4  E9 0EBA R                         JMP     HIP_OUT          ;HIGH SPEED ENTRY OF CRT PIM
```

```
                                 ;
                                 ; SEARCH COMTBL FOR SINGLE BYTE CONTROL FUNCTION
                                 ;
08A7                             CCMTCH:
08A7  8B 0E 0265 R                     MOV     CX,CTABLEN      ;CX=LENGTH OF COMTBL
08AB  BB 0267 R                        MOV     BX,OFFSET COMTBL;POINTER TO COMTBL
08AE  3A 07                      CNT:  CMP     AL,BYTE PTR [BX]
08B0  74 06                            JZ      CNTFD           ;JMP IF MATCH FOUND
08B2  83 C3 03                         ADD     BX,3            ;MOVE POINTER TO NEXT ENTRY
08B5  E2 F7                            LOOP    CNT             ;LOOP TILL MATCH OR END OF TABLE
08B7  C3                               RET                     ;RET IF NOT IMPLEMENTED
                                 ;
08B8                             CNTFD:
08B8  FF 67 01                         JMP     [BX+1]  ;START PROCESSING
                                 ;
                                 ; CHARACTER TRANSLATION ROUTINE
                                 ;
08BB                             CHRTRN:
08BB  8A 16 0000 E                     MOV     DL,language     ;DL=LANGUAGE CODE FROM KBD DRIVER
08BF  80 FA 32                         CMP     DL,32H          ;TEST FOR HEBREW                    *10*
08C2  74 54                            JZ      HEBREW          ;JUMP IF TRUE                       *10*
08C4  BB 02B4 R                        MOV     BX,OFFSET CRTTBL ;POINTER TO CRT TRANSLATION TABLE
08C7  F6 C2 10                         TEST    DL,10H          ;SEE IF COUNTRY GROUP I
08CA  74 0C                            JZ      NOTI            ; JUMP IF NOT
08CC  80 FA 12                         CMP     DL,12H          ;SET 12H TO 05H
08CF  75 02                            JNZ     NOTII           ;SKIP IF NOT 12H
08D1  B2 15                            MOV     DL,15H          ;12H AND 05H ARE THE SAME
08D3  80 F2 10                   NOTII: XOR    DL,10H          ;OTHERS OF 1XH SAME AS CORR..0XH
08D6  EB 1A                            JMP     SHORT LANT      ;EXIT ..LANGUAGE CODE CONVERTED
08D8  F6 C2 20                   NOTI:  TEST   DL,20H          ;SEE IF COUNTRY GROUP II
08DB  74 15                            JZ      LANT            ; EXIT IF NOT
08DD  80 FA 22                         CMP     DL,22H          ;MAP 20H AND 21H TO 08H
08E0  73 04                            JNB     GR21            ;JUMP IF 22H OR GREATER
08E2  B2 08                            MOV     DL,08H
08E4  EB 0C                            JMP SHORT LANT
08E6                             GR21:
08E6  75 04                            JNZ     GR22            ;JUMP IF ABOVE 22H
08E8  B2 02                            MOV     DL,02H          ;MAP 22H TO 02H
08EA  EB 06                            JMP SHORT LANT
08EC                             GR22:
08EC  80 C2 06                         ADD     DL,6            ;MAP 23H .. 27H TO 09H .. 0DH
08EF  80 E2 0F                         AND     DL,0FH
                                 ;
                                 ; DL CONTAINS TRANSLATED LANGUAGE CODE NOW
                                 ;
08F2  FE CA                      LANT:  DEC    DL              ;SETUP POINTER TO ASSOC. COUNTRY TABLE
08F4  78 08                            JS      LPOISET
08F6  32 ED                            XOR     CH,CH
08F8  8A 0F                            MOV     CL,BYTE PTR [BX] ;MOVE POINTER BY THE VALUE FOUND IN TABLE
08FA  03 D9                            ADD     BX,CX
08FC  EB F4                            JMP SHORT LANT          ;LOOP TILL LANGUAGE FOUND
08FE  8A 0F                      LPOISET: MOV  CL,BYTE PTR [BX] ;CL=LENGTH OF COUNTRY TABLE
0900  B5 00                            MOV     CH,0            ;CX=LENGTH NOW
0902  FE C9                            DEC     CL
0904  D0 E9                            SHR     CL,1            ;(LENGTH-1)/2 = NUMBER OF ENTRIES
0906  43                               INC     BX              ;[BX] = POINTER TO FIRST ENTRY
0907                             SETACTTBL:
0907  BF 0376 R                        MOV     DI,OFFSET CRTACTTBL-20H
```

IO.SYS

```
090A  8A 17                     MOV    DL,BYTE PTR [BX] ;SET POINTER INTO CRTACTTBL
090C  B6 00                     MOV    DH,0
090E  03 FA                     ADD    DI,DX
0910  43                        INC    BX                ;[BX] = TABLE POINTER FOR THIS BYTE
0911  8A 27                     MOV    AH,BYTE PTR [BX] ;TRANSLATED CHARACTER
0913  88 25                     MOV    BYTE PTR [DI],AH
0915  43                        INC    BX                ;MOVE POINTER TO NEXT ENTRY
0916  E2 EF                     LOOP   SETACTTBL         ;LOOP TILL ALL REQUIRED CHANGES MADE
0918  C7 06 0613 R 091E R   HEBREW: MOV    CHTRANS,OFFSET SECREQ   ;SET TRANSLATION ROUTINE ADDRESS*10*
                            ;
                            ; CRTACTTBL SETUP NOW
                            ;
091E                        SECREQ:
091E  F6 06 0000 E 02           TEST   flag_buf,02H     ;TEST HEBREW ACTIVE                  *10*
0923  75 05                     JNZ    HEB_ACTIVE       ;JUMP IF ACTIVE                      *10*
0925  BB 0376 R                 MOV    BX,OFFSET CRTACTTBL-20H ;20H MAPPED TO TABLE OFFSET 00
0928  D7                        XLAT                    ;[BX+AL] --> AL
0929  C3                        RET                     ;TRANSLATED CHARACETR IN AL
092A                        HEB_ACTIVE:
092A  3C 60                     CMP    AL,60H           ;CHARACTER CODES 60H TO 7AH ARE TRANSLATED
                                                        ; TO 00H TO 1AH ACCORDINGLY          *10*
092C  72 06                     JB     HEB_NOT          ;JUMP IF < 60H                       *10*
092E  3C 7A                     CMP    AL,7AH           ;TEST FOR > 7AH                      *10*
0930  77 02                     JA     HEB_NOT          ;JUMP IF > 7AH                       *10*
0932  2C 60                     SUB    AL,60H           ;FOR AL = 60H TO 7AH SUBTRACT 60H    *10*
0934                        HEB_NOT:
0934  C3                        RET                     ;                                    *10*
                            ;
                            ;
                            ; Get binary parameter from storage and return a one if = 0
                            ;

0935  E8 0942 R             GETONE: CALL   GETPARM          ;Get parameter form list.
0938  0A C0                     OR     AL,AL            ;Verify for non-zero.
093A  75 02                     JNZ    GETRET           ;Good, then return to caller.
093C  FE C0                     INC    AL               ;Bad, make it at least a one.
093E  98                    GETRET: CBW                     ;Sign extend AL.
093F  8B C8                     MOV    CX,AX            ;Copy of it to CX.
0941  C3                        RET

0942  FF 06 0617 R          GETPARM:INC    WORD PTR [PRMPNT]       ;Increment parameter pointer.
0946  8B 3E 0617 R          GOTPARM:MOV    DI,[PRMPNT]      ;Get parameter pointer.
094A  8A 05                     MOV    AL,[DI]          ;Get parameter value.
094C  C3                        RET
```

```
                                ;
                                ; Cursor Positioning routines.
                                ;
094D  B3 41                     CUU:    MOV    BL,'A'          ;Cursor up.
094F  EB 0A                             JMP    SHORT CURPOS
0951  B3 42                     CUD:    MOV    BL,'B'          ;Cursor down.
0953  EB 06                             JMP    SHORT CURPOS
0955  B3 43                     CUF:    MOV    BL,'C'          ;Cursor forward.
0957  EB 02                             JMP    SHORT CURPOS
0959  B3 44                     CUB:    MOV    BL,'D'          ;Cursor back.

095B  E8 0935 R                 CURPOS: CALL   GETONE          ;Get number of positions to move into CX.
095E  A1 07BA R                         MOV    AX,WORD PTR CRTPARB ;Get current cursor position
                                                                 ; AL=COLUMN..AH=ROW
0961  80 FB 41                          CMP    BL,'A'          ;Cursor up
0964  75 15                             JNZ    CDOWN
0966  2A E1                     CUP1:   SUB    AH,CL
0968  73 02                             JNB    CMOVEEX         ;OK..NEW POSITION ON SCREEN
096A  32 E4                             XOR    AH,AH           ;STOP ON LINE 1..NO SCROLLING
096C                            CMOVEEX:       ;SET UP CRTPARB AND CALL CRTPIM
096C  A3 07BA R                         MOV    WORD PTR CRTPARB,AX     ;SET NEW COLUMN AND ROW
096F  C6 06 07BD R 08                   MOV    CRTPARB+3,POSCUR        ;POSCUR=POSITION CURSOR ONLY (08H)
0974  BB 07BA R                 CNTC1B: MOV    BX,OFFSET CRTPARB       ;BX=POINTER TO CRTPARB
0977  E8 0E58 R                         CALL   CRTPIM                  ;EXECUTE CURSOR POSITIONING
097A  C3                                RET

097B  80 FB 42                  CDOWN:  CMP    BL,'B'          ;Cursor down
097E  75 0B                             JNZ    CFORW           ;SKIP IF NOT
0980  02 E1                             ADD    AH,CL           ;SEE IF WE REACH LOWER SCREEN BOUNDARY
0982  80 FC 18                          CMP    AH,ROWS         ;WE STOP AT BOTTOM LINE (25)
0985  7E E5                             JNG    CMOVEEX         ;EXECUTE IF NOT BELOW BOTTOM LINE
0987  B4 18                             MOV    AH,ROWS         ; ELSE FORGET REST OF COUNT
0989  EB E1                             JMP    SHORT CMOVEEX   ; SET CURSOR TO BOTTOM LINE
                                ;
098B  80 FB 43                  CFORW:  CMP    BL,'C'          ;Cursor forward
098E  75 0A                             JNZ    CBACK
0990  02 C1                             ADD    AL,CL           ;CURSOR FORWARD
0992  3C 50                             CMP    AL,SWIDTH       ;SEE IF MOVE WOULD GO TO OUTSIDE SCREEN
0994  72 D6                             JB     CMOVEEX         ; NO..MOVE
0996  B0 4F                             MOV    AL,SWIDTH-1     ; YES..STOP AT LAST COLUMN
0998  EB D2                             JMP    SHORT CMOVEEX   ;MOVE NOW
                                ;
099A  2A C1                     CBACK:  SUB    AL,CL           ;Cursor back
099C  73 CE                             JNB    CMOVEEX         ;MOVE IT IT'S WITHIN LINE
099E  32 C0                             XOR    AL,AL           ; ELSE SET TO FIRST COLUMN
09AD  EB CA                             JMP    SHORT CMOVEEX
```

```
                              ;
                              ; RING THE BELL
                              ;
09A2                          BELL:
09A2  B1 07                          MOV     CL,7            ;BELL CODE
09A4  E8 0000 E                      CALL    kbd_out         ;HANDLED BY KEYBOARD DRIVER
09A7  C3                             RET
                              ;
                              ; NON DESTRUCTIVE BACKWARD SPACE
                              ;
09A8                          BACKSP:
09A8  A1 07BA R                      MOV     AX,WORD PTR CRTPARB ;AL=COLUMN..AH=ROW
09AB  FE C8                          DEC     AL
09AD  79 0A                          JNS     BKSP1           ;JUMP IF NOT TO PRECEDING ROW
09AF  FE CC                          DEC     AH              ;CHANGE ROW
09B1  79 04                          JNS     BKSP2           ;JUMP IF NOT BEYOND TOP OF SCREEN
09B3  32 E4                          XOR     AH,AH           ; ELSE SET ROW 0
09B5  EB 02                          JMP SHORT BKSP1
09B7  B0 4F                   BKSP2: MOV     AL,SWIDTH-1     ;CURSOR TO END OF LINE
09B9  EB 81                   BKSP1: JMP     CMOVEEX         ;SET CURSOR
                              ;
                              ; LINEFEED
                              ;
09BB                          LINEFD:
09BB  C6 06 07BD R 0B                MOV     CRTPARB+3,0BH   ;LINE FEED ESC CODE FOR CRTPIM
09C0  EB B2                          JMP     CNTC1B          ;SEND ESCAPE CODE
                              ;
                              ; REVERSE LINEFEED
                              ;
09C2                          RLF:
09C2  A1 07BA R                      MOV     AX,WORD PTR CRTPARB
09C5  B1 01                          MOV     CL,1            ;CURSOR ONE LINE UP
09C7  32 C0                          XOR     AL,AL           ;RETURN TO COLUMN ZERO
09C9  EB 9B                          JMP     CUP1
                              ;
                              ; NON DESTRUCTIVE FORWARD SPACE
                              ;
09CB                          NDFS:
09CB  A1 07BA R                      MOV     AX,WORD PTR CRTPARB
09CE  FE C0                          INC     AL              ;COLUMN +1
09D0  3C 50                          CMP     AL,SWIDTH       ;TEST IF BEYOND SCREEN
09D2  73 02                          JNB     NDFS1           ; IF YES..JUMP
09D4  EB 96                          JMP     CMOVEEX         ;SET CURSOR
09D6  EB E3                   NDFS1: JMP     LINEFD          ;PERFORM LINEFEED
                              ;
                              ; CARRIAGE RETURN
                              ;
09D8                          CARRET:
09D8  A1 07BA R                      MOV     AX,WORD PTR CRTPARB
09DB  32 C0                          XOR     AL,AL           ;SET COLUMN 0
09DD  EB 8D                          JMP     CMOVEEX         ;SET CURSOR
```

```
                                ;
                                ; HOME CURSOR
                                ;
09DF                            VHOME:
09DF  33 C0                            XOR    AX,AX          ;COLUMN 0 ROW ZERO
09E1  EB 89                            JMP    CMOVEEX        ;SET CURSOR
                                ;
                                ; SET HALF INTENSITY
                                ;
09E3                            SHALF_INT:
09E3  F6 06 060C R 04                  TEST   MON_ATT,04H    ;SEE IF HALF INTENSITY ALREADY SET
09E8  75 41                            JNZ    S_STAT1        ;JUMP IF TRUE
09EA  80 0E 060C R 04                  OR     MON_ATT,04H    ;SET HALF INTENSITY FLAG
09EF  80 3E 060D R 43                  CMP    MONO_COLOR,'C' ;TEST FOR COLOR CRT
09F4  75 19                            JNZ    SHALF1         ;JUMP IF NOT
                                       ;
                                       ; COLOR CRT
                                ;
09F6                            REAC_HI:
09F6  F6 06 060C R 01                  TEST   MON_ATT,01H    ;TEST FOR INVERSE ACTIVE
09FB  75 1A                            JNZ    SHALFBG        ;JUMP IF TRUE
09FD  F6 06 07BC R 04                  TEST   CRTPARB+2,04H  ;TEST FOR RED ALREADY SET IN FG
0A02  74 03                            JZ     SHALF2         ;JUMP IF NOT
0A04  EB 25 90                         JMP    S_STAT1        ;RETURN TO STATE1 NO ACTION
0A07                            SHALF2:
0A07  80 0E 07BC R 05                  OR     CRTPARB+2,05H  ;SET RED FG BIT + HALF INTENSITY
0A0C  E9 0CF1 R                        JMP    RETSTAT1       ;SET ATTRIBUTE AND RETURN TO STATE1
0A0F                            SHALF1:
0A0F  80 0E 07BC R 04                  OR     CRTPARB+2,04H  ;SET HALF INTENSITY FOR MONOCHROME
0A14  E9 0CF1 R                        JMP    RETSTAT1
0A17                            SHALFBG:
0A17  F6 06 07BC R 20                  TEST   CRTPARB+2,20H  ;TEST FOR RED IN BG
0A1C  74 0D                            JZ     S_STAT1        ;JUMP IF RED SET
0A1E  80 26 07BC R DF                  AND    CRTPARB+2,0DFH ;SET RED BIT IN BG
0A23  80 0E 07BC R 01                  OR     CRTPARB+2,01H  ;SET HALF INTENSITY
0A28  E9 0CF1 R                        JMP    RETSTAT1
                                ;
                                ; SET STATE 1 AGAIN
                                ;
0A2B                            S_STAT1:
0A2B  C7 06 0615 R 07C5 R              MOV    STATE,OFFSET ST1
0A31  C3                               RET
                                ;
                                ; RESET HALF INTENSITY
                                ;
0A32                            RHALF_INT:
0A32  F6 06 060C R 04                  TEST   MON_ATT,04H    ;TEST FOR HALF INTENSITY ACTIVE
0A37  74 F2                            JZ     S_STAT1        ; NO ACTION IF NOT
0A39  80 26 060C R FB                  AND    MON_ATT,0FBH   ;RESET HALF INTENSITY FLAG
0A3E  80 3E 060D R 43                  CMP    MONO_COLOR,'C' ;TEST FOR COLOR CRT
0A43  74 08                            JZ     RHALFC         ; JUMP IF TRUE
                                ;
                                ; MONOCHROME CRT
                                ;
0A45                            RHALF1:
0A45  80 26 07BC R FB                  AND    CRTPARB+2,0FBH ;RESET HALF INTENSITY IN ATTRIBUTE
0A4A  E9 0CF1 R                        JMP    RETSTAT1       ;EXEC SETTING
                                ;
```

IO.SYS

```
                                      ; COLOR CRT
                                      ;
      0A4D                            RHALFC:
      0A4D  F6 06 07BC R 01                   TEST    CRTPARB+2,01H   ;TEST FOR HALF INTENSITY ACTIVE
      0A52  74 D7                             JZ      S_STAT1         ; IF ZERO --> NO ACTION
      0A54  80 26 07BC R FE                   AND     CRTPARB+2,0FEH  ;RESET COLOR HALF INTENSITY BIT
      0A59  F6 06 060C R 01                   TEST    MON_ATT,01H     ;TEST FOR INVERS ACTIVE
      0A5E  74 E5                             JZ      RHALF1          ; IF ZERO HALF INTENSITY WAS NOT ACTIVE
                                              ;
                                              ; INVERSE IS ACTIVE
                                              ;
      0A60  80 0E 07BC R 20                   OR      CRTPARB+2,20H   ;RESET RED BACKGROUND
      0A65  E9 0CF1 R                         JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
                                      ;
                                      ; SET INVERSE VIDEO
                                      ;
      0A68                            SINV_VIDEO:
      0A68  F6 06 060C R 01                   TEST    MON_ATT,01H     ;TEST FOR INVERS ACTIVE
      0A6D  75 BC                             JNZ     S_STAT1         ;JUMP IF TRUE --> NO ACTION
      0A6F  80 0E 060C R 01                   OR      MON_ATT,01H     ;SET INVERS FLAG
      0A74  80 3E 060D R 43                   CMP     MONO_COLOR,'C'  ;TEST FOR COLOR CRT
*2*   0A79  75 06                             JNZ     SINV_VID1       ;JUMP IF MONOCHROME
                                              ;
                                              ; COLOR CRT
                                              ;
      0A7B  E8 0AAA R                         CALL    REV_COLOR       ;EXCHANGE FG BG
      0A7E  E9 0CF1 R                         JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
*2*   0A81                            SINV_VID1:
      0A81  80 0E 07BC R 01                   OR      CRTPARB+2,01H   ;SET INVERS BIT FOR MONOCHROME CRT
      0A86  E9 0CF1 R                         JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
                                      ;
                                      ; RESET INVERS VIDEO
                                      ;
*2*   0A89                            RINV_VIDEO:
      0A89  F6 06 060C R 01                   TEST    MON_ATT,01H     ;TEST FOR INVERS ACTIVE
      0A8E  74 9B                             JZ      S_STAT1         ; JUMP IF NOT --> NO ACTION
      0A90  80 26 060C R FE                   AND     MON_ATT,0FEH    ;RESET INVERSE FLAG
      0A95  80 3E 060D R 43                   CMP     MONO_COLOR,'C'  ;TEST FOR COLOR CRT
      0A9A  75 06                             JNZ     RINV_VID1       ;JUMP IF MONOCHROME
*2*                                           ;
                                              ; COLOR CRT
                                              ;
      0A9C  E8 0AAA R                         CALL    REV_COLOR       ;EXCHANGE FG BG
      0A9F  E9 0CF1 R                         JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
      0AA2                            RINV_VID1:
*2*   0AA2  80 26 07BC R FE                   AND     CRTPARB+2,0FEH  ;RESET INVERSE BIT FOR MONOCHROME
      0AA7  E9 0CF1 R                         JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
                                      ;
                                      ; SUBROUTINE REVERSING FOREGROUND AND BACKGROUND COLOR
                                      ;
      0AAA                            REV_COLOR:
      0AAA  A0 07BC R                         MOV     AL,CRTPARB+2    ;AL=ATTRIBUTE
      0AAD  8A E0                             MOV     AH,AL           ;COPY TO AH
      0AAF  80 E4 E0                          AND     AH,0E0H         ;SEPARATE BG COLOR BITS
      0AB2  D0 EC                             SHR     AH,1
      0AB4  D0 EC                             SHR     AH,1
      0AB6  D0 EC                             SHR     AH,1            ;BG BITS IN FG POSITION NOW
      0AB8  24 1C                             AND     AL,01CH         ;SEPARATE FG COLOR BITS
      0ABA  D0 E0                             SHL     AL,1
      0ABC  D0 E0                             SHL     AL,1
```

```
 0ABE  D0 E0                          SHL     AL,1            ;FG BITS IN BG POSITION NOW
 0AC0  0A C4                          OR      AL,AH           ;AL HOLDS BOTH NOW
 0AC2  34 FC                          XOR     AL,0FCH         ;REVERSE THEM
 0AC4  80 26 07BC R 03                AND     CRTPARB+2,03H
 0AC9  08 06 07BC R                   OR      CRTPARB+2,AL
 0ACD  C3                             RET
                               ;
                               ; SET BLINKING
                               ;
 0ACE                          SBLINK:
 0ACE  80 0E 07BC R 02                OR      CRTPARB+2,02H   ;SET BLINKING
 0AD3  E9 0CF1 R                      JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
                               ;
                               ; RESET BLINKING
                               ;
 0AD6                          RBLINK:
 0AD6  80 26 07BC R FD                AND     CRTPARB+2,0FDH  ;RESET BLINKING
 0ADB  E9 0CF1 R                      JMP     RETSTAT1        ;ACTIVATE ATTRIBUTE
                               ;
                               ;
                               ; ERASE TO END OF SCREEN
                               ;
 0ADE                          BLEOS:
 0ADE  C6 06 07BD R 02                MOV     CRTPARB+3,2     ;
 0AE3  E9 0B7E R                      JMP     SESC
                               ;
                               ; INSERT LINE
                               ;
 0AE6                          INSLIN:
 0AE6  C6 06 07BD R 04                MOV     CRTPARB+3,4     ;
 0AEB  E9 0B7E R                      JMP     SESC
                               ;
                               ; DELETE LINE
                               ;
 0AEE                          DELLIN:
 0AEE  C6 06 07BD R 05                MOV     CRTPARB+3,5     ;
 0AF3  E9 0B7E R                      JMP     SESC
                               ;
                               ; INSERT CHARACTER
                               ;
 0AF6                          ICHR:
 0AF6  C6 06 07BD R 06                MOV     CRTPARB+3,6     ;
 0AFB  E9 0B7E R                      JMP     SESC
                               ;
                               ; DELETE CHARACTER
                               ;
 0AFE                          DCHR:
 0AFE  C6 06 07BD R 07                MOV     CRTPARB+3,7     ;
 0B03  EB 79 90                       JMP     SESC
```

```
                                ;
                                ; POSITION CURSOR
                                ;
0B06                            POSIT:
0B06  C6 06 060F R FF                   MOV     DREQ,-1         ;SET DATA REQUEST FLAG
0B0B  C7 06 060A R 0B12 R               MOV     RADDR,OFFSET POSC1 ;REQUESTERS ADDRESS
0B11  C3                                RET
0B12                            POSC1:
0B12  2C 20                             SUB     AL,20H          ;ROW 1 TRANSLATED TO ZERO A.S.O.      *9*
0B14  3C 19                             CMP     AL,ROWS+1       ;TEST FOR OUT OF RANGE
0B16  73 03                             JNB     POS1            ; JUMP IF OUT
0B18  A2 07BB R                         MOV     CRTPARB+1,AL    ;SET NEW ROW
0B1B  C7 06 060A R 0B22 R       POS1:   MOV     RADDR,OFFSET POSC2 ;REQUESTERS ADDRESS
0B21  C3                                RET
0B22  2C 20                     POSC2:  SUB     AL,20H          ;COLUMN 1 TRANSLATED TO ZERO A.S.O.  *9*
0B24  3C 50                             CMP     AL,SWIDTH       ;TEST FOR OUT OF RANGE COLUMN
0B26  73 03                             JNB     POS2            ; JUMP IF OUT
0B28  A2 07BA R                         MOV     CRTPARB,AL      ;SET NEW COLUMN
0B2B  C6 06 060F R 00           POS2:   MOV     DREQ,0          ;CLEAR DATA REQUEST
0B30  C6 06 07BD R 08                   MOV     CRTPARB+3,8   ;POSITION CURSOR AND VALID ATTRIBUTE      *2*
0B35  EB 47 90                          JMP     SESC
                                ;
                                ; SET / RESET REVERSE VIDEO AND BLINKING
                                ;
0B38                            REVERSE:
0B38  C6 06 060F R FF                   MOV     DREQ,-1         ;SET DATA REQUEST
0B3D  C7 06 060A R 0B44 R               MOV     RADDR,OFFSET RV0 ; AND REQUESTER'S ADDRESS
0B43  C3                                RET
0B44  3C 30                     RV0:    CMP     AL,'0'
0B46  75 08                             JNZ     RV1             ;JUMP IF NO RESET OF THIS FLAG
0B48  E8 0A89 R                         CALL    RINV_VIDEO      ;RESET INVERSE VIDEO
0B4B  E8 0A06 R                         CALL    RBLINK          ;RESET BLINKING
0B4E  EB 10                             JMP SHORT RVE
0B50                            RV1:
0B50  3C 32                             CMP     AL,'2'          ;BLINKING TO SET ?
0B52  75 05                             JNZ     RV2             ; JUMP IF NOT
0B54  E8 0ACE R                         CALL    SBLINK          ;SET BLINKING
0B57  EB 07                             JMP SHORT RVE
0B59                            RV2:
0B59  3C 34                             CMP     AL,'4'          ;WOULD BE SET FLAG
0B5B  75 03                             JNZ     RVE             ;NO ACTION IF NOT '4'
0B5D  E8 0A68 R                         CALL    SINV_VIDEO      ;SET INVERSE VIDEO
0B60  C6 06 060F R 00           RVE:    MOV     DREQ,0          ;CLEAR DATA REQUEST
0B65  E9 0CF1 R                         JMP     RETSTAT1        ;SET STATE 1 AGAIN
                                ;
                                ; Direct cursor positioning routine.
                                ;

0B68  E8 0935 R                 CUP:    CALL    GETONE          ;Get X position.
0B6B  FE C8                             DEC     AL
0B6D  8A F0                             MOV     DH,AL           ;Save in DH.
0B6F  E8 0935 R                         CALL    GETONE          ;Get Y position.
0B72  FE C8                             DEC     AL
0B74  8A E6                             MOV     AH,DH           ;AH=ROW, AL=COLUMN
0B76  E9 096C R                         JMP     CMOVEEX         ;POSITION CURSOR
```

```
                            ;
                            ; Erase all of screen.
                            ;

0B79  C6 06 07BD R 81       ED:     MOV     CRTPARB+3,81H   ;CLEAR SCREEN AND SET ATTRIBUTE
                            ;
                            ; SEND ESC SEQUENCE TO CRTPIM
                            ;
0B7E  BB 07BA R             SESC:   MOV     BX,OFFSET CRTPARB ;BX=POINTER TO CRT PARAMETER BLOCK
0B81  C7 06 0615 R 07C5 R           MOV     STATE,OFFSET ST1 ;SET STATE 1 AGAIN
0B87  E9 0E58 R                     JMP     CRTPIM          ;PERFORM FUNCTION


                            ; Erase all/part of a line.
                            ;

0B8A  C6 06 07BD R 03       EL:     MOV     CRTPARB+3,3     ;ERASE TO END OF LINE
0B8F  EB ED                         JMP     SHORT SESC      ;PERFORM FUNCTION
                            ;
                            ; Special video modes.
                            ;

0B91  81 E9 0618 R          SGR:    SUB     CX,OFFSET PARMS-1 ;CX NOW HOLDS NUMBER OF PARAMETERS
0B95  E8 0942 R             SGRX:   CALL    GETPARM         ;Get trinary command type.
0B98  3C 00                         CMP     AL,0            ;TEST FOR RESET ALL ATTRIBUTES
0B9A  75 09                         JNZ     SGR1
0B9C  E8 0A89 R                     CALL    RINV_VIDEO      ;RESET INVERSE VIDEO
0B9F  E8 0A32 R                     CALL    RHALF_INT       ;RESET HALF INTENSITY
0BA2  E8 0AD6 R                     CALL    RBLINK          ;RESET BLINKING
0BA5  3C 07                 SGR1:   CMP     AL,7            ;TEST FOR INVERSE VIDEO              *1*
0BA7  75 03                         JNZ     SGR2
0BA9  E8 0A68 R                     CALL    SINV_VIDEO      ;SET INVERSE VIDEO
0BAC  3C 05                 SGR2:   CMP     AL,5            ;TEST FOR BLINKING
0BAE  75 03                         JNZ     SGR3
0BB0  E8 0ACE R                     CALL    SBLINK          ;SET BLINKING
0BB3  3C 01                 SGR3:   CMP     AL,1            ;TEST FOR BOLD ON
0BB5  75 03                         JNZ     SGR4
0BB7  E8 0A32 R                     CALL    RHALF_INT       ;RESET HALF INTENSITY
0BBA  3C 08                 SGR4:   CMP     AL,8            ;TEST FOR CONCEALED
0BBC  75 03                         JNZ     SGR5
0BBE  E8 09E3 R                     CALL    SHALF_INT       ;SET HALF INTENSITY
0BC1  80 3E 060D R 43       SGR5:   CMP     BYTE PTR MONO_COLOR,'C' ;TEST FOR COLOR CRT
0BC6  74 03                         JZ      SGRC                    ;JUMP IF TRUE
0BC8  E2 CB                         LOOP    SGRX                    ;GET NEXT PARAMETER
0BCA  C3                            RET
                            ;
                            ; SYSTEM HAS COLOR CRT -- CHECK FOR COLOR SETTINGS
                            ;
0BCB  3C 1E                 SGRC:   CMP     AL,30                   ;PARAMETERS < 30 ILLEGAL
0BCD  72 68                         JB      SGEXIT                  ;FORGET < 30 ONES
0BCF  3C 2F                         CMP     AL,47                   ;PARAMETERS > 47 ILLEGAL
0BD1  77 64                         JA      SGEXIT                  ;FORGET > 47
0BD3  3C 26                         CMP     AL,38                   ;38 AND 39 ARE ILLEGAL TOO
0BD5  74 60                         JZ      SGEXIT
0BD7  3C 27                         CMP     AL,39
0BD9  74 5C                         JZ      SGEXIT
0BDB  3C 28                         CMP     AL,40                   ;TEST FOR BACKGROUND SETTING
0BDD  73 07                         JAE     SGRBG                   ;JUMP FOR 40 OR ABOVE
```

```
0BDF  80 0E 060E R 02           OR      BYTE PTR BG_FG,02H      ;FLAG FOREGROUND DEFINED
0BE4  EB 05                     JMP SHORT SGRC1
0BE6  80 0E 060E R 01   SGRBG:  OR      BYTE PTR BG_FG,01H      ;FLAG BACKGROUND DEFINED
0BEB  BB 0C20 R         SGRC1:  MOV     BX,OFFSET COLOR_TBL-30  ;BX = BASE FOR COLOR_TBL
0BEE  D7                        XLAT                            ;TRANSLATE PARAMETER INTO COLOR BITS
0BEF  F6 06 060C R 01           TEST    MON_ATT,01H             ;TEST FOR INVERSE ACTIVE
0BF4  75 0B                     JNZ     SGRINV                  ;JUMP IF TRUE
                                ;
                                ; CHECK FG FOR RED
                                ;
0BF6  A8 04                     TEST    AL,04H                  ;TEST FOR FG RED SET
0BF8  74 10                     JZ      SGRCDO                  ;JUMP IF NOT
0BFA  80 26 07BC R FE           AND     CRTPARB+2,0FEH          ;RESET HALF INT ATTRIBUTE BIT
0BFF  EB 09                     JMP SHORT SGRCDO
0C01                    SGRINV:
DCD1  A8 20                     TEST    AL,20H                  ;TEST FOR RED IN BG
0C03  75 05                     JNZ     SGRCDO                  ;JUMP IF NOT
0C05  80 26 07BC R FE           AND     CRTPARB+2,0FEH          ;RESET HALF INT ATTRIBUTE BIT
0C0A                    SGRCDO:
0C0A  F6 06 060E R 02           TEST    BG_FG,02H               ;TEST FOR FG SETTING
0C0F  75 0B                     JNZ     SGRSFG                  ;JUMP IF TRUE
0C11  80 26 07BC R 1F           AND     CRTPARB+2,01FH          ;RESET BG BITS FOR ORING
0C16  08 06 078C R              OR      CRTPARB+2,AL            ;SET NEW BG
0C1A  EB 09                     JMP SHORT SGRCEX
0C1C                    SGRSFG:
0C1C  80 26 078C R E3           AND     CRTPARB+2,0E3H          ;RESET FG BITS FOR ORING
0C21  08 06 07BC R              OR      CRTPARB+2,AL            ;SET NEW FG
0C25                    SGRCEX:
0C25  C6 06 060E R 00           MOV     BG_FG,0                 ;RESET FLAGS
0C2A  E8 0CF1 R                 CALL    RETSTAT1                ;ACTIVATE NEW ATTRIBUTE
0C2D  F6 06 060C R 04           TEST    MON_ATT,04H             ;TEST FOR HALF INT ACTIVE
0C32  74 03                     JZ      SGEXIT
0C34  E8 09F6 R                 CALL    REAC_HI                 ;REACTIVATE HALF INT
0C37  49                SGEXIT: DEC     CX
0C38  74 03                     JZ      SGRENDE                 ;EXIT IF LAST PARAMETER PROCESSED
0C3A  E9 0B95 R                 JMP     SGRX                    ;GET NEXT PARAMETER
0C3D                    SGRENDE:
0C3D  C3                        RET
                        ;
                        ; COLOR TABLE
                        ;
0C3E  00                COLOR_TBL       DB      00              ;BLACK   FOREGROUND
0C3F  04                                DB      04H             ;RED     FOREGROUND
0C40  08                                DB      08H             ;GREEN   FOREGROUND
0C41  0C                                DB      0CH             ;YELLOW  FOREGROUND
0C42  10                                DB      10H             ;BLUE    FOREGROUND
0C43  14                                DB      14H             ;MAGENTA FOREGROUND
0C44  18                                DB      18H             ;CYAN    FOREGROUND
0C45  1C                                DB      1CH             ;WHITE   FOREGROUND
0C46  FF                                DB      -1              ;DUMMY
0C47  FF                                DB      -1              ;DUMMY
0C48  E0                                DB      0E0H            ;BLACK   BACKGROUND
0C49  C0                                DB      0C0H            ;RED     BACKGROUND
0C4A  A0                                DB      0A0H            ;GREEN   BACKGROUND
0C4B  80                                DB      080H            ;YELLOW  BACKGROUND
0C4C  60                                DB      060H            ;BLUE    BACKGROUND
0C4D  40                                DB      040H            ;MAGENTA BACKGROUND
0C4E  20                                DB      020H            ;CYAN    BACKGROUND
0C4F  00                                DB      0               ;WHITE   BACKGROUND
                        ;
```

```
                                ; END COLOR TABLE
                                ;
                                ;
                                ; Save / restore cursor position.
                                ;
0C50  A1 07BA R                 PSCP:   MOV     AX,WORD PTR CRTPARB ;Set save cursor posit. mode.
0C53  A3 0610 R                         MOV     SAVCRTPARB,AX   ;SAVE CURRENT CURSOR POSITION
0C56  C3                                RET
0C57  A1 0610 R                 PRCP:   MOV     AX,SAVCRTPARB    ;Restore last cursor save.
0C5A  E9 096C R                         JMP     CMOVEEX         ;POSITION CURSOR                *12*
```

IO.SYS

```
                                   ;
                                   ;
                                   ; Passing of FUNCTION KEY information to keyboard routine.
                                   ;
                                   ;
0C5D                               DEFFK:
0C5D  C6 06 0612 R 00                      MOV     STRINGF,0       ;Clear string flag.
0C62  BB 061A R                            MOV     BX,OFFSET PARMS+1
0C65  1E                                   PUSH    DS
0C66  07                                   POP     ES              ;ES:BX now points to esc sequence.
0C67  B8 0619 R                            MOV     AX,OFFSET PARMS ;Begin of parameter buffer.
0C6A  2B C8                                SUB     CX,AX           ;CX = length in bytes.
0C6C  80 3E 0617 R 00                      CMP     BYTE PTR [PRMPNT],0
0C71  75 01                                JNZ     FK1             ;Out if no adjust necessary.
0C73  49                                   DEC     CX
0C74  E8 0000 E                    FK1:    CALL    def_fun         ;Pass parameters to keyboard driver
                                                                   ;AL= Status.
0C77  A8 FF                                TEST    AL,0FFH
0C79  74 0C                                JZ      FK2             ;Out if no error.
0C7B  80 3E 0609 R 29                      CMP     INTTYPE,29H     ;ENTERED BY INT 29H ?
0C80  74 05                                JZ      FK2
0C82  5B                                   POP     BX              ;CLEAN STACK
0C83  5B                                   POP     BX
0C84  E9 05C0 R                            JMP     ERROR_12        ;STRING WAS NOT CORRECT OR DID  NOT FIT IN
                                                                   ; FUNCTION KEY BUFFER
0C87  C3                           FK2:    RET
```

```
                              ;
                              ; ANSI CURSOR POSITION REPORT (DEVICE STATUS REPORT)
                              ;
0C88                          XDSR:
0C88  A1 07BA R                       MOV     AX,WORD PTR CRTPARB      ;AL = COLUMN, AH = ROW
0C8B  FE C0                           INC     AL
0C8D  FE C4                           INC     AH                       ;ADD 1 TO DEFINE HOME AS L1 P1
0C8F  BB 0CC4 R                       MOV     BX,OFFSET CPR_MESS+2     ;POINTER TO ROW DIGITS IN CPR_MESS
0C92  E8 0CAE R                       CALL    THEX_ASCII               ;TRANSLATE AH (ROW HEX) TO TWO ASCII
                                                                       ;BYTES AND PLACE IN CPR_MESS
0C95  43                              INC     BX
0C96  43                              INC     BX                       ;MOVE POINTER TO COLUMN DIGITS
0C97  8A E0                           MOV     AH,AL                    ;COLUMN TO AH
0C99  E8 0CAE R                       CALL    THEX_ASCII               ;TRANSLATE
0C9C  C7 06 0000 E 0009               MOV     WORD PTR remnum,9        ;SET LENGTH OF MESSAGE FOR KEYBOARD
0CA2  C7 06 0000 E 0CC2 R             MOV     WORD PTR funcoff,OFFSET CPR_MESS ;MESSAGE POINTER FOR KBD
0CA8  80 0E 0000 E 20                 OR      BYTE PTR flag_buf,20H    ;TELL KBD TO RETURN CPR_MESS
0CAD  C3                              RET

0CAE                          THEX_ASCII:
0CAE  C7 07 3030                      MOV     WORD PTR [BX],3030H      ;INITIALIZE DIGITS
0CB2  80 EC 0A                THEX1:  SUB     AH,10
0CB5  72 04                           JB      LT10                     ;JUMP IF LESS THAN TEN
0CB7  FE 07                           INC     BYTE PTR [BX]
0CB9  EB F7                           JMP SHORT THEX1

0CBB                          LT10:
0CBB  43                              INC     BX
0CBC  80 C4 0A                        ADD     AH,10
0CBF  08 27                           OR      BYTE PTR [BX],AH
0CC1  C3                              RET

                              ;
                              ; CURSOR POSITION REPORT MESSAGE (ANSI)
                              ;
0CC2                          CPR_MESS:
0CC2  1B                              DB      1BH     ;ESC
0CC3  5B                              DB      '['
0CC4  30                              DB      30H     ;ROW (TENS)
0CC5  30                              DB      30H     ;ROW (ONES)
0CC6  3B                              DB      ';'     ;DECIMAL SEPERATOR
0CC7  30                              DB      30H     ;COLUMN (TENS)
0CC8  30                              DB      30H     ;COLUMN (ONES)
0CC9  52                              DB      'R'
0CCA  0D                              DB      0DH     ;CARRIAGE RETURN

                              ;
                              ; MUSIC ROUTINE
                              ;
0CCB                          PLAY_MUSIC:
0CCB  C6 06 060F R FF                 MOV     DREQ,-1          ;INDICATE REQUEST FOR MORE DATA
0CD0  C7 06 060A R 0CDC R             MOV     RADDR,OFFSET PM_FREQ ;FREQUENCY IS NEXT
0CD6  C6 06 07BD R 09                 MOV     CRTPARB+3,9      ;CRTPIN ESCAPE CODE FOR MUSIC
0CDB  C3                              RET

0CDC                          PM_FREQ:
0CDC  A2 07BE R                       MOV     CRTPARB+4,AL     ;SET FREQUENCY
0CDF  C7 06 060A R 0CE6 R             MOV     RADDR,OFFSET PM_TLENGTH ;TONE LENGTH IS NEXT
0CE5  C3                              RET
```

IO.SYS

```
0CE6                          PM_TLENGTH:
0CE6  A2 078F R                      MOV     CRTPARB+5,AL    ;SET LENGTH OF TONE
0CE9  C6 06 060F R 00                MOV     DREQ,0          ;CLEAR DATA REQUEST
0CEE  E9 0B7E R                      JMP     SESC            ;SEND SEQUENCE TO CRTPIM

                              ;
                              ; SET STATE 1 AND SET ATTRIBUTE .... STARTS LOOKING FOR ESC AGAIN
                              ;
0CF1                          RETSTAT1:
0CF1  C7 06 0615 R 07C5 R            MOV     STATE,OFFSET ST1
0CF7  C6 06 078D R 80                MOV     CRTPARB+3,80H   ;NEW ATTRIBUTE                       *2*
0CFC  BB 078A R                      MOV     BX,OFFSET CRTPARB ;                                  *2*
0CFF  E9 0E58 R                      JMP     CRTPIM          ;                                    *2*
                              ;
```

```
                            ;
                            ; PRINTER INITIALIZATION
                            ;

0D02                        PRN_INIT:
0D02  80 3E 0019 R 53               CMP    PRINTER_IF_TYPE,'S' ;SEE IF SERIAL PRINTER IF   *5*
0D07  74 06                         JZ     PRNSER_INIT         ;JUMP IF SERIAL IF          *5*
0D09  E8 0E1D R                     CALL   PINIT               ; ELSE INIT PARALLEL        *5*
0D0C  E9 05C7 R                     JMP    EXIT                ;DONE
                            ;
0D0F                        PRNSER_INIT:
0D0F  E8 0E04 R                     CALL   SIOINIT             ;INIT SERIAL IF             *5*
0D12  E9 05C7 R                     JMP    EXIT
                            ;
                            ; Printer status routine.
                            ;

0D15                        PRN_STA:
0D15  80 3E 0019 R 53               CMP    PRINTER_IF_TYPE,'S' ;TEST FOR SERIAL OR PARALLEL I/F
0D1A  74 0B                         JZ     PRN_SERST       ;JUMP IF PARALLEL I/F
0D1C  E8 0E36 R                     CALL   P1STATUS        ;GET STATUS OF PARALLEL I/F
0D1F  75 03                         JNZ    PRN_RDY_ST      ;JUMP IF PRINTER READY
0D21  E9 058C R                     JMP    BUS_EXIT        ;RETURN BUSY STATUS
0D24                        PRN_RDY_ST:
0D24  E9 05C7 R                     JMP    EXIT            ;PRINTER READY

0D27                        PRN_SERST:
0D27  E8 0DA0 R                     CALL   SRLSTAT         ;GET STATUS OF SERIAL I/F
0D2A  75 F8                         JNZ    PRN_RDY_ST      ;READY IF NOT ZERO
0D2C  E9 058C R                     JMP    BUS_EXIT        ;RETURN BUSY STATUS


                            ;
                            ; Printer write routine.
                            ;
0D2F  0000                   PRINTER_OUT DW 0              ;ADDRESS POINTER FOR PRINTERS
                            ;

0D31                        PRN_WRT:
0D31  80 3E 0019 R 53               CMP    PRINTER_IF_TYPE,'S' ;TEST FOR SERIAL OR PARALLEL I/F
0D36  74 09                         JZ     PRN_WR1         ;JUMP IF SERIAL I/F

0D38  C7 06 0D2F R 0E2C R           MOV    PRINTER_OUT,OFFSET P1CHROUT ;SET PARALLEL OUT POINTER
0D3E  EB 07 90                      JMP    PRN_WRTL
0D41                        PRN_WR1:
0D41  C7 06 0D2F R 0D83 R           MOV    PRINTER_OUT,OFFSET SRLOUT ;SET SERIAL OUT POINTER

0D47                        PRN_WRTL:
0D47  8B F7                         MOV    SI,DI           ;DET DESTINATION TO SOURCE
0D49                        PRN_WRX:
0D49  26: AC                        LODS   BYTE PTR ES:[SI] ;GET BYTE FROM TRANSFER ADDRESS
0D4B  51                            PUSH   CX               ;SAVE BYTE COUNTER
0D4C  86 C1                         XCHG   AL,CL            ;PIM NEEDS CHARACTER IN CL
0D4E  FF 16 0D2F R                  CALL   [PRINTER_OUT]    ;CALL SERIAL OR PARALLEL I/F
0D52  59                            POP    CX               ;RESTORE BYTE COUNTER
0D53  E2 F4                         LOOP   PRN_WRX          ;LOOP TILL USER BUFFER THROUGH
0D55  E9 05C7 R                     JMP    EXIT             ;DONE
```

```
0D58     02 [                TIM_DAYS: DB    2 DUP (?)       ;Number of days since 1-1-80.
                ??
                   ]

0D5A  ??                     TIM_MINS: DB    ?               ;Minutes.
0D5B  ??                     TIM_HRS:  DB    ?               ;Hours.
0D5C  ??                     TIM_HSEC: DB    ?               ;Hundreths of a second.
0D5D  ??                     TIM_SECS: DB    ?               ;Seconds.

                             ;
                             ; Time write routine.
                             ;

0D5E                         TIM_WRT:
0D5E  BE 0D58 R                      MOV     SI,OFFSET TIM_DAYS
0D61  87 F7                          XCHG    SI,DI
0D63  06                             PUSH    ES
0D64  8C D8                          MOV     AX,DS
0D66  1F                             POP     DS
0D67  8E C0                          MOV     ES,AX
0D69  B9 0006                        MOV     CX,6
0D6C  F3/ A4                         REP     MOVSB
0D6E  80 00                          MOV     AL,0
0D70  E9 05C7 R                      JMP     EXIT

                             ;
                             ; Time read routine.
                             ;

0D73                         TIM_RED:
0D73  BE 0D58 R                      MOV     SI,OFFSET TIM_DAYS
0D76  B9 0006                        MOV     CX,6
0D79  F3/ A4                         REP     MOVSB
0D7B  B0 00                          MOV     AL,0
0D70  E9 05C7 R                      JMP     EXIT
```

```
                           C  ;**********************************************************************
                           C  ;*                                                                    *
                           C  ;*                    EQUATES used by the SER PIM                     *
                           C  ;*                                                                    *
                           C  ;**********************************************************************
                           C  ;
                           C
                           C  ;
                           C  ;      PORT ADDRESSES FOR SERIAL IF RS232 (2651)
                           C  ;
 = 0060                    C         SPRDATA EQU     60H     ;READ DATA
 = 0061                    C         SPRSTAT EQU     61H     ;READ STATUS
 = 0063                    C         SPRCOM  EQU     63H     ;READ COMMAND
 = 0064                    C         SPWDATA EQU     64H     ;WRITE DATA
 = 0066                    C         SPWMODE EQU     66H     ;WRITE MODE
 = 0067                    C         SPWCOM  EQU     67H     ;WRITE COMMAND
                           C  ;
                           C  ;      XON-XOFF VALUES
                           C  ;
 = 0011                    C         XON     EQU     11H
 = 0013                    C         XOFF    EQU     13H
                           C  ;
                           C  ;      STATUS EQUATES FOR SERIAL IF RS232 (2651)...BIT MAPPED
                           C  ;
 = 0001                    C         TXRDY   EQU     01H     ;TRANSMIT HOLDING REGISTER EMPTY
 = 0002                    C         RXRDY   EQU     02H     ;RECEIVE HOLDING REGISTER EMPTY
 = 0004                    C         TXEMT   EQU     04H     ;CHANGE IN DSR OR DCD OR TRANSMIT
 = 0008                    C         PARITY  EQU     08H     ;PARITY ERROR
 = 0010                    C         OVERRUN EQU     10H     ;OVERRUN ERROR
 = 0020                    C         FRAMING EQU     20H     ;FRAMING ERROR
 = 0040                    C         DCD     EQU     40H     ;DATA CARRIER DETECT
 = 0080                    C         DSR     EQU     80H     ;DATA SET READY
                           C  ;
                           C  ;**********************************************************************
                           C  ;*           VARIABLES TO BE PROVIDED BY THE USER                     *
                           C  ;*                                                                    *

                           C  ;**********************************************************************
                           C  ;
                           C  ;      M1RS232 BYTE    BIT MAPPED : NUMBER OF STOP BITS
                           C  ;                                   PARITY EVEN OR ODD
                           C  ;                                   PARITY ENABLE OR DISABLE
                           C  ;                                   BITS PER CHARACTER
                           C  ;                                   ASYNC OR SYNC COMMUNICATION
                           C  ;
                           C  ;      M2RS232 BYTE    BIT MAPPED : INTERNAL OR EXTERNAL CLOCKS
                           C  ;                                   BAUD RATE
                           C  ;
                           C  ;      PVRS232 BYTE    00H          PROTOKOL VECTOR (FOR FUTURE EXPANSION)
                           C  ;                                   CURRENTLY 00H
                           C  ;
                           C  ;                                   THE SERIAL INTERFACE
                           C  ;
                           C  ;-----------------------------------------------------------------------
                           C  ;
                           C  ;**********************************************************************
                           C  ;*                                                                    *
                           C  ;*                      INTERNAL VARIABLES                            *
                           C  ;*                                                                    *
                           C  ;**********************************************************************
                           C  ;
 0080  00                  C  SACTIVE       DB      0       ;SERIAL I/F ACTIVE FLAG
 0081  00                  C  PACTIVE       DB      0       ;PARALLEL I/F ACTIVE FLAG
 0082  00                  C  XOFFFLG       DB      0       ;XOFF FLAG
                           C  ;
                           C  ;-----------------------------------------------------------------------
```

```
                          C  ;*************************************************************************
                          C  ;*                                                                      *
                          C  ;*          SERIAL INTERFACE PERIPHERAL INTERFACE MODULE                *
                          C  ;*                                                                      *
                          C  ;*************************************************************************
                          C  ;
                          C  ;
                          C  ; SERIAL OUTPUT ENTRY POINT
                          C  ;
0D83  BB 0D90 R           C  SRLOUT:        MOV   BX,OFFSET SO_DISP_TBL
0D86  A0 001C R           C  SIF_DISP:      MOV   AL,PVRS232      ;GET PROTOCOL VECTOR
0D89  D0 E0               C                 SHL   AL,1            ;AL*2...TABLE TYPE WORD
0D8B  98                  C                 CBW                   ;EXPAND BYTE IN AL TO WORD IN AX
0D8C  03 D8               C                 ADD   BX,AX           ;BX = POINTER TO ROUTINE ADDRESS
0D8E  FF 27               C                 JMP   [BX]            ;JUMP TO ROUTINE FOR DEFINED PROTOCOL
                          C
0D90  0DFA R              C  SO_DISP_TBL:   DW    SPAOUT
0D92  0DFA R              C                 DW    SPAOUT
0D94  0DFA R              C                 DW    SPAOUT
0D96  0DFA R              C                 DW    SPAOUT
                          C
0D98  0DCA R              C  SST_DISP_TBL:  DW    SPAOST
0D9A  0DCA R              C                 DW    SPAOST
0D9C  0DCA R              C                 DW    SPAOST
0D9E  0DCA R              C                 DW    SPAOST
                          C  ;
                          C  ; SERIAL OUTPUT STATUS
                          C  ;
0DA0  BB 0D98 R           C  SRLSTAT:       MOV   BX,OFFSET SST_DISP_TBL
0DA3  EB E1               C                 JMP   SIF_DISP        ;JUMP TO ROUTINE ACCORDING TO PROTOCOL
                          C  ;
                          C  ; GET INPUT STATUS
                          C  ;
0DA5  F6 06 0D80 R FF     C  SPAIST:        TEST  SACTIVE,-1      ;TEST FOR SERIAL I/F ACTIVE
0DAA  75 03               C                 JNZ   SPAI1           ; JUMP IF TRUE
0DAC  E8 0E04 R           C                 CALL  SIOINIT         ;INITIALIZE SERIAL I/F IF REQUIRED
0DAF  E4 61               C  SPAI1:         IN    AL,SPRSTAT
0DB1  24 38               C                 AND   AL,OVERRUN OR PARITY OR FRAMING
0DB3  74 03               C                 JZ    SPAI2           ;JUMP IF NONE OF CHECKED ERRORS OCCURED
0DB5  E8 0DC1 R           C                 CALL  TRERR           ;CALL ERROR ROUTINE, ERROR ENCOUNTERED
                          C                                       ; IN RECEIVER
0DB8  E4 61               C  SPAI2:         IN    AL,SPRSTAT
0DBA  24 02               C                 AND   AL,RXRDY        ;TEST FOR CHARACTER RECEIVED
0DBC  74 02               C                 JZ    SPAI3           ; JUMP IF NOT
0DBE  0C FF               C                 OR    AL,-1           ;FLAG CHARACTER RECEIVED
0DC0  C3                  C  SPAI3:         RET
                          C
0DC1  E4 60               C  TRERR:         IN    AL,SPRDATA      ;DUMMY READ
0DC3  E4 63               C                 IN    AL,SPRCOM       ;READ COMMAND BYTE
0DC5  0C 10               C                 OR    AL,10H          ;RESET ERROR
0DC7  E6 67               C                 OUT   SPWCOM,AL
0DC9  C3                  C                 RET
                          C  ;
```

```
                         C  ; GET PRINTER STATUS
                         C  ;
0DCA  F6 06 0D80 R FF    C  SPAOST:     TEST   SACTIVE,-1     ;TEST FOR SERIAL I/F ACTIVE
0DCF  75 03              C              JNZ    SPA1           ; SKIP INITIALIZATION IF TRUE
0DD1  E8 0E04 R          C              CALL   SIOINIT        ; INITIALIZE THE SERIAL I/F
0DD4  E8 0DA5 R          C  SPA1:       CALL   SPAIST         ;CHECK INPUT STATUS
0DD7  74 06              C              JZ     SPA2           ;JUMP IF NO INPUT
0DD9  E8 0DF2 R          C              CALL   SPAIN          ;GET INPUT CHARACTER
0DDC  A2 0D82 R          C              MOV    XOFFFLG,AL
0DDF  80 3E 0D82 R 13    C  SPA2:       CMP    XOFFFLG,XOFF   ;TEST FOR PRINTER NOT READY
0DE4  74 09              C              JZ     SPA3           ;JUMP IF XOFF .. PRINTER NOT READY
0DE6  E4 61              C              IN     AL,SPRSTAT
0DE8  24 01              C              AND    AL,TXRDY       ;TEST FOR TRANSMITTER READY
0DEA  74 02              C              JZ     SPA4           ; JUMP IF NOT
0DEC  0C FF              C              OR     AL,-1          ;FLAG TRANSMITTER READY
0DEE  C3                 C  SPA4:       RET
0DEF  32 C0              C  SPA3:       XOR    AL,AL          ;FLAG PRINTER NOT READY
0DF1  C3                 C              RET
                         C  ;
                         C  ; GET CHARACTER FROM INTERFACE
                         C  ;
0DF2  E8 0DA5 R          C  SPAIN:      CALL   SPAIST         ;CHECK INPUT STATUS
0DF5  74 FB              C              JZ     SPAIN          ;WAIT IF ZERO
0DF7  E4 60              C              IN     AL,SPRDATA     ;GET CHARACTER
0DF9  C3                 C              RET
                         C  ;
                         C  ; OUTPUT CHARACTER
                         C  ;
0DFA  E8 0DCA R          C  SPAOUT:     CALL   SPAOST         ;CHECK OUTPUT STATUS
0DFD  74 FB              C              JZ     SPAOUT         ;WAIT IF ZERO
0DFF  86 C1              C              XCHG   AL,CL          ;CHARACTER TO AL
0E01  E6 64              C              OUT    SPWDATA,AL     ;OUTPUT THE CHARACTER
0E03  C3                 C              RET
                         C  ;
                         C  ; INITIALIZE THE SERIAL I/O
                         C  ;
0E04  A0 001A R          C  SIOINIT:    MOV    AL,M1RS232     ;GET FRAMING AND MODE
0E07  E6 66              C              OUT    SPWMODE,AL     ;OUT MODE 1 BYTE
0E09  A0 001B R          C              MOV    AL,M2RS232     ;CLOCK AND SPEED
0E0C  E6 66              C              OUT    SPWMODE,AL     ;OUT MODE 2 BYTE
0E0E  B0 37              C              MOV    AL,37H         ;ENABLE TRANSMITTER AND RECEIVER
0E10  E6 67              C              OUT    SPWCOM,AL      ; SET DTR AND RTS, RESET ERROR
0E12  C6 06 0D80 R FF    C              MOV    SACTIVE,-1     ;FLAG  SERIAL INTERFACE AS ENABLED
0E17  C6 06 0D81 R 00    C              MOV    PACTIVE,0      ;FLAG PARALLEL INTERFACE DISABLED
0E1C  C3                 C              RET
                         C  ;
                         C  ;
                         C
```

```
IO.SYS
                           C  ;***********************************************************************
                           C  ;***********************************************************************
                           C  ;
                           C  ;               PARALLEL INTERFACE (CENTRONICS)
                           C  ;
                           C  ;***********************************************************************
                           C  ;***********************************************************************
                           C  ;
                           C  ;
                           C  ;
                           C  ;***********************************************************************
                           C  ;*                                                                    *
                           C  ;*                    EQUATES used by the PAR PIM                     *
                           C  ;*                                                                    *
                           C  ;***********************************************************************
                           C  ;
                           C
                           C  ;
                           C  ;       PORT ADDRESSES FOR PARALLEL I/F (CENTRONICS)
                           C  ;
= 0060                     C          PBDA    EQU     60H     ;DATA PORT
= 0061                     C          PBSTA   EQU     61H     ;STATUS PORT
= 0063                     C          PBCOM   EQU     63H     ;CONTROL PORT
                           C  ;
                           C  ;       STATUS EQUATES FOR PARALLEL I/F (CENTRONICS)
                           C  ;
= 0020                     C          BUSY    EQU     20H     ;PRINTER BUSY
= 0002                     C          POBF    EQU     02H     ;OUTPUT BUFFER FULL
                           C  ;***********************************************************************
                           C  ;***********************************************************************
                           C  ;
                           C  ;               PARALLEL INTERFACE (CENTRONICS)
                           C  ;
                           C  ;***********************************************************************
                           C  ;***********************************************************************
                           C  ;
                           C  ; INITIALIZE PARALLEL INTERFACE
                           C  ;
0E1D  B0 AA                C  PINIT:          MOV     AL,0AAH
0E1F  E6 63                C                  OUT     PBCOM,AL        ;INITIALIZE INTERFACE
0E21  C6 06 0D80 R 00      C                  MOV     SACTIVE,0       ;DISABLE SERIAL INTERFACE
0E26  C6 06 0D81 R FF      C                  MOV     PACTIVE,-1      ;FLAG PARALLEL I/F AS ACTIVE
0E2B  C3                   C                  RET
                           C
                           C  ;
                           C  ; OUTPUT CHARACTER IN CL
                           C  ;
0E2C  E8 0E36 R            C  P1CHROUT:       CALL    P1STATUS        ;CHECK INTERFACE STATUS
0E2F  74 FB                C                  JZ      P1CHROUT        ; WAIT
0E31  86 C1                C                  XCHG    AL,CL           ;CHARACTER TO AL
0E33  E6 60                C                  OUT     PBDA,AL         ;OUTPUT THE CHARACTER IN AL
0E35  C3                   C                  RET
                           C  ;
                           C  ; GET PRINTER STATUS
                           C  ;
0E36  F6 06 0D81 R FF      C  P1STATUS:       TEST    PACTIVE,-1      ;TEST FOR PARALLEL I/F ACTIVE
0E3B  75 03                C                  JNZ     P1STA1          ;JUMP IF ACTIVE
0E3D  E8 0E1D R            C                  CALL    PINIT           ;INITIALIZE PARALLEL I/F
0E40  E4 61                C  P1STA1:         IN      AL,PBSTA        ;GET PRINTER STATUS
0E42  24 22                C                  AND     AL,BUSY OR POBF
0E44  74 03                C                  JZ      P1STATX         ;JUMP IF PRINTER ACCEPTS A BYTE
0E46  32 C0                C                  XOR     AL,AL           ;ZERO INDICATES PRINTER NOT READY
0E48  C3                   C                  RET
                           C
0E49  0C FF                C  P1STATX:        OR      AL,-1           ;NOT ZERO INDICATES PRINTER READY
0E4B  C3                   C                  RET
```

```
                         C ;*********************************************************************
                         C ;*                                                                   *
                         C ;*                  EQUATES used by the CRT PIM                      *
                         C ;*                                                                   *
                         C ;*********************************************************************
                         C ;
                         C ; EQUATES to the CRT Paramater Block (CRTPB)
                         C ;
= 0000                   C CPB_COL EQU     0       ; column
= 0001                   C CPB_ROW EQU     1       ; row
= 0002                   C CPB_ATTR EQU    2       ; attribute
= 0003                   C CPB_ESC EQU     3       ; PIM escape code
= 0004                   C CPB_FREQ EQU    4       ; Music frequency    *change 05*
= 0004                   C CPB_RES1 EQU    4       ; reserved
= 0005                   C CPB_FLEN EQU    5       ; Length of Music frequency  *change 05*
= 0005                   C CPB_RES2 EQU    5       ; reserved
                         C ;
                         C ;  General EQUATES
                         C ;
= 0018                   C ROWS    EQU     24      ; Rows on the screen
= 0050                   C SCWID   EQU     80      ; Screen width
= 0040                   C CL_MASK EQU     040H    ; "Send Character" Mask
= 0080                   C ATTR_MASK EQU   80H     ; Set Attribute Bit of Escape Byte
= 000F                   C ESC_MASK  EQU   0FH     ; Mask to isolate Escape Code of Escape Byte
                         C ;
                         C ;               MACRO LIBRARY FOR NCR DM-5 GDC 15/11/82 10:00 WF
                         C ;               ------------------------------------------------
                         C ;
                         C ;               'GDC-EQUATES' 27/10/82 10:00 WF
                         C ;                ============================
                         C ;
                         C ;       READ
= 00A0                   C GDCSTA  EQU     0A0H            ;STATUS PORT
= 00A1                   C FIFO    EQU     0A1H            ;GDC FIFO PORT ADDR
                         C ;
                         C ;
```

IO.SYS

```
                         C ;      WRITE
 = 00A0                  C GDCPAR  EQU    0A0H          ;PARAMETER INTO FIFO
 = 00A1                  C GDCCOM  EQU    0A1H          ;COMMAND INTO FIFO
                         C ;
                         C ;
                         C ;      ORGANISATION OF GRAPHIC RAM
                         C ;
                         C ;      576 X 400  PIXELS
                         C ;
 = 1FFF                  C GRAEND  EQU    1FFFH         ;END ADDRESS OF GRAPHIC RAM
 = 0048                  C NRWAPL  EQU    72            ;NUMBER OF WORD ADDR PER LINE
 = 0024                  C WPL     EQU    NRWAPL/2      ;WORDS / LINE
 = 000A                  C LPC     EQU    10            ;LINES / CHARACTER
                         C ;
                         C ;      MEANING OF GDC STATUS BITS
                         C ;
 = 0001                  C DATRDY  EQU    01H           ;A BYTE IS AVAILABLE TO READ
 = 0002                  C FIFULL  EQU    02H           ;FIFO IS FULL
 = 0004                  C FIFEMP  EQU    04H           ;FIFO IS EMPTY
 = 0008                  C DRWINP  EQU    08H           ;DRAWING IN PROCESS
 = 0010                  C DMAEXC  EQU    10H           ;DMA DATA TRANSFER IN PROCESS
 = 0020                  C VERETR  EQU    20H           ;VERTICAL RETRACE IN PROCESS
 = 0040                  C HORETR  EQU    40H           ;HORIZONTAL RETRACE IN PROCESS
 = 0080                  C LIPDET  EQU    80H           ;LIGHT PEN DETECT (ADDRESS VALID)
                         C ;
                         C ;
                         C ;      COMMANDS
                         C ;
 = 0000                  C GDCRES  EQU    0             ;RESET - BLANK DISPLAY, IDLE MODE, INITIALIZE
 = 006E                  C VSYNCS  EQU    06EH          ;SLAVE MODE
 = 006F                  C VSYNCM  EQU    06FH          ;MASTER MODE
 = 004B                  C CCHAR   EQU    04BH          ;CURSOR & CHARACTER CHARACTERISTICS
 = 006B                  C START   EQU    06BH          ;START DISPLAY & END IDLE MODE
 = 0046                  C ZOOM    EQU    046H          ;SPECIFY ZOOM FACTOR
 = 0049                  C CURS    EQU    049H          ;SPECIFY CURSOR POSITION
 = 0047                  C PITCH   EQU    047H          ;PITCH SPECIFICATION
 = 004A                  C MASKREG EQU    04AH          ;LOAD MASK REGISTER
 = 004C                  C FIGS    EQU    04CH          ;SPECIFY FIGURE DRAWING PARAMETER
 = 006C                  C FIGD    EQU    06CH          ;START FIGURE DRAW
 = 0068                  C GCHRD   EQU    068H          ;START GRAPHICS CHARACTER DRAW
 = 00E0                  C CURD    EQU    0E0H          ;READ CURSOR ADDRESS
 = 00C0                  C LPRD    EQU    0C0H          ;READ LIGHT PEN ADDRESS
                         C ;
 = 0070                  C PRAM    EQU    070H          ;LOAD PARAMETER RAM
 = 0000                  C PRAMSA  EQU    0             ;LOWER 4 BITS ARE STARTING ADDRESS IN RAM
                         C                              ;( COMMAND + SA )
                         C ;
 = 0020                  C WDAT    EQU    020H          ;WRITE DATA INTO DISPLAY MEMORY
                         C                              ;( COMMAND + TYPE + MODE )
                         C                              ;DATA TRANSFER TYPES
 = 0000                  C TYWORD  EQU    0             ;WORD, LOW THEN HIGH BYTE
 = 0010                  C TYLOBY  EQU    010H          ;LOW BYTE OF THE WORD
 = 0018                  C TYHIBY  EQU    018H          ;HIGH BYTE OF THE WORD
                         C                              ;MODE OF RMW MEMORY CYCLE
 = 0000                  C MOREPL  EQU    0             ;REPLACE WITH PATTERN
 = 0001                  C MOCOMP  EQU    01H           ;COMPLEMENT
 = 0002                  C MORES   EQU    02H           ;RESET TO 0
 = 0003                  C MOSET   EQU    03H           ;SET TO 1
                         C ;
 = 00A0                  C RDAT    EQU    0A0H          ;READ DATA FROM DISPLAY MEMORY
```

```
                        C                               ;( COMMAND + TYPE )
                        C                               ;TYPES AS AT WDAT
                        C  ;
 = 00A4                 C  DMAR    EQU    0A4H          ;DMA READ REQUEST
                        C                               ;( COMMAND + TYPE )
                        C                               ;TYPES AS AT WDAT
                        C  ;
 = 0024                 C  DMAW    EQU    024H          ;DMA WRITE REQUEST
                        C                               ;( COMMAND + TYPE + MODE )
                        C                               ;TYPES AND MODES AS AT WDAT
                        C  ;
                        C  ;       PARAMETERS
                        C -;
                        C  ;       RESET
                        C  ;
 = 0000                 C  RESMOP  EQU    0             ;MODE OF OPERATION SELECT BITS
                        C                               ;( RESMOP + DISPLAY + FRAME + DYNRAM + WINDOW )
                        C                               ;DISPLAY MODE
 = 0000                 C  MIXGAC  EQU    0H            ;MIXED GRAPHICS & CHARACTER
 = 0002                 C  GRAMOD  EQU    02H           ;GRAPHICS MODE
 = 0020                 C  CHAMOD  EQU    020H          ;CHARACTER MODE
                        C                               ;VIDEO FRAMING
 = 0000                 C  NOINTL  EQU    0             ;NON-INTERLACED
 = 0008                 C  INLRPF  EQU    08H           ;INTERLACED REPEAT FIELD FOR CHARACTER DISPLAYS
 = 0009                 C  INTLAC  EQU    09H           ;INTERLACED
                        C                               ?DYNAMIC RAM REFRESH CYCLES ENABLE
 = 0000                 C  SATRM   EQU    0             ;NO REFRESH - STATIC RAM
 = 0004                 C  DYNRAM  EQU    04H           ;REFRESH - DYNAMIC RAM
                        C                               ;DRAWING TIME WINDOW
 = 0000                 C  DRWALL  EQU    0             ;DRAWING DURING ACTIVE DISPLAY TIME AND RETRACE
 = 0010                 C  DRWRET  EQU    010H          ;DRAWING ONLY DURING RETRACE BLANKING
                        C  ;
                        C  ;
                        C  ;
                        C  ;*** CRT PERIPHERAL INTERFACE MODULE    DATA AREA
                        C  ;
                        C  ;
                        C  ;  CURSOR POSITION VARIABLES
                        C  ;
                        C  ; **** WARNING  ** for performance reasons these bytes are sometimes loaded
                        C  ;                  in pairs!!
 0E4C  00               C  CURCOL  DB     0
 0E4D  00               C  CURROW  DB     0
 0E4E  00               C  ATTRIBUTE DB   0
 0E4F  00               C  OUTCHAR DB     0
                        C  ;
                        C  ;
                        C  ;  DEFINITION OF CRT PAGE VARIABLES
                        C  ;
 0E50  0000             C  SP1     DW     0       ; START OF PAGE 1
 0E52  00               C  LP11    DB     0       ; LENGTH OF PAGE1 LOW
 0E53  00               C  LP12    DB     0       ; LENGTH OF PAGE1 HIGH
 0E54  0000             C  SP2     DW     0       ; START OF PAGE 2
 0E56  00               C  LP21    DB     0       ; LENGTH OF PAGE2 LOW
 0E57  00               C  LP22    DB     0       ; LENGTH OF PAGE2 HIGH
                        C  ;
                        C  ;
                        C
```

```
                       C  ;***********************************************************************
                       C  ;*                                                                     *
                       C  ;*              CRT Peripheral Interface Module                        *
                       C  ;*                                                                     *
                       C  ;***********************************************************************
                       C  ;
                       C  ;  This Module is a hardware dependent, Operating System independent driver
                       C  ;  for CRT display output
                       C  ;
                       C  ;  Entry Parameters:
                       C  ;       CL = Character to be OUTPUT
                       C  ;       BX = Address of CRT Parameter Block
                       C  ;
                       C  ;  Exit:  All registers unchanged
                       C  ;
0E58                   C  CRTPIN:
0E58  50               C         PUSH    AX
0E59  53               C         PUSH    BX
0E5A  51               C         PUSH    CX
0E5B  52               C         PUSH    DX       ; SAVE ALL OF THE REGISTERS WE WILL BE WORKING WITH
0E5C  56               C         PUSH    SI
0E5D  88 0E 0E4F R     C         MOV     OUTCHAR,CL      ; SAVE OUT CHARACTER IN MEMORY FOR LATER REF
0E61  8B 07            C         MOV     AX,CPB_COL[BX]
0E63  A3 0E4C R        C         MOV     WORD PTR CURCOL,AX      ; ALSO SAVE ROW/COLUMN IN MEMORY
0E66  8A 47 03         C         MOV     AL,CPB_ESC[BX]
0E69  A8 FF            C         TEST    AL,0FFH         ;
0E6B  74 23            C         JZ      DO_OUTCHAR              ; IF ESCAPE = 0 THEN JUST OUTPUT CHARACTER
0E6D  A8 80            C         TEST    AL,ATTR_MASK    ;
0E6F  74 07            C         JZ      DO_ESC          ; JUMP IF NO SET ATTRIBUTE SPECIFIED
0E71  8A 67 02         C         MOV     AH,CPB_ATTR[BX] ;
0E74  88 26 0E4E R     C         MOV     ATTRIBUTE,AH    ; SET ATTRIBUTE BYTE
```

```
0E78                    C  DO_ESC:
0E78  24 0F             C          AND     AL,ESC_MASK
0E7A  74 0E             C          JZ      TEST_VID_OUT    ; SKIP ESCAPE PROCESSING IF NO ESCAPE FUNCTION
0E7C  D0 E0             C          SHL     AL,1            ; FOR TABLE REFERENCING
0E7E  98                C          CBW                     ; EXPAND AL INTO AH
0E7F  BE 0EEA R         C          MOV     SI,OFFSET ESC_TABLE
0E82  03 F0             C          ADD     SI,AX           ; AX = ADDRESS OF ESCAPE ROUTINE ADDRESS
0E84  53                C          PUSH    BX              ; SAVE CRT PARAMETER BLOCK ADDRESS
0E85  51                C          PUSH    CX              ; SAVE CHARACTER TO OUTPUT
0E86  FF 14             C          CALL    WORD PTR [SI]   ; PERFORM ESCAPE FUNCTION
0E88  59                C          POP     CX              ; RESTORE CHARACTER AND CRTPB ADDRESS
0E89  5B                C          POP     BX
0E8A                    C  TEST_VID_OUT:
0E8A  F6 47 03 40       C          TEST    BYTE PTR CPB_ESC[BX],CL_MASK
0E8E  74 1F             C          JZ      CRT_EXIT
0E90                    C  DO_OUTCHAR:
0E90  80 3E 0E4C R 50   C          CMP     CURCOL,SCWID    ; COLUMN > 80?
0E95  75 03             C          JNZ     01              ; JUMP IF NO
0E97  E8 12CD R         C          CALL    SCLUP4          ; ELSE SCROLL UP SCREEN
0E9A                    C  01:
0E9A  8B 16 0E4E R      C          MOV     DX,WORD PTR ATTRIBUTE   ; DH=OUTCAR DL=ATTRIBUTE
0E9E  E8 0F0B R         C          CALL    WRGCHR
0EA1  FE 06 0E4C R      C          INC     CURCOL
0EA5  80 3E 0E4C R 50   C          CMP     CURCOL,SCWID
0EAA  72 03             C          JB      CRT_EXIT
0EAC  E8 115C R         C          CALL    BMPCR1          ; IF CURCOL>80, BUMP CUR
0EAF                    C  CRT_EXIT:
0EAF  5E                C          POP     SI
0EB0  5A                C          POP     DX
0EB1  59                C          POP     CX
0EB2  5B                C          POP     BX
0EB3  A1 0E4C R         C          MOV     AX,WORD PTR CURCOL
0EB6  89 07             C          MOV     CPB_COL[BX],AX  ; Restore CRTPB COL/ROW to latest state
0EB8  58                C          POP     AX
0EB9  C3                C          RET
                        C  ;
                        C  ;***  High Performance Screen Write     HIP_OUT
                        C  ;
                        C  ;        Entry Conditions - BX = CRTPB Address
                        C  ;                           CL = Character to OUTPUT
                        C  ;        Exit Conditions  - BX - Preserved
                        C  ;                           AX, CX, DX - Destroyed
                        C  ;                           CPB_COL and CPB_ROW fields of CRTPB updated
                        C  ;
0EBA                    C  HIP_OUT:                                ; *Additn of HIP_OUT Routine-Change *04
0EBA  8B 07             C          MOV     AX,CPB_COL[BX]          ; *Performance Optimized *1.09*
0EBC  A3 0E4C R         C          MOV     WORD PTR CURCOL,AX      ; Set-up CURCOL, CURROW, OUTCHAR fields
0EBF  88 0E 0E4F R      C          MOV     OUTCHAR,CL
0EC3  53                C          PUSH    BX
0EC4  80 3E 0E4C R 50   C          CMP     CURCOL,SCWID            ; COLUMN > 80?
0EC9  75 03             C          JNZ     H1              ; JUMP IF NO
0ECB  E8 12CD R         C          CALL    SCLUP4          ; ELSE SCROLL UP SCREEN
0ECE                    C  H1:
0ECE  8B 16 0E4E R      C          MOV     DX,WORD PTR ATTRIBUTE   ; DH=OUTCAR DL=ATTRIBUTE
0ED2  E8 0F0B R         C          CALL    WRGCHR
0ED5  FE 06 0E4C R      C          INC     CURCOL
0ED9  80 3E 0E4C R 50   C          CMP     CURCOL,SCWID
0EDE  72 03             C          JB      H2
0EE0  E8 115C R         C          CALL    BMPCR1          ; IF CURCOL>80, BUMP CUR
0EE3                    C  H2:
```

IO.SYS

```
 0EE3  5B                 C          POP    BX
 0EE4  A1 0E4C R          C          MOV    AX,WORD PTR CURCOL      ; Update CRTPB with CURCOL and CURROW
 0EE7  89 07              C          MOV    CPB_COL[BX],AX
 0EE9  C3                 C          RET
                          C  ;
                          C  ;***  Escape Table - Routines will be called indirect using the escape code * 2
                          C  ;                    as an offset to the routine address
                          C  ;
 0EEA                     C  ESC_TABLE:
 0EEA  0F0A R             C          DW     OFFSET(NO_OP)
 0EEC  126D R             C          DW     OFFSET(VCLEAR)
 0EEE  122A R             C          DW     OFFSET(CLEOS)
 0EF0  1209 R             C          DW     OFFSET(ICLEOL)
 0EF2  12FB R             C          DW     OFFSET(SCROLLDN)
 0EF4  129F R             C          DW     OFFSET(SCROLLUP)
 0EF6  1199 R             C          DW     OFFSET(INSCHR)
 0EF8  1105 R             C          DW     OFFSET(DELCHR)
 0EFA  1171 R             C          DW     OFFSET(WRITEPOS)
 0EFC  1187 R             C          DW     OFFSET(MUSIC)
 0EFE  0F0A R             C          DW     OFFSET(NO_OP)
 0F00  125A R             C          DW     OFFSET(ILF)
 0F02  0F0A R             C          DW     OFFSET(NO_OP)
 0F04  0F0A R             C          DW     OFFSET(NO_OP)
 0F06  0F0A R             C          DW     OFFSET(NO_OP)
 0F08  0F0A R             C          DW     OFFSET(NO_OP)
                          C  ;
                          C  ;*** NO_OP   SIMPLY RETURNS IF ESCAPE CODE NOT IMPLEMENTED
                          C  ;
 0F0A                     C  NO_OP:
 0F0A  C3                 C          RET
                          C  ;
                          C  ;
                          C  ;    WRGCHR, RDGCHR   WRITE AND READ GRAPHICS CHARACTER ROUTINES
                          C  ;
                          C  ;            WRITE OR READ ONE CHARACTER TO/FROM GDC IN MIXED MODE
                          C  ;
                          C  ;
                          C  ;***   WRGCHR  - Write Graphics Character
                          C  ;                   ENTRY - DL = ATTRIBUTE
                          C  ;                           DH = CHARACTER
                          C  ;
 0F0B                     C  WRGCHR:
 0F0B  E4 A0              C  XX1:    IN     AL,GDCSTA
 0F0D  24 02              C          AND    AL,FIFULL
 0F0F  75 FA              C          JNZ    XX1             ;LOOP UNTIL FIFO NOT FULL
 0F11  B0 20              C          MOV    AL,WDAT OR TYWORD OR MOREPL
 0F13  E6 A1              C          OUT    GDCCOM,AL       ;SEND COMMAND TO GDC
 0F15  E4 A0              C  XX16:   IN     AL,GDCSTA
 0F17  24 02              C          AND    AL,FIFULL
 0F19  75 FA              C          JNZ    XX16            ;LOOP UNTIL FIFO NOT FULL
 0F1B  8A C6              C          MOV    AL,DH
 0F1D  E6 A0              C          OUT    GDCPAR,AL       ;SEND PARAMETER TO GDC
 0F1F  E4 A0              C  XX17:   IN     AL,GDCSTA
 0F21  24 02              C          AND    AL,FIFULL
 0F23  75 FA              C          JNZ    XX17            ;LOOP UNTIL FIFO NOT FULL
 0F25  8A C2              C          MOV    AL,DL
 0F27  E6 A0              C          OUT    GDCPAR,AL       ;SEND PARAMETER TO GDC
 0F29  C3                 C          RET
                          C  ;
```

```
                           C  ;***   RDGCHR  - Read Graphics Character
                           C  ;              ENTRY - NONE
                           C  ;              EXIT  - DL = ATTRIBUTE
                           C  ;                      DH = CHARACTER
                           C  ;                      AL destroyed
                           C  ;
0F2A                       C  RDGCHR:
0F2A  E4 A0                C  XX2:    IN      AL,GDCSTA
0F2C  24 02                C          AND     AL,FIFULL
0F2E  75 FA                C          JNZ     XX2             ;LOOP UNTIL FIFO NOT FULL
0F30  B0 4C                C          MOV     AL,FIGS         ;FIGURE DRAWING PARAMETER
0F32  E6 A1                C          OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
0F34  E4 A0                C  XX18:   IN      AL,GDCSTA
0F36  24 02                C          AND     AL,FIFULL
0F38  75 FA                C          JNZ     XX18            ;LOOP UNTIL FIFO NOT FULL
0F3A  B0 02                C          MOV     AL,2            ;DIRECTION = 2
0F3C  E6 A0                C          OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
0F3E  E4 A0                C  XX19:   IN      AL,GDCSTA
0F40  24 02                C          AND     AL,FIFULL
0F42  75 FA                C          JNZ     XX19            ;LOOP UNTIL FIFO NOT FULL
0F44  B0 01                C          MOV     AL,1            ;DC = 1
0F46  E6 A0                C          OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
0F48  E4 A0                C  XX3:    IN      AL,GDCSTA
0F4A  24 02                C          AND     AL,FIFULL
0F4C  75 FA                C          JNZ     XX3             ;LOOP UNTIL FIFO NOT FULL
0F4E  B0 A0                C          MOV     AL,RDAT OR TYWORD       ;READ WORD FROM DISPLAY MEMORY
0F50  E6 A1                C          OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
0F52  E8 0FBF R            C          CALL    INPAR           ; GET ASCII CHARACTER
0F55  8A F0                C          MOV     DH,AL
0F57  E8 0FBF R            C          CALL    INPAR           ; GET ATTRIBUTE
0F5A  8A D0                C          MOV     DL,AL
0F5C  C3                   C          RET
                           C  ;
                           C  ;*** SPCLEAR1  ENTRY: BX = Cursor Posistion
                           C  ;                     CX = No. of bytes to clear
                           C  ;
                           C  ;
0F5D                       C  SPCLEAR1:
0F5D  03 D9                C          ADD     BX,CX
0F5F  81 FB 07D0           C          CMP     BX,07D0H
0F63  76 0E                C          JBE     SPCLEAR2        ; JUMP IF ENTIRE REGION TO CLEAR WITHIN 1ST PG
0F65  81 EB 07D0           C          SUB     BX,07D0H
0F69  E8 0F73 R            C          CALL    SPCLEAR2
0F6C  8B CB                C          MOV     CX,BX
0F6E  33 DB                C          XOR     BX,BX   ;ZERO OUT BX
0F70  E8 0FE4 R            C          CALL    SETCUR1
0F73                       C  SPCLEAR2:
0F73  49                   C          DEC     CX
0F74  E8 1010 R            C          CALL    SETMSK
0F77  E4 A0                C  XX4:    IN      AL,GDCSTA
0F79  24 02                C          AND     AL,FIFULL
0F7B  75 FA                C          JNZ     XX4             ;LOOP UNTIL FIFO NOT FULL
0F7D  B0 4C                C          MOV     AL,FIGS
0F7F  E6 A1                C          OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
0F81  E4 A0                C  XX20:   IN      AL,GDCSTA
0F83  24 02                C          AND     AL,FIFULL
0F85  75 FA                C          JNZ     XX20            ;LOOP UNTIL FIFO NOT FULL
0F87  B0 02                C          MOV     AL,2
0F89  E6 A0                C          OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
0F8B  E4 A0                C  XX21:   IN      AL,GDCSTA
0F8D  24 02                C          AND     AL,FIFULL
```

```
0F8F  75 FA            C          JNZ     XX21           ;LOOP UNTIL FIFO NOT FULL
0F91  8A C1            C          MOV     AL,CL
0F93  E6 A0            C          OUT     GDCPAR,AL      ;SEND PARAMETER TO GDC
0F95  E4 A0            C  XX22:   IN      AL,GDCSTA
0F97  24 02            C          AND     AL,FIFULL
0F99  75 FA            C          JNZ     XX22           ;LOOP UNTIL FIFO NOT FULL
0F9B  8A C5            C          MOV     AL,CH
0F9D  E6 A0            C          OUT     GDCPAR,AL      ;SEND PARAMETER TO GDC
0F9F  E4 A0            C  XX5:    IN      AL,GDCSTA
0FA1  24 02            C          AND     AL,FIFULL
0FA3  75 FA            C          JNZ     XX5            ;LOOP UNTIL FIFO NOT FULL
0FA5  B0 20            C          MOV     AL,WDAT OR TYWORD OR MOREPL
0FA7  E6 A1            C          OUT     GDCCOM,AL      ;SEND COMMAND TO GDC
0FA9  E4 A0            C  XX23:   IN      AL,GDCSTA
0FAB  24 02            C          AND     AL,FIFULL
0FAD  75 FA            C          JNZ     XX23           ;LOOP UNTIL FIFO NOT FULL
0FAF  B0 20            C          MOV     AL,020H
0FB1  E6 A0            C          OUT     GDCPAR,AL      ;SEND PARAMETER TO GDC
0FB3  E4 A0            C  XX24:   IN      AL,GDCSTA
0FB5  24 02            C          AND     AL,FIFULL
0FB7  75 FA            C          JNZ     XX24           ;LOOP UNTIL FIFO NOT FULL
0FB9  A0 0E4E R        C          MOV     AL,ATTRIBUTE   ;*** WHAT ABOUT COLOR? ***
0FBC  E6 A0            C          OUT     GDCPAR,AL      ;SEND PARAMETER TO GDC
0FBE  C3               C          RET
                       C  ;
0FBF                   C  INPAR:
0FBF  E4 A0            C          IN      AL,GDCSTA             ; READ GDC STATUS
0FC1  24 01            C          AND     AL,DATRDY
0FC3  74 FA            C          JZ      INPAR          ; AND WAIT IF NO CHARACTER READY
0FC5  E4 A1            C          IN      AL,FIFO
0FC7  C3               C          RET
                       C  ;
                       C  ;*** SENPAR  SEND PARAMETERS TO SCREEN
                       C  ;      ENTRY: BX = ADDRESS OF PARAMETER
                       C  ;             CX = LENGTH
                       C  ;      EXIT:  AL,BX,CX ARE DESTROYED
                       C  ;             AH,DX  ARE PRESERVED
                       C  ;
0FC8                   C  SENPAR:
0FC8  E4 A0            C  XX25:   IN      AL,GDCSTA
0FCA  24 02            C          AND     AL,FIFULL
0FCC  75 FA            C          JNZ     XX25           ;LOOP UNTIL FIFO NOT FULL
0FCE  8A 07            C          MOV     AL,0[BX]
0FD0  E6 A0            C          OUT     GDCPAR,AL      ;SEND PARAMETER TO GDC
0FD2  43               C          INC     BX             ; BUMP TO NEXT PARAMETER
0FD3  E2 F3            C          LOOP    SENPAR         ; LOOP UNTIL CX PARAMETERS HAVE BEEN SENT
0FD5  C3               C          RET
                       C  ;
                       C  ;*** SETCUR  - SET CURSOR
                       C  ;                ENTRY:   BX=GDC CURSOR POSITION
                       C  ;                EXIT:    AL,BX destroyed
                       C  ;                         CX,DX preserved
                       C  ;
0FD6                   C  SETCUR:
0FD6  03 1E 0E50 R     C          ADD     BX,SP1
0FDA  81 FB 07D0       C          CMP     BX,07D0H
0FDE  72 04            C          JB      SETCUR1
0FE0  81 EB 07D0       C          SUB     BX,07D0H
0FE4                   C  SETCUR1:
0FE4  E4 A0            C  XX6:    IN      AL,GDCSTA
```

```
0FE6  24 02            C          AND    AL,FIFULL
0FE8  75 FA            C          JNZ    XX6          ;LOOP UNTIL FIFO NOT FULL
0FEA  B0 49            C          MOV    AL,CURS
0FEC  E6 A1            C          OUT    GDCCOM,AL    ;SEND COMMAND TO GDC
0FEE  E4 A0            C  XX26:   IN     AL,GDCSTA
0FF0  24 02            C          AND    AL,FIFULL
0FF2  75 FA            C          JNZ    XX26         ;LOOP UNTIL FIFO NOT FULL
0FF4  8A C3            C          MOV    AL,BL
0FF6  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
0FF8  E4 A0            C  XX27:   IN     AL,GDCSTA
0FFA  24 02            C          AND    AL,FIFULL
0FFC  75 FA            C          JNZ    XX27         ;LOOP UNTIL FIFO NOT FULL
0FFE  8A C7            C          MOV    AL,BH
1000  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
1002  E4 A0            C  XX28:   IN     AL,GDCSTA
1004  24 02            C          AND    AL,FIFULL
1006  75 FA            C          JNZ    XX28         ;LOOP UNTIL FIFO NOT FULL
1008  32 C0            C          XOR    AL,AL
100A  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
100C  E8 1010 R        C          CALL   SETMSK
100F  C3               C          RET
                       C  ;
                       C  ;*** SETMASK ROUTINE    (AL destroyed, all other registers preserved)
                       C  ;
1010                   C  SETMSK:
1010  E4 A0            C  XX7:    IN     AL,GDCSTA
1012  24 02            C          AND    AL,FIFULL
1014  75 FA            C          JNZ    XX7          ;LOOP UNTIL FIFO NOT FULL
1016  B0 4A            C          MOV    AL,MASKREG
1018  E6 A1            C          OUT    GDCCOM,AL    ;SEND COMMAND TO GDC
101A  E4 A0            C  XX29:   IN     AL,GDCSTA
101C  24 02            C          AND    AL,FIFULL
101E  75 FA            C          JNZ    XX29         ;LOOP UNTIL FIFO NOT FULL
1020  B0 FF            C          MOV    AL,-1
1022  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
1024  E4 A0            C  XX30:   IN     AL,GDCSTA
1026  24 02            C          AND    AL,FIFULL
1028  75 FA            C          JNZ    XX30         ;LOOP UNTIL FIFO NOT FULL
102A  B0 FF            C          MOV    AL,-1
102C  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
102E  C3               C          RET
                       C  ;
                       C  ;*** RDLIN      READ 1 ROW INTO LINBUF
                       C  ;
                       C  ;        Entry registers: none
                       C  ;        Exit registers: AL, BX, CX destroyed
                       C  ;                        DX preserved
                       C  ;
102F                   C  RDLIN:
102F  E4 A0            C  XX8:    IN     AL,GDCSTA
1031  24 02            C          AND    AL,FIFULL
1033  75 FA            C          JNZ    XX8          ;LOOP UNTIL FIFO NOT FULL
1035  B0 4C            C          MOV    AL,FIGS
1037  E6 A1            C          OUT    GDCCOM,AL    ;SEND COMMAND TO GDC
1039  E4 A0            C  XX31:   IN     AL,GDCSTA
103B  24 02            C          AND    AL,FIFULL
103D  75 FA            C          JNZ    XX31         ;LOOP UNTIL FIFO NOT FULL
103F  B0 02            C          MOV    AL,2         ;DIRECTION = 2
1041  E6 A0            C          OUT    GDCPAR,AL    ;SEND PARAMETER TO GDC
1043  E4 A0            C  XX32:   IN     AL,GDCSTA
```

IO.SYS

```
1045  24 02              C         AND     AL,FIFULL
1047  75 FA              C         JNZ     XX32            ;LOOP UNTIL FIFO NOT FULL
1049  B0 50              C         MOV     AL,80           ;LENGTH = 80 WORDS [CHAR + ATTR]
104B  E6 A0              C         OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
104D  E4 A0              C  XX33:  IN      AL,GDCSTA
104F  24 02              C         AND     AL,FIFULL
1051  75 FA              C         JNZ     XX33            ;LOOP UNTIL FIFO NOT FULL
1053  32 C0              C         XOR     AL,AL
1055  E6 A0              C         OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1057  E4 A0              C  XX9:   IN      AL,GDCSTA
1059  24 02              C         AND     AL,FIFULL
105B  75 FA              C         JNZ     XX9             ;LOOP UNTIL FIFO NOT FULL
105D  B0 A0              C         MOV     AL,RDAT
105F  E6 A1              C         OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
1061  BB 071A R          C         MOV     BX,OFFSET LINBUF
1064  B9 00A0            C         MOV     CX,160          ; FOR READ LOOP
1067                     C  RDLIN1:
1067  E8 0FBF R          C         CALL    INPAR
106A  88 07              C         MOV     0[BX],AL
106C  43                 C         INC     BX
106D  E2 F8              C         LOOP    RDLIN1
106F  C3                 C         RET
                         C  ;
                         C  ;*** WRLIN  WRITE 1 ROW INTO GDC
                         C  ;
                         C  ;      Entry registers: none
                         C  ;      Exit:            AL, BX, CX destoyed
                         C  ;                       DX preserved
                         C  ;
1070                     C  WRLIN:
1070  E4 A0              C  XX10:  IN      AL,GDCSTA
1072  24 02              C         AND     AL,FIFULL
1074  75 FA              C         JNZ     XX10            ;LOOP UNTIL FIFO NOT FULL
1076  B0 4C              C         MOV     AL,FIGS
1078  E6 A1              C         OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
107A  E4 A0              C  XX34:  IN      AL,GDCSTA
107C  24 02              C         AND     AL,FIFULL
107E  75 FA              C         JNZ     XX34            ;LOOP UNTIL FIFO NOT FULL
1080  B0 02              C         MOV     AL,2
1082  E6 A0              C         OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1084  E4 A0              C  XX35:  IN      AL,GDCSTA
1086  24 02              C         AND     AL,FIFULL
1088  75 FA              C         JNZ     XX35            ;LOOP UNTIL FIFO NOT FULL
108A  32 C0              C         XOR     AL,AL
108C  E6 A0              C         OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
108E  E4 A0              C  XX36:  IN      AL,GDCSTA
1090  24 02              C         AND     AL,FIFULL
1092  75 FA              C         JNZ     XX36            ;LOOP UNTIL FIFO NOT FULL
1094  32 C0              C         XOR     AL,AL
1096  E6 A0              C         OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1098  E4 A0              C  XX11:  IN      AL,GDCSTA
109A  24 02              C         AND     AL,FIFULL
109C  75 FA              C         JNZ     XX11            ;LOOP UNTIL FIFO NOT FULL
109E  B0 20              C         MOV     AL,WDAT OR TYWORD OR MOREPL
10A0  E6 A1              C         OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
10A2  BB 071A R          C         MOV     BX,OFFSET LINBUF
10A5  B9 00A0            C         MOV     CX,160          ; FOR WRITE LOOP
10A8                     C  WRLIN1:
10A8  E4 A0              C  XX37:  IN      AL,GDCSTA
10AA  24 02              C         AND     AL,FIFULL
```

```
10AC  75 FA              C          JNZ     XX37          ;LOOP UNTIL FIFO NOT FULL
10AE  8A 07              C          MOV     AL,0[BX]
10B0  E6 A0              C          OUT     GDCPAR,AL     ;SEND PARAMETER TO GDC
10B2  43                 C          INC     BX
10B3  E2 F3              C          LOOP    WRLIN1
10B5  C3                 C          RET
                         C  ;
                         C  ;*** CUROFF     ROUTINE TO TURN CURSOR OFF   (destroys AL)
                         C  ;
10B6                     C  CUROFF:
10B6  E4 A0              C  XX12:   IN      AL,GDCSTA
10B8  24 02              C          AND     AL,FIFULL
10BA  75 FA              C          JNZ     XX12          ;LOOP UNTIL FIFO NOT FULL
10BC  B0 4B              C          MOV     AL,CCHAR
10BE  E6 A1              C          OUT     GDCCOM,AL     ;SEND COMMAND TO GDC
10C0  E4 A0              C  XX38:   IN      AL,GDCSTA
10C2  24 02              C          AND     AL,FIFULL
10C4  75 FA              C          JNZ     XX38          ;LOOP UNTIL FIFO NOT FULL
10C6  B0 0F              C          MOV     AL,0FH
10C8  E6 A0              C          OUT     GDCPAR,AL     ;SEND PARAMETER TO GDC
10CA  C3                 C          RET
                         C  ;
                         C  ;*** CURON      ROUTINE TO TURN CURSOR ON    (destoyes AL)
                         C  ;
10CB                     C  CURON:
10CB  E4 A0              C  XX13:   IN      AL,GDCSTA
10CD  24 02              C          AND     AL,FIFULL
10CF  75 FA              C          JNZ     XX13          ;LOOP UNTIL FIFO NOT FULL
10D1  B0 4B              C          MOV     AL,CCHAR
10D3  E6 A1              C          OUT     GDCCOM,AL     ;SEND COMMAND TO GDC
10D5  E4 A0              C  XX39:   IN      AL,GDCSTA
10D7  24 02              C          AND     AL,FIFULL
10D9  75 FA              C          JNZ     XX39          ;LOOP UNTIL FIFO NOT FULL
10DB  B0 8F              C          MOV     AL,08FH
10DD  E6 A0              C          OUT     GDCPAR,AL     ;SEND PARAMETER TO GDC
10DF  E4 A0              C  XX40:   IN      AL,GDCSTA
10E1  24 02              C          AND     AL,FIFULL
10E3  75 FA              C          JNZ     XX40          ;LOOP UNTIL FIFO NOT FULL
10E5  B0 CE              C          MOV     AL,0CEH
10E7  E6 A0              C          OUT     GDCPAR,AL     ;SEND PARAMETER TO GDC
10E9  E4 A0              C  XX41:   IN      AL,GDCSTA
10EB  24 02              C          AND     AL,FIFULL
10ED  75 FA              C          JNZ     XX41          ;LOOP UNTIL FIFO NOT FULL
10EF  B0 72              C          MOV     AL,072H
10F1  E6 A0              C          OUT     GDCPAR,AL     ;SEND PARAMETER TO GDC
10F3  C3                 C          RET
                         C  ;
                         C  ;*** INIT10   INITIALIZE SCREEN PAGE VALUES
                         C  ;
10F4                     C  INIT10:
10F4  33 C0              C          XOR     AX,AX
10F6  A3 0E50 R          C          MOV     SP1,AX
10F9  A3 0E54 R          C          MOV     SP2,AX        ; START OF PAGES 1 AND 2 = 0
10FC  A2 0E57 R          C          MOV     LP22,AL       ; LENGTH OF PAGE 2 = 0
10FF  C6 06 0E53 R 19    C          MOV     LP12,25       ; LENGTH OF PAGE 1 = 25
1104  E4 A0              C  XX14:   IN      AL,GDCSTA
1106  24 02              C          AND     AL,FIFULL
1108  75 FA              C          JNZ     XX14          ;LOOP UNTIL FIFO NOT FULL
110A  B0 4C              C          MOV     AL,FIGS
110C  E6 A1              C          OUT     GDCCOM,AL     ;SEND COMMAND TO GDC
```

```
110E  E4 A0             C  XX42:   IN      AL,GDCSTA
1110  24 02             C          AND     AL,FIFULL
1112  75 FA             C          JNZ     XX42            ;LOOP UNTIL FIFO NOT FULL
1114  B0 02             C          MOV     AL,2
1116  E6 A0             C          OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1118  C3                C          RET
                        C  ;
                        C  ;***    SCROLL ROUTINE
                        C  ;
1119                    C  SCROLLX:
1119  33 DB             C          XOR     BX,BX           ; START OF PAGE 1
111B  B9 0050           C          MOV     CX,80
111E  E8 1293 R         C          CALL    SPCLEAR
1121  8B 1E 0E50 R      C          MOV     BX,SP1
1125  83 C3 50          C          ADD     BX,80
1128  89 1E 0E50 R      C          MOV     SP1,BX
112C  FE 0E 0E53 R      C          DEC     LP12
1130  75 06             C          JNZ     SCROL2
1132  E8 10F4 R         C          CALL    INIT10
1135  E8 05 90          C          JMP     SCROL1
1138                    C  SCROL2:
1138  FE 06 0E57 R      C          INC     LP22
113C                    C  SCROL1:
113C  E4 A0             C  XX15:   IN      AL,GDCSTA
113E  24 02             C          AND     AL,FIFULL
1140  75 FA             C          JNZ     XX15            ;LOOP UNTIL FIFO NOT FULL
1142  B0 70             C          MOV     AL,PRAM+0       ;SCROL1 SENDS THE 8 BYTE SCREEN PAGES INFO
1144  E6 A1             C          OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
1146  B9 0008           C          MOV     CX,8
1149  BB 0E50 R         C          MOV     BX,OFFSET SP1
114C  E8 0FC8 R         C          CALL    SENPAR
114F  C3                C          RET
                        C  ;
                        C  ;***  BUMPCUR  -  BUMP CURSOR AND UPDATE CURCOL & CURROW
                        C  ;                         CRTPB WILL BE UPDATED WITH THESE VALUES
                        C  ;                         BEFOR EXITING THE CRTPIM
                        C  ;
1150                    C  BUMPCUR:
1150  FE 06 0E4C R      C          INC     CURCOL
1154  80 3E 0E4C R 50   C          CMP     CURCOL,SCWID
1159  73 01             C          JAE     BMPCR1          ; JUMP IF CURCOL+1 IS GREATER THAN 80
115B  C3                C          RET
115C                    C  BMPCR1:
115C  80 3E 0E4D R 17   C          CMP     CURROW,ROWS-1
1161  75 01             C          JNZ     BMPCR2          ; IF WE ARE ON LAST ROW, DO NOTHING (WILL BE
1163  C3                C          RET                     ;      CHECKED LATER FOR SCROLLING)
1164                    C  BMPCR2:
1164  C6 06 0E4C R 00   C          MOV     CURCOL,0
1169  FE 06 0E4D R      C          INC     CURROW
116D  E8 1171 R         C          CALL    WRITEPOS
1170  C3                C          RET
                        C  ;
                        C  ;***   WRITEPOS    WRITE CURSOR POSITION ROUTINE
                        C  ;                  ENTRY: NONE
                        C  ;                  EXIT: AL, BX     -DESTROYED
                        C  ;                        AH, CX, DX -PRESERVED
                        C  ;
1171                    C  WRITEPOS:
1171  8B 1E 0E4C R      C          MOV     BX,WORD PTR CURCOL
1175  E8 117C R         C          CALL    WRHLPOS         ; COMPUTE ADDRESS IN CRT BUFFER
```

```
1178  E8 0FD6 R           C          CALL    SETCUR
117B  C3                  C          RET
                          C  ;
                          C  ;***   WRHLPOS  COMPUTE ADDRESS WITHIN CRT-BUFFER
                          C  ;               ENTER - BL = COLUMN
                          C  ;                       BH = ROW
                          C  ;               EXIT  - BX = ADDRESS IN CRT BUFFER
                          C  ;                       AX, CX, DX -PRESERVED
                          C  ;
117C                      C  WRHLPOS:
117C  50                  C          PUSH    AX
117D  B0 50               C          MOV     AL,SCWID          ; CHARS/ROW IN AL
117F  F6 E7               C          MUL     BH                ; MULTIPLY BY ROW NO. - RESULT IN AX
1181  32 FF               C          XOR     BH;BH             ; BH = 0
1183  03 D8               C          ADD     BX,AX             ; NOW BX IS CORRECT POSITION IN CRT BUFFER
1185  58                  C          POP     AX
1186  C3                  C          RET
                          C  ;
                          C  ;***  MUSIC     PLAY MUSIC
                          C  ;
1187                      C  MUSIC:
1187  B1 06               C          MOV     CL,06
1189  E8 0000 E           C          CALL    KBD_OUT           ; CALL KEYBOARD PIM WITH MUSIC FUNCTION CODE
118C  8A 4F 04            C          MOV     CL,CPB_FREQ[BX]
118F  E8 0000 E           C          CALL    KBD_OUT           ; SEND FREQUENCE TO KEYBOARD
1192  8A 4F 05            C          MOV     CL,CPB_FLEN[BX]
1195  E8 0000 E           C          CALL    KBD_OUT           ; SEND LENGTH OF FREQUENCE TO KEYBOARD
1198  C3                  C          RET
                          C  ;
                          C  ;***  INSCHR    INSERT CHARACTER ROUTINE
                          C  ;
1199                      C  INSCHR:
1199  E8 11F8 R           C          CALL    TEST_POS
119C  74 27               C          JZ      BLANK_ONE
119E  8A 3E 0E4D R        C          MOV     BH,CURROW
11A2  B3 4E               C          MOV     BL,SCWID-2
11A4  E8 117C R           C          CALL    WRHLPOS           ; GET CHARACTER POINTER IN BX
11A7  E8 10B6 R           C          CALL    CUROFF            ; SWITCH CURSOR OFF
11AA                      C  INSCH1:
11AA  53                  C          PUSH    BX
11AB  E8 0FD6 R           C          CALL    SETCUR            ; SET CURSOR
11AE  E8 0F2A R           C          CALL    RDGCHR            ; GET CHARACTER
11B1  5B                  C          POP     BX
11B2  43                  C          INC     BX
11B3  53                  C          PUSH    BX
11B4  E8 0FD6 R           C          CALL    SETCUR            ; SET CURSOR
11B7  E8 0F0B R           C          CALL    WRGCHR            ; SET CHARACTER
11BA  5B                  C          POP     BX
11BB  4B                  C          DEC     BX
11BC  4B                  C          DEC     BX
11BD  FE C9               C          DEC     CL                ; DECREMENT COUNTER
11BF  75 E9               C          JNZ     INSCH1            ; LOOP UNTIL ZERO
11C1  E8 10CB R           C          CALL    CURON             ; SWITCH CURSOR ON
11C4  43                  C          INC     BX
11C5                      C  BLANK_ONE:
11C5  B6 20               C          MOV     DH,' '            ; CHARACTER REQUIRED IN DH
11C7  E8 0FD6 R           C          CALL    SETCUR            ; SET CURSOR
11CA  8A 16 0E4E R        C          MOV     DL,ATTRIBUTE      ; GET ATTRIBUTE
11CE  E8 0F0B R           C          CALL    WRGCHR            ; CLEAR CHARACTER
11D1  E8 1171 R           C          CALL    WRITEPOS          ; SET CURSOR
```

```
11D4  C3                 C          RET
                         C  ;
                         C  ;*** DELCHR     DELETE ONE CHARACTER
                         C  ;
11D5                     C  DELCHR:
11D5  E8 11FB R          C          CALL    TEST_POS        ; RETURNS: CL = NO. OF POSITIONS TO MOVE
                         C                                  ;          BX = ROW*80+COL
                         C                                  ;          ZF SET IF ZERO POSITIONS TO MOVE
11D8  74 EB              C          JZ      BLANK_ONE       ; EXIT IF NONE TO MOVE
11DA  43                 C          INC     BX              ; START AT PRES + 1
11DB  E8 10B6 R          C          CALL    CUROFF          ; SWITCH OFF CURSOR
11DE                     C  DELCHR1:
11DE  53                 C          PUSH    BX
11DF  E8 0FD6 R          C          CALL    SETCUR          ; SET CURSOR
11E2  E8 0F2A R          C          CALL    RDGCHR          ; GET CHARACTER
11E5  5B                 C          POP     BX
11E6  4B                 C          DEC     BX
11E7  53                 C          PUSH    BX
11E8  E8 0FD6 R          C          CALL    SETCUR          ; SET CURSOR
11EB  E8 0F0B R          C          CALL    WRGCHR          ; SET CHARACTER
11EE  5B                 C          POP     BX
11EF  43                 C          INC     BX
11F0  43                 C          INC     BX
11F1  FE C9              C          DEC     CL              ; DECREMENT COUNTER OF CHARACTER TO MOVE
11F3  75 E9              C          JNZ     DELCHR1         ; LOOP UNTIL ZERO  (*Corrected 1.07*)
11F5  E8 10C8 R          C          CALL    CURON           ; SWITCH ON CURSOR
11F8  4B                 C          DEC     BX
11F9  EB CA              C          JMP     BLANK_ONE
                         C  ;
                         C  ;***  TEST_POS   RETURNS CURSOR POSITION AND LENGTH
                         C  ;               ENTRY REGS: NONE
                         C  ;               EXIT REGS:  BX = CUR POSITION (ROW*80+COL)
                         C  ;                           CL = LENGTH TO MOVE
                         C  ;                     ZF SET TO ZERO MEANS NO CHARACTERS TO MOVE!
11FB                     C  TEST_POS:
11FB  8B 1E 0E4C R       C          MOV     BX,WORD PTR CURCOL      ; BL = COLUMN ; BH = ROW
11FF  E8 117C R          C          CALL    WRHLPOS         ; COMPUTE ADDRESS WITHIN CRT BUFFER
1202  B1 4F              C          MOV     CL,SCWID-1      ; TEST IF CURRENT COLUMN = SCWID-1
1204  2A 0E 0E4C R       C          SUB     CL,CURCOL       ; CL = COUNT
1208  C3                 C          RET
                         C  ;
                         C  ;***    ICLEOL  ERASE TO END OF LINE
                         C  ;
1209                     C  ICLEOL:
1209  8A 1E 0E4C R       C          MOV     BL,CURCOL       ; CURRENT COLUMN NUMBER TO CH AND BL
120D  8A EB              C          MOV     CH,BL
120F  B0 50              C          MOV     AL,SCWID        ; SUBTRACT COLUMN NUMBER FROM SCREEN WIDTH TC
1211  2A C5              C          SUB     AL,CH           ;       GET NUMBER OF BYTES TO CLEAR
1213  74 14              C          JZ      ICLEOL_RET
1215  8A C8              C          MOV     CL,AL           ; CX = NUMBER OF BYTES TO CLEAR
1217  32 ED              C          XOR     CH,CH
1219  51                 C          PUSH    CX
121A  A0 0E4D R          C          MOV     AL,CURROW
121D  8A F8              C          MOV     BH,AL
121F  E8 117C R          C          CALL    WRHLPOS         ; BX = ADDRESS OF CHARACTER IN CRT RAM
1222  59                 C          POP     CX              ; CX = NUMBER OF BYTES TO CLEAR
1223  E8 1293 R          C          CALL    SPCLEAR         ; CLEAR
1226  E8 1171 R          C          CALL    WRITEPOS
1229  C3                 C  ICLEOL_RET:     RET
                         C  ;
```

```
                              C ;***    CLEOS   CLEAR FROM CURRENT ROW TO END OF SCREEN
                              C ;
122A                          C CLEOS:
122A  B0 17                   C         MOV     AL,ROWS-1        ; CALCULATE NUMBER OF ROWS TO BE CLEARED
122C  2A 06 0E4D R            C         SUB     AL,CURROW
1230  74 1A                   C         JZ      CLEOS1           ; IF ZERO, JUST CLEAR CURRENT ROW
1232  8A 3E 0E4D R            C         MOV     BH,CURROW
1236  FE C7                   C         INC     BH               ; BH = CURRENT ROW + 1
1238  32 DB                   C         XOR     BL,BL            ; BL = 0  (COLLUMN 0)
123A  E8 117C R               C         CALL    WRHLPOS
123D  B2 50                   C         MOV     DL,SCWID
123F  F6 E2                   C         MUL     DL               ; AX = NUMBER OF BYTES TO CLEAR
1241  8B C8                   C         MOV     CX,AX
1243  E8 10B6 R               C         CALL    CUROFF           ; SWITCH OFF CURSOR
1246  E8 1293 R               C         CALL    SPCLEAR          ; CLEAR TO SPACES
1249  E8 10CB R               C         CALL    CURON            ; SWITCH ON CURSOR
124C                          C CLEOS1:
124C  E8 1209 R               C         CALL    ICLEOL
124F  C3                      C         RET
                              C ;
                              C ;***  IHOME   PHYSICAL HOME CURSOR
                              C ;
1250                          C IHOME:
1250  C7 06 0E4C R 0000       C         MOV     WORD PTR CURCOL,0        ; ZERO OUT CURCOL AND CURROW
1256  E8 1171 R               C         CALL    WRITEPOS
1259  C3                      C         RET
                              C ;
                              C ;***    ILF     INTERNAL LINE FEED
                              C ;
125A                          C ILF:
125A  A0 0E4D R               C         MOV     AL,CURROW
125D  FE C0                   C         INC     AL
125F  3C 18                   C         CMP     AL,ROWS
1261  73 07                   C         JAE     ILF1
1263  A2 0E4D R               C         MOV     CURROW,AL
1266  E8 1171 R               C         CALL    WRITEPOS
1269  C3                      C         RET
126A                          C ILF1:
126A  EB 67 90                C         JMP     SCLUP3
                              C ;
                              C ;***  VCLEAR  CLEAR SCREEN; HOME CURSOR
                              C ;
126D                          C VCLEAR:
126D  E8 10B6 R               C         CALL    CUROFF  ; CURSOR OFF
1270  E8 10F4 R               C         CALL    INIT10
1273  E8 113C R               C         CALL    SCROL1  ; INITIALIZE PAGES
1276  BB 0000                 C         MOV     BX,0
1279  B9 0700                 C         MOV     CX,ROWS*SCWID+SCWID
127C  E8 1293 R               C         CALL    SPCLEAR ; DO IT TO THE SCREEN
127F  E8 1250 R               C         CALL    IHOME
1282  E8 10CB R               C         CALL    CURON   ; TURN CURSOR BACK ON
1285  C3                      C         RET             ; *** ADDITION OF RETURN ** Change 1.09 **
                              C ;
                              C ;*** CLRLIN   CLEAR ROW (AL) TO SPACES
                              C ;
1286                          C CLRLIN:
1286  B3 50                   C         MOV     BL,SCWID
1288  F6 E3                   C         MUL     BL      ; CALCULATE ABSOLUTE CURSOR POSITION
128A  8B D8                   C         MOV     BX,AX   ; AND MOVE IT TO BX
128C  B9 0050                 C         MOV     CX,SCWID
```

IO.SYS

```
128F  E8 1293 R             C         CALL    SPCLEAR
1292  C3                    C         RET
                            C ;
                            C ;*** SPCLEAR   ENTRY: BX - START ADDRESS IN CRT RAM
                            C ;                     CX - NO. OF BYTES TO CLEAR
                            C ;              EXIT: ALL REGISTERS DESTROYED!
                            C ;
1293                        C SPCLEAR:
1293  90                    C         NOP     ;*10
1294  E8 0FD6 R             C         CALL    SETCUR  ; SET CURSOR
1297  90                    C         NOP     ;*10
1298  E8 0F5D R             C         CALL    SPCLEAR1
129B  E8 1171 R             C         CALL    WRITEPOS
129E  C3                    C         RET
                            C ;
                            C ;*** SCROLLUP
                            C ;
                            C ;       ENTRY REGISTERS: NONE
                            C ;       EXIT REGISTERS:  ALL REGISTERS DESTROYED!
129F                        C SCROLLUP:
129F  A0 0E4D R             C         MOV     AL,CURROW
12A2  0A C0                 C         OR      AL,AL
12A4  74 2D                 C         JZ      SCLUP3
12A6  8A E8                 C         MOV     CH,AL           ; CH = ROW NO.
12A8  B0 17                 C         MOV     AL,ROWS-1
12AA  2A C5                 C         SUB     AL,CH
12AC  74 11                 C         JZ      SCLUP2
12AE  8A C8                 C         MOV     CL,AL           ; CL = NO. OF ROWS TO MOVE
12B0  E8 10B6 R             C         CALL    CUROFF          ; TURN OFF CURSOR
12B3                        C SCLUP1:
12B3  E8 1323 R             C         CALL    MUROW   ; ROW NO. IN CH
12B6  FE C5                 C         INC     CH              ; INCREMENT ROW NO.
12B8  FE C9                 C         DEC     CL              ; DECREMENT NO. OF ROWS TO MOVE
12BA  75 F7                 C         JNZ     SCLUP1
12BC  E8 10CB R             C         CALL    CURON           ; TURN CURSOR BACK ON
                            C ;
12BF                        C SCLUP2:
12BF  B0 17                 C         MOV     AL,ROWS-1
12C1  E8 1286 R             C         CALL    CLRLIN          ; CLEAR LINE
12C4  C6 06 0E4C R 00       C         MOV     CURCOL,0
12C9  E8 1171 R             C         CALL    WRITEPOS
12CC  C3                    C         RET
12CD                        C SCLUP4:
12CD  C7 06 0E4C R 1700     C         MOV     WORD PTR CURCOL,1700H   ; LOAD COL/ROW WITH 0/23  * CHANGE 05*
12D3                        C SCLUP3:
12D3  E8 10B6 R             C         CALL    CUROFF
12D6  BB 0780               C         MOV     BX,24*80
12D9  E8 0FD6 R             C         CALL    SETCUR
12DC  E8 102F R             C         CALL    RDLIN
12DF  BB 0780               C         MOV     BX,24*80
12E2  B9 0050               C         MOV     CX,80           ; *03* MOV CH corrected to MOV CX
12E5  E8 1293 R             C         CALL    SPCLEAR         ; CLEAR STATUS LINE
12E8  E8 1119 R             C         CALL    SCROLLX
12EB  BB 0780               C         MOV     BX,24*80
12EE  E8 0FD6 R             C         CALL    SETCUR
12F1  E8 1070 R             C         CALL    WRLIN
12F4  E8 10CB R             C         CALL    CURON
12F7  E8 1171 R             C         CALL    WRITEPOS
12FA  C3                    C         RET
                            C ;
```

```
                          C ;*** SCOLLDN  - SCROLL DOWN  - ENTRY REGISTERS: NONE
                          C ;                              EXIT REGISTERS: ALL DESTROYED!
                          C ;
12FB                      C SCROLLDN:
12FB  A0 0E4D R           C         MOV     AL,CURROW
12FE  50                  C         PUSH    AX
12FF  B1 17               C         MOV     CL,ROWS-1
1301  2A C8               C         SUB     CL,AL         ; CL = ROWS TO MOVE
1303  74 11               C         JZ      SCLDN2
1305  B5 16               C         MOV     CH,ROWS-2     ; CH = ROW TO START     *Change 06*
1307  E8 10B6 R           C         CALL    CUROFF
130A                      C SCLDN1:
130A  E8 1341 R           C         CALL    MDROW
130D  FE CD               C         DEC     CH
130F  FE C9               C         DEC     CL
1311  75 F7               C         JNZ     SCLDN1
1313  E8 10CB R           C         CALL    CURON
1316                      C SCLDN2:
1316  58                  C         POP     AX
1317  E8 1286 R           C         CALL    CLRLIN        ; CLEAR CURRENT LINE
131A  C6 06 0E4C R 00     C         MOV     CURCOL,0
131F  E8 1171 R           C         CALL    WRITEPOS
1322  C3                  C         RET
                          C ;
                          C ;*** MUROW  MOVE ROW UP  - MOVE ROW [CH+1] TO ROW CH
                          C ;
                          C ;       Entry Register: CH = Row
                          C ;       Exit:           CX - Preserved  (Both CH and CL must be preserved!)
                          C ;                       AX, BX, DX  Destroyed
                          C ;
1323                      C MUROW:
1323  51                  C         PUSH    CX
1324  8A C5               C         MOV     AL,CH
1326  B1 50               C         MOV     CL,SCWID
1328  F6 E1               C         MUL     CL            ; AX = ROW * CHR/ROW
132A  8B D0               C         MOV     DX,AX
132C  05 0050             C         ADD     AX,SCWID      ; AX = (ROW+1)*(CHR/ROW)
132F  8B D8               C         MOV     BX,AX         ; DX = ROW B; BX = ROW B+1
1331  E8 0FD6 R           C         CALL    SETCUR        ; CURSOR TO THE START OF ROW B+1
1334  E8 102F R           C         CALL    RDLIN         ; READ IN A ROW (CHAR AND ATTRIBUTE)
1337  8B DA               C         MOV     BX,DX         ; NOW SET CURSOR TO START OF ROW B
1339  E8 0FD6 R           C         CALL    SETCUR
133C  E8 1070 R           C         CALL    WRLIN         ; WRITE OUT A ROW
133F  59                  C         POP     CX
1340  C3                  C         RET
                          C ;
                          C ;*** MDROW  MOVE A ROW DOWN
                          C ;
                          C ;       Entry:  CH = row number
                          C ;       Exit:   AX, BX, DX destroyed
                          C ;               CX preserved
                          C ;
1341                      C MDROW:
1341  51                  C         PUSH    CX
1342  8A C5               C         MOV     AL,CH
1344  B1 50               C         MOV     CL,SCWID
1346  F6 E1               C         MUL     CL            ; MULTIPLY ROW NO. TIMES CHAR/ROW
1348  8B D8               C         MOV     BX,AX
134A  8B D0               C         MOV     DX,AX
134C  E8 0FD6 R           C         CALL    SETCUR        ; SET CURSOR TO START OF ROW B
```

IO.SYS

```
 134F  E8 102F R            C          CALL    RDLIN          ; READ IN A ROW TO LINBUF  .
 1352  8B DA                C          MOV     BX,DX
 1354  83 C3 50             C          ADD     BX,SCWID
 1357  E8 0FD6 R            C          CALL    SETCUR         ; SET CURSOR TO START OF ROW B+1
 135A  E8 1070 R            C          CALL    WRLIN          ; WRITE ROW IN LINBUF
 135D  59                   C          POP     CX
 135E  C3                   C          RET
                            C


 135F                          CSEG    ENDS

                                       END
```

```
IO.SYS


Structures and records:

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

IODAT. . . . . . . . . . . . . . .      0016    0009
  CMDLEN . . . . . . . . . . . . . .    0000
  UNIT . . . . . . . . . . . . . . .    0001
  CMD. . . . . . . . . . . . . . . .    0002
  STATUS . . . . . . . . . . . . . .    0003
  MEDIA. . . . . . . . . . . . . . .    000D
  TRANS. . . . . . . . . . . . . . .    000E
  COUNT. . . . . . . . . . . . . . .    0012
  BEGIN. . . . . . . . . . . . . . .    0014

Segments and groups:

                N a m e                 Size    align   combine class

CSEG . . . . . . . . . . . . . . .      135F    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                 Type    Value   Attr

ANSI_ESC_SEQ . . . . . . . . . . .      L WORD  0005    CSEG
ATTRIBUTE. . . . . . . . . . . . .      L BYTE  0E4E    CSEG
ATTR_MASK. . . . . . . . . . . . .      Number  0080
AUSTRALIA. . . . . . . . . . . . .      L NEAR  0346    CSEG
AUXDEV . . . . . . . . . . . . . .      L NEAR  0488    CSEG
AUXTBL . . . . . . . . . . . . . .      L NEAR  04EA    CSEG
AUX_INT. . . . . . . . . . . . . .      L NEAR  054D    CSEG
BACKSP . . . . . . . . . . . . . .      L NEAR  09A8    CSEG
BELL . . . . . . . . . . . . . . .      L NEAR  09A2    CSEG
BG_FG. . . . . . . . . . . . . . .      L BYTE  060E    CSEG
BIOSSEG. . . . . . . . . . . . . .      Number  0040
BKSP1. . . . . . . . . . . . . . .      L NEAR  09B9    CSEG
BKSP2. . . . . . . . . . . . . . .      L NEAR  09B7    CSEG
BLANK_ONE. . . . . . . . . . . . .      L NEAR  11C5    CSEG
BLEOS. . . . . . . . . . . . . . .      L NEAR  0ADE    CSEG
BMPCR1 . . . . . . . . . . . . . .      L NEAR  115C    CSEG
BMPCR2 . . . . . . . . . . . . . .      L NEAR  1164    CSEG
BUMPCUR. . . . . . . . . . . . . .      L NEAR  1150    CSEG
BUSY . . . . . . . . . . . . . . .      Number  0020
BUS_EXIT . . . . . . . . . . . . .      L NEAR  058C    CSEG    Global
CANADA2. . . . . . . . . . . . . .      L NEAR  0349    CSEG
CARRET . . . . . . . . . . . . . .      L NEAR  09D8    CSEG
CBACK. . . . . . . . . . . . . . .      L NEAR  099A    CSEG
CCHAR. . . . . . . . . . . . . . .      Number  004B
CCNTCH . . . . . . . . . . . . . .      L NEAR  08A7    CSEG
CDOWN. . . . . . . . . . . . . . .      L NEAR  097B    CSEG
CFORW. . . . . . . . . . . . . . .      L NEAR  098B    CSEG
CHAMOD . . . . . . . . . . . . . .      Number  0020
CHRTRN . . . . . . . . . . . . . .      L NEAR  088B    CSEG    Global
CHTRANS. . . . . . . . . . . . . .      L WORD  0613    CSEG    Global
CLEAR_1. . . . . . . . . . . . . .      L BYTE  0223    CSEG    Global
CLEAR_2. . . . . . . . . . . . . .      L BYTE  0233    CSEG    Global
CLEOS. . . . . . . . . . . . . . .      L NEAR  122A    CSEG
CLEOS1 . . . . . . . . . . . . . .      L NEAR  124C    CSEG
CLRLIN . . . . . . . . . . . . . .      L NEAR  1286    CSEG
CL_MASK. . . . . . . . . . . . . .      Number  0040
```

```
IO.SYS


CMDTABL. . . . . . . . . . . . .        L BYTE   023D    CSEG
CMOVEEX. . . . . . . . . . . . .        L NEAR   096C    CSEG
CNT. . . . . . . . . . . . . . .        L NEAR   08AE    CSEG
CNTC1B . . . . . . . . . . . . .        L NEAR   0974    CSEG
CNTFC. . . . . . . . . . . . . .        L NEAR   0888    CSEG
COLOR_TBL. . . . . . . . . . . .        L BYTE   0C3E    CSEG
COMTBL . . . . . . . . . . . . .        L BYTE   0267    CSEG
CONDEV . . . . . . . . . . . . .        L NEAR   0476    CSEG
CONOUT . . . . . . . . . . . . .        L NEAR   07C0    CSEG
CONTBL . . . . . . . . . . . . .        L NEAR   04D0    CSEG
CON_INT. . . . . . . . . . . . .        L NEAR   0547    CSEG
CON_WRI1 . . . . . . . . . . . .        L NEAR   05FE    CSEG
CON_WRIT . . . . . . . . . . . .        L NEAR   05FC    CSEG
CPB_ATTR . . . . . . . . . . . .        Number   0002
CPB_COL. . . . . . . . . . . . .        Number   0000
CPB_ESC. . . . . . . . . . . . .        Number   0003
CPB_FLEN . . . . . . . . . . . .        Number   0005
CPB_FREQ . . . . . . . . . . . .        Number   0004
CPB_RES1 . . . . . . . . . . . .        Number   0004
CPB_RES2 . . . . . . . . . . . .        Number   0005
CPB_ROW. . . . . . . . . . . . .        Number   0001
CPR_MESS . . . . . . . . . . . .        L NEAR   0CC2    CSEG
CRTACTTBL. . . . . . . . . . . .        L NEAR   0396    CSEG
CRTPARB. . . . . . . . . . . . .        L BYTE   07BA    CSEG
CRTPIM . . . . . . . . . . . . .        L NEAR   0E58    CSEG
CRTTBL . . . . . . . . . . . . .        L NEAR   0284    CSEG
CRT_EXIT . . . . . . . . . . . .        L NEAR   0EAF    CSEG
CRT_SB_FUNCT . . . . . . . . . .        L WORD   0003    CSEG
CRT_TR_TABLE . . . . . . . . . .        L WORD   0007    CSEG
CTABLEN. . . . . . . . . . . . .        L WORD   0265    CSEG
CUB. . . . . . . . . . . . . . .        L NEAR   0959    CSEG
CUD. . . . . . . . . . . . . . .        L NEAR   0951    CSEG
CUF. . . . . . . . . . . . . . .        L NEAR   0955    CSEG
CUP. . . . . . . . . . . . . . .        L NEAR   0868    CSEG
CUP1 . . . . . . . . . . . . . .        L NEAR   0966    CSEG
CURCOL . . . . . . . . . . . . .        L BYTE   0E4C    CSEG
CURD . . . . . . . . . . . . . .        Number   0DED
CUROFF . . . . . . . . . . . . .        L NEAR   10B6    CSEG
CURON. . . . . . . . . . . . . .        L NEAR   10C8    CSEG
CURPOS . . . . . . . . . . . . .        L NEAR   0958    CSEG
CURROW . . . . . . . . . . . . .        L BYTE   0E4D    CSEG
CURS . . . . . . . . . . . . . .        Number   0049
CUU. . . . . . . . . . . . . . .        L NEAR   094D    CSEG
DANSK. . . . . . . . . . . . . .        L NEAR   02FA    CSEG
DATRDY . . . . . . . . . . . . .        Number   0001
DCD. . . . . . . . . . . . . . .        Number   0040
DCHR . . . . . . . . . . . . . .        L NEAR   0AFE    CSEG
DEC_SIGN_1 . . . . . . . . . . .        L BYTE   0227    CSEG    Global
DEC_SIGN_2 . . . . . . . . . . .        L BYTE   0237    CSEG    Global
DEFFK. . . . . . . . . . . . . .        L NEAR   0C5D    CSEG
DEF_FUN. . . . . . . . . . . . .        L NEAR   0000            External
DELCHR . . . . . . . . . . . . .        L NEAR   1105    CSEG
DELCHR1. . . . . . . . . . . . .        L NEAR   110E    CSEG
DELLIN . . . . . . . . . . . . .        L NEAR   0AEE    CSEG
DEVSTART . . . . . . . . . . . .        L WORD   0476    CSEG    Global
DMAEXC . . . . . . . . . . . . .        Number   0010
DMAR . . . . . . . . . . . . . .        Number   00A4
DMAW . . . . . . . . . . . . . .        Number   0024
DO_ESC . . . . . . . . . . . . .        L NEAR   0E78    CSEG
DO_OUTCHAR . . . . . . . . . . .        L NEAR   0E90    CSEG
```

```
DREQ . . . . . . . . . . . . . . .        L BYTE  060F    CSEG
DRVMAX . . . . . . . . . . . . . .        L BYTE  04C8    CSEG    Global
DRWALL . . . . . . . . . . . . . .        Number  0000
DRWINP . . . . . . . . . . . . . .        Number  0008
DRWRET . . . . . . . . . . . . . .        Number  0010
DSKDEV . . . . . . . . . . . . . .        L NEAR  048E    CSEG
DSK_INT. . . . . . . . . . . . . .        L NEAR  0000            External
DSR. . . . . . . . . . . . . . . .        Number  0080
DYNRAM . . . . . . . . . . . . . .        Number  0004
ED . . . . . . . . . . . . . . . .        L NEAR  0B79    CSEG    Global
EL . . . . . . . . . . . . . . . .        L NEAR  0B8A    CSEG
ENTRY. . . . . . . . . . . . . . .        L NEAR  055D    CSEG
ERROR_0. . . . . . . . . . . . . .        L NEAR  059D    CSEG    Global
ERROR_1. . . . . . . . . . . . . .        L NEAR  0594    CSEG    Global
ERROR_10 . . . . . . . . . . . . .        L NEAR  05B8    CSEG    Global
ERROR_11 . . . . . . . . . . . . .        L NEAR  05BC    CSEG    Global
ERROR_12 . . . . . . . . . . . . .        L NEAR  05C0    CSEG    Global
ERROR_2. . . . . . . . . . . . . .        L NEAR  0598    CSEG    Global
ERROR_3. . . . . . . . . . . . . .        L NEAR  059C    CSEG    Global
ERROR_4. . . . . . . . . . . . . .        L NEAR  05A0    CSEG    Global
ERROR_5. . . . . . . . . . . . . .        L NEAR  05A4    CSEG    Global
ERROR_6. . . . . . . . . . . . . .        L NEAR  05A8    CSEG    Global
ERROR_7. . . . . . . . . . . . . .        L NEAR  05AC    CSEG    Global
ERROR_8. . . . . . . . . . . . . .        L NEAR  05B0    CSEG    Global
ERROR_9. . . . . . . . . . . . . .        L NEAR  05B4    CSEG    Global
ERR_EXIT . . . . . . . . . . . . .        L NEAR  05C2    CSEG    Global
ESC. . . . . . . . . . . . . . . .        Number  001B
ESCFAL . . . . . . . . . . . . . .        L NEAR  0886    CSEG
ESCNON . . . . . . . . . . . . . .        L NEAR  0879    CSEG
ESCNONA. . . . . . . . . . . . . .        L NEAR  086B    CSEG
ESCTBL . . . . . . . . . . . . . .        L BYTE  0287    CSEG
ESC_MASK . . . . . . . . . . . . .        Number  000F
ESC_TABLE. . . . . . . . . . . . .        L NEAR  0EEA    CSEG
ETBLENT. . . . . . . . . . . . . .        L WORD  0285    CSEG
EXIT . . . . . . . . . . . . . . .        L NEAR  05C7    CSEG    Global
EXIT1. . . . . . . . . . . . . . .        L NEAR  05C9    CSEG    Global
EXITP. . . . . . . . . . . . . . .        F PROC  05C7    CSEG    Length =0014
FIFEMP . . . . . . . . . . . . . .        Number  0004
FIFO . . . . . . . . . . . . . . .        Number  00A1
FIFULL . . . . . . . . . . . . . .        Number  0002
FIGD . . . . . . . . . . . . . . .        Number  006C
FIGS . . . . . . . . . . . . . . .        Number  004C
FK1. . . . . . . . . . . . . . . .        L NEAR  0C74    CSEG
FK2. . . . . . . . . . . . . . . .        L NEAR  0C87    CSEG
FLAG_BUF . . . . . . . . . . . . .        V BYTE  0000            External
FLOPPY_DRIVES. . . . . . . . . . .        L BYTE  0013    CSEG    Global
FLTAB. . . . . . . . . . . . . . .        L WORD  0011    CSEG    Global
FL_IN_RETRIES. . . . . . . . . . .        L BYTE  0015    CSEG    Global
FL_OUT_RETRIES . . . . . . . . . .        L BYTE  0014    CSEG    Global
FRAMING. . . . . . . . . . . . . .        Number  0020
FRANCE . . . . . . . . . . . . . .        L NEAR  02BF    CSEG
FUN1 . . . . . . . . . . . . . . .        L BYTE  001D    CSEG
FUN10. . . . . . . . . . . . . . .        L BYTE  0038    CSEG
FUN11. . . . . . . . . . . . . . .        L BYTE  003C    CSEG
FUN12. . . . . . . . . . . . . . .        L BYTE  0040    CSEG
FUN13. . . . . . . . . . . . . . .        L BYTE  0044    CSEG
FUN14. . . . . . . . . . . . . . .        L BYTE  0048    CSEG
FUN15. . . . . . . . . . . . . . .        L BYTE  004C    CSEG
FUN16. . . . . . . . . . . . . . .        L BYTE  0050    CSEG
FUN17. . . . . . . . . . . . . . .        L BYTE  0054    CSEG
```

IO.SYS

```
FUN18. . . . . . . . . . . . . . .      L BYTE  0058   CSEG
FUN19. . . . . . . . . . . . . . .      L BYTE  005C   CSEG
FUN2 . . . . . . . . . . . . . . .      L BYTE  0020   CSEG
FUN20. . . . . . . . . . . . . . .      L BYTE  0060   CSEG
FUN3 . . . . . . . . . . . . . . .      L BYTE  0023   CSEG
FUN4 . . . . . . . . . . . . . . .      L BYTE  0026   CSEG
FUN5 . . . . . . . . . . . . . . .      L BYTE  0029   CSEG
FUN6 . . . . . . . . . . . . . . .      L BYTE  002C   CSEG
FUN7 . . . . . . . . . . . . . . .      L BYTE  002F   CSEG
FUN8 . . . . . . . . . . . . . . .      L BYTE  0032   CSEG
FUN9 . . . . . . . . . . . . . . .      L BYTE  0035   CSEG
FUNCLEN. . . . . . . . . . . . . .      Number  0200
FUNCOFF. . . . . . . . . . . . . .      V WORD  0000          External
FUNCTBL. . . . . . . . . . . . . .      E BYTE  001D   CSEG   Global
FUNFILL. . . . . . . . . . . . . .      Number  01B9
GCHRD. . . . . . . . . . . . . . .      Number  0068
GDCCOM . . . . . . . . . . . . . .      Number  00A1
GDCPAR . . . . . . . . . . . . . .      Number  00A0
GDCRES . . . . . . . . . . . . . .      Number  0000
GDCSTA . . . . . . . . . . . . . .      Number  00A0
GERMANY. . . . . . . . . . . . . .      L NEAR  02D4   CSEG
GETONE . . . . . . . . . . . . . .      L NEAR  0935   CSEG
GETPARM. . . . . . . . . . . . . .      L NEAR  0942   CSEG
GETRET . . . . . . . . . . . . . .      L NEAR  093E   CSEG
GOTPARM. . . . . . . . . . . . . .      L NEAR  0946   CSEG
GR21 . . . . . . . . . . . . . . .      L NEAR  08E6   CSEG
GR22 . . . . . . . . . . . . . . .      L NEAR  08EC   CSEG
GRAEND . . . . . . . . . . . . . .      Number  1FFF
GRAMOD . . . . . . . . . . . . . .      Number  0002
H1 . . . . . . . . . . . . . . . .      L NEAR  0ECE   CSEG
H2 . . . . . . . . . . . . . . . .      L NEAR  0EE3   CSEG
HEBREW . . . . . . . . . . . . . .      L NEAR  0918   CSEG
HEB_ACTIVE . . . . . . . . . . . .      L NEAR  092A   CSEG
HEB_NOT. . . . . . . . . . . . . .      L NEAR  0934   CSEG
HIP_OUT. . . . . . . . . . . . . .      L NEAR  0EBA   CSEG
HORETR . . . . . . . . . . . . . .      Number  0040
HWINIT . . . . . . . . . . . . . .      L NEAR  0000          External
I29_HANDLER. . . . . . . . . . . .      L NEAR  050B   CSEG   Global
ICHR . . . . . . . . . . . . . . .      L NEAR  0AF6   CSEG
ICLEOL . . . . . . . . . . . . . .      L NEAR  1209   CSEG
ICLEOL_RET . . . . . . . . . . . .      L NEAR  1229   CSEG
IHOME. . . . . . . . . . . . . . .      L NEAR  1250   CSEG
ILF. . . . . . . . . . . . . . . .      L NEAR  125A   CSEG
ILF1 . . . . . . . . . . . . . . .      L NEAR  126A   CSEG
INIT . . . . . . . . . . . . . . .      L NEAR  0000   CSEG
INIT10 . . . . . . . . . . . . . .      L NEAR  10F4   CSEG
INLRPF . . . . . . . . . . . . . .      Number  0008
INPAR. . . . . . . . . . . . . . .      L NEAR  0FBF   CSEG
INSCH1 . . . . . . . . . . . . . .      L NEAR  11AA   CSEG
INSCHR . . . . . . . . . . . . . .      L NEAR  1199   CSEG
INSLIN . . . . . . . . . . . . . .      L NEAR  0AE6   CSEG
INTLAC . . . . . . . . . . . . . .      Number  0009
INTTYPE. . . . . . . . . . . . . .      L BYTE  0609   CSEG
INT_TRAP . . . . . . . . . . . . .      L NEAR  05F8   CSEG   Global
ITALY. . . . . . . . . . . . . . .      L NEAR  031A   CSEG
KBD_OUT. . . . . . . . . . . . . .      L NEAR  0000          External
KBD_TT . . . . . . . . . . . . . .      L BYTE  021D   CSEG   Global
KEYBFUNC_TBL . . . . . . . . . . .      L WORD  0000   CSEG
KEYBOARD_TBL . . . . . . . . . . .      L WORD  000B   CSEG
KEY_IN . . . . . . . . . . . . . .      L NEAR  0000          External
```

```
KEY_INIT . . . . . . . . . . . .        L NEAR  0000            External
KEY_IN_FL. . . . . . . . . . . .        L NEAR  0000            External
KEY_NO_IN. . . . . . . . . . . .        L NEAR  0000            External
KEY_ST . . . . . . . . . . . . .        L NEAR  0000            External
KSPAIN . . . . . . . . . . . . .        L NEAR  030D    CSEG
LANGUAGE . . . . . . . . . . . .        V BYTE  0000            External
LANT . . . . . . . . . . . . . .        L NEAR  08F2    CSEG
LASTPRM. . . . . . . . . . . . .        L BYTE  0719    CSEG
LEN1 . . . . . . . . . . . . . .        Number  0003
LEN10. . . . . . . . . . . . . .        Number  0004
LEN11. . . . . . . . . . . . . .        Number  0004
LEN12. . . . . . . . . . . . . .        Number  0004
LEN13. . . . . . . . . . . . . .        Number  0004
LEN14. . . . . . . . . . . . . .        Number  0004
LEN15. . . . . . . . . . . . . .        Number  0004
LEN16. . . . . . . . . . . . . .        Number  0004
LEN17. . . . . . . . . . . . . .        Number  0004
LEN18. . . . . . . . . . . . . .        Number  0004
LEN19. . . . . . . . . . . . . .        Number  0004
LEN2 . . . . . . . . . . . . . .        Number  0003
LEN20. . . . . . . . . . . . . .        Number  0004
LEN3 . . . . . . . . . . . . . .        Number  0003
LEN4 . . . . . . . . . . . . . .        Number  0003
LEN5 . . . . . . . . . . . . . .        Number  0003
LEN6 . . . . . . . . . . . . . .        Number  0003
LEN7 . . . . . . . . . . . . . .        Number  0003
LEN8 . . . . . . . . . . . . . .        Number  0003
LEN9 . . . . . . . . . . . . . .        Number  0003
LINBUF . . . . . . . . . . . . .        L BYTE  071A    CSEG    Length =00A0
LINEFD . . . . . . . . . . . . .        L NEAR  09BB    CSEG
LIPDET . . . . . . . . . . . . .        Number  0080
LP11 . . . . . . . . . . . . . .        L BYTE  0E52    CSEG
LP12 . . . . . . . . . . . . . .        L BYTE  0E53    CSEG
LP21 . . . . . . . . . . . . . .        L BYTE  0E56    CSEG
LP22 . . . . . . . . . . . . . .        L BYTE  0E57    CSEG
LPC. . . . . . . . . . . . . . .        Number  000A
LPOISET. . . . . . . . . . . . .        L NEAR  08FE    CSEG
LPRD . . . . . . . . . . . . . .        Number  00CD
LT10 . . . . . . . . . . . . . .        L NEAR  0CBB    CSEG
LVAR0. . . . . . . . . . . . . .        L NEAR  02B4    CSEG
LVAR1. . . . . . . . . . . . . .        L NEAR  02B7    CSEG
LVAR10 . . . . . . . . . . . . .        L NEAR  0348    CSEG
LVAR11 . . . . . . . . . . . . .        L NEAR  0357    CSEG
LVAR12 . . . . . . . . . . . . .        L NEAR  0368    CSEG
LVAR13 . . . . . . . . . . . . .        L NEAR  0379    CSEG
LVAR2. . . . . . . . . . . . . .        L NEAR  02BE    CSEG
LVAR3. . . . . . . . . . . . . .        L NEAR  02D3    CSEG
LVAR4. . . . . . . . . . . . . .        L NEAR  02E6    CSEG
LVAR5. . . . . . . . . . . . . .        L NEAR  02F9    CSEG
LVAR6. . . . . . . . . . . . . .        L NEAR  030C    CSEG
LVAR7. . . . . . . . . . . . . .        L NEAR  0319    CSEG
LVAR8. . . . . . . . . . . . . .        L NEAR  0330    CSEG
LVAR9. . . . . . . . . . . . . .        L NEAR  0345    CSEG
M1RS232. . . . . . . . . . . . .        L BYTE  001A    CSEG
M2RS232. . . . . . . . . . . . .        L BYTE  001B    CSEG
MASKREG. . . . . . . . . . . . .        Number  004A
MDROW. . . . . . . . . . . . . .        L NEAR  1341    CSEG
MIXGAC . . . . . . . . . . . . .        Number  0000
MOCOMP . . . . . . . . . . . . .        Number  0001
MONO_COLOR . . . . . . . . . . .        L BYTE  0600    CSEG    Global
```

IO.SYS

```
MON_ATT. . . . . . . . . . . . .  L BYTE  060C    CSEG
MOREPL . . . . . . . . . . . . .  Number  0000
MORES. . . . . . . . . . . . . .  Number  0002
MOSET. . . . . . . . . . . . . .  Number  0003
MUROW. . . . . . . . . . . . . .  L NEAR  1323    CSEG
MUSIC. . . . . . . . . . . . . .  L NEAR  1187    CSEG
NOFS . . . . . . . . . . . . . .  L NEAR  09CB    CSEG
NOFS1. . . . . . . . . . . . . .  L NEAR  09D6    CSEG
NOINTL . . . . . . . . . . . . .  Number  0000
NON_ANSI_ESC . . . . . . . . . .  L WORD  0009    CSEG
NOTI . . . . . . . . . . . . . .  L NEAR  08D8    CSEG
NOTII. . . . . . . . . . . . . .  L NEAR  08D3    CSEG
NO_OP. . . . . . . . . . . . . .  L NEAR  0F0A    CSEG
NRWAPL . . . . . . . . . . . . .  Number  0048
01 . . . . . . . . . . . . . . .  L NEAR  0E9A    CSEG
OUTCHAR. . . . . . . . . . . . .  L BYTE  0E4F    CSEG
OUTCTR . . . . . . . . . . . . .  L NEAR  0897    CSEG   Global
OVERRUN. . . . . . . . . . . . .  Number  0010
P1CHROUT . . . . . . . . . . . .  L NEAR  0E2C    CSEG
P1STA1 . . . . . . . . . . . . .  L NEAR  0E40    CSEG
P1STATUS . . . . . . . . . . . .  L NEAR  0E36    CSEG
P1STATX. . . . . . . . . . . . .  L NEAR  0E49    CSEG
PACTIVE. . . . . . . . . . . . .  L BYTE  0081    CSEG
PARITY . . . . . . . . . . . . .  Number  0008
PARMS. . . . . . . . . . . . . .  L BYTE  0619    CSEG   Length =0100
PASSCH . . . . . . . . . . . . .  L NEAR  0892    CSEG
PBCOM. . . . . . . . . . . . . .  Number  0063
PBDA . . . . . . . . . . . . . .  Number  0060
PBSTA. . . . . . . . . . . . . .  Nunber  0061
PINIT. . . . . . . . . . . . . .  L NEAR  0E1D    CSEG
PITCH. . . . . . . . . . . . . .  Number  0047
PLAY_MUSIC . . . . . . . . . . .  L NEAR  0CCB    CSEG
PM_FREQ. . . . . . . . . . . . .  L NEAR  0CDC    CSEG
PM_TLENGTH . . . . . . . . . . .  L NEAR  0CE6    CSEG
POBF . . . . . . . . . . . . . .  Number  0002
PORTUG . . . . . . . . . . . . .  L NEAR  0369    CSEG
POS1 . . . . . . . . . . . . . .  L NEAR  0B1B    CSEG
POS2 . . . . . . . . . . . . . .  L NEAR  0B28    CSEG
POSC1. . . . . . . . . . . . . .  L NEAR  0B12    CSEG
POSC2. . . . . . . . . . . . . .  L NEAR  0B22    CSEG
POSCUR . . . . . . . . . . . . .  Number  0008
POSIT. . . . . . . . . . . . . .  L NEAR  0B06    CSEG
PRAM . . . . . . . . . . . . . .  Number  0070
PRAMSA . . . . . . . . . . . . .  Number  0000
PRCP . . . . . . . . . . . . . .  L NEAR  0C57    CSEG
PRINTER_IF_TYPE. . . . . . . . .  L BYTE  0019    CSEG
PRINTER_OUT. . . . . . . . . . .  L WORD  0D2F    CSEG
PRMPNT . . . . . . . . . . . . .  L WORD  0617    CSEG
PRNDEV . . . . . . . . . . . . .  L NEAR  049A    CSEG
PRNSER_INIT. . . . . . . . . . .  L NEAR  000F    CSEG
PRNTBL . . . . . . . . . . . . .  L NEAR  051E    CSEG
PRN_INIT . . . . . . . . . . . .  L NEAR  0D02    CSEG
PRN_INT. . . . . . . . . . . . .  L NEAR  0553    CSEG
PRN_RDY_ST . . . . . . . . . . .  L NEAR  0D24    CSEG
PRN_SERST. . . . . . . . . . . .  L NEAR  0D27    CSEG
PRN_STA. . . . . . . . . . . . .  L NEAR  0D15    CSEG
PRN_WR1. . . . . . . . . . . . .  L NEAR  0D41    CSEG
PRN_WRT. . . . . . . . . . . . .  L NEAR  0D31    CSEG
PRN_WRTL . . . . . . . . . . . .  L NEAR  0D47    CSEG
PRN_WRX. . . . . . . . . . . . .  L NEAR  0D49    CSEG
```

```
PSCP . . . . . . . . . . . . . .        L NEAR  0C50    CSEG
PTRSAV . . . . . . . . . . . . .        L DWORD 0538    CSEG    Global
PVRS232. . . . . . . . . . . . .        L BYTE  001C    CSEG
RADDR. . . . . . . . . . . . . .        L WORD  060A    CSEG
RBLINK . . . . . . . . . . . . .        L NEAR  0AD6    CSEG
RDAT . . . . . . . . . . . . . .        Number  00A0
RDGCHR . . . . . . . . . . . . .        L NEAR  0F2A    CSEG
RDLIN. . . . . . . . . . . . . .        L NEAR  102F    CSEG
RDLIN1 . . . . . . . . . . . . .        L NEAR  1067    CSEG
REAC_HI. . . . . . . . . . . . .        L NEAR  09F6    CSEG
REMNUM . . . . . . . . . . . . .        V WORD  0000            External
RESMOP . . . . . . . . . . . . .        Number  0000
RETST3 . . . . . . . . . . . . .        L NEAR  0816    CSEG
RETSTAT1 . . . . . . . . . . . .        L NEAR  0CF1    CSEG
REVERSE. . . . . . . . . . . . .        L NEAR  0B38    CSEG
REV_COLOR. . . . . . . . . . . .        L NEAR  0AAA    CSEG
RE_INIT. . . . . . . . . . . . .        L NEAR  0546    CSEG    Global
RHALF1 . . . . . . . . . . . . .        L NEAR  0A45    CSEG
RHALFC . . . . . . . . . . . . .        L NEAR  0A4D    CSEG
RHALF_INT. . . . . . . . . . . .        L NEAR  0A32    CSEG
RINV_VID1. . . . . . . . . . . .        L NEAR  0AA2    CSEG
RINV_VIDEO . . . . . . . . . . .        L NEAR  0A89    CSEG
RLF. . . . . . . . . . . . . . .        L NEAR  09C2    CSEG
ROWS . . . . . . . . . . . . . .        Number  0018
RV1. . . . . . . . . . . . . . .        L NEAR  0B50    CSEG
RV2. . . . . . . . . . . . . . .        L NEAR  0B59    CSEG
RVE. . . . . . . . . . . . . . .        L NEAR  0B60    CSEG
RVO. . . . . . . . . . . . . . .        L NEAR  0B44    CSEG
RXRDY. . . . . . . . . . . . . .        Number  0002
SACTIVE. . . . . . . . . . . . .        L BYTE  0D80    CSEG
SAFRICA. . . . . . . . . . . . .        L NEAR  0358    CSEG
SATRM. . . . . . . . . . . . . .        Number  0000
SAVCRTPARB . . . . . . . . . . .        L WORD  0610    CSEG
SBLINK . . . . . . . . . . . . .        L NEAR  0ACE    CSEG
SCLDN1 . . . . . . . . . . . . .        L NEAR  130A    CSEG
SCLDN2 . . . . . . . . . . . . .        L NEAR  1316    CSEG
SCLUP1 . . . . . . . . . . . . .        L NEAR  12B3    CSEG
SCLUP2 . . . . . . . . . . . . .        L NEAR  12BF    CSEG
SCLUP3 . . . . . . . . . . . . .        L NEAR  12D3    CSEG
SCLUP4 . . . . . . . . . . . . .        L NEAR  12CD    CSEG
SCROL1 . . . . . . . . . . . . .        L NEAR  113C    CSEG
SCROL2 . . . . . . . . . . . . .        L NEAR  1138    CSEG
SCROLLDN . . . . . . . . . . . .        L NEAR  12FB    CSEG
SCROLLUP . . . . . . . . . . . .        L NEAR  129F    CSEG
SCROLLX. . . . . . . . . . . . .        L NEAR  1119    CSEG
SCWID. . . . . . . . . . . . . .        Number  0050
SECREQ . . . . . . . . . . . . .        L NEAR  091E    CSEG
SENPAR . . . . . . . . . . . . .        L NEAR  0FC8    CSEG
SESC . . . . . . . . . . . . . .        L NEAR  0B7E    CSEG
SETACTTBL. . . . . . . . . . . .        L NEAR  0907    CSEG
SETCUR . . . . . . . . . . . . .        L NEAR  0FD6    CSEG
SETCUR1. . . . . . . . . . . . .        L NEAR  0FE4    CSEG
SETMSK . . . . . . . . . . . . .        L NEAR  1010    CSEG
SGEXIT . . . . . . . . . . . . .        L NEAR  0C37    CSEG
SGR. . . . . . . . . . . . . . .        L NEAR  0B91    CSEG
SGR1 . . . . . . . . . . . . . .        L NEAR  0BA5    CSEG
SGR2 . . . . . . . . . . . . . .        L NEAR  0BAC    CSEG
SGR3 . . . . . . . . . . . . . .        L NEAR  0BB3    CSEG
SGR4 . . . . . . . . . . . . . .        L NEAR  0BBA    CSEG
SGR5 . . . . . . . . . . . . . .        L NEAR  0BC1    CSEG
```

```
IO.SYS

SGRBG. . . . . . . . . . . . . .          L NEAR  0BE6    CSEG
SGRC . . . . . . . . . . . . . .          L NEAR  0BC8    CSEG
SGRC1. . . . . . . . . . . . . .          L NEAR  0BEB    CSEG
SGRCDO . . . . . . . . . . . . .          L NEAR  0C0A    CSEG
SGRCEX . . . . . . . . . . . . .          L NEAR  0C25    CSEG
SGRENDE. . . . . . . . . . . . .          L NEAR  0C3D    CSEG
SGRINV . . . . . . . . . . . . .          L NEAR  0C01    CSEG
SGRSFG . . . . . . . . . . . . .          L NEAR  0C1C    CSEG
SGRX . . . . . . . . . . . . . .          L NEAR  0B95    CSEG
SHALF1 . . . . . . . . . . . . .          L NEAR  0A0F    CSEG
SHALF2 . . . . . . . . . . . . .          L NEAR  0A07    CSEG
SHALFBG. . . . . . . . . . . . .          L NEAR  0A17    CSEG
SHALF_INT. . . . . . . . . . . .          L NEAR  09E3    CSEG
SIF_DISP . . . . . . . . . . . .          L NEAR  0D86    CSEG
SINV_VID1. . . . . . . . . . . .          L NEAR  0A81    CSEG
SINV_VIDEO . . . . . . . . . . .          L NEAR  0A68    CSEG
SIOINIT. . . . . . . . . . . . .          L NEAR  0E04    CSEG
SO_DISP_TBL. . . . . . . . . . .          L NEAR  0D90    CSEG
SP1. . . . . . . . . . . . . . .          L WORD  0E50    CSEG
SP2. . . . . . . . . . . . . . .          L WORD  0E54    CSEG
SPA1 . . . . . . . . . . . . . .          L NEAR  0DD4    CSEG
SPA2 . . . . . . . . . . . . . .          L NEAR  0DDF    CSEG
SPA3 . . . . . . . . . . . . . .          L NEAR  0DEF    CSEG
SPA4 . . . . . . . . . . . . . .          L NEAR  0DEE    CSEG
SPAI1. . . . . . . . . . . . . .          L NEAR  0DAF    CSEG
SPAI2. . . . . . . . . . . . . .          L NEAR  0DB8    CSEG
SPAI3. . . . . . . . . . . . . .          L NEAR  0DC0    CSEG
SPAIN. . . . . . . . . . . . . .          L NEAR  0DF2    CSEG
SPAIST . . . . . . . . . . . . .          L NEAR  0DA5    CSEG
SPAOST . . . . . . . . . . . . .          L NEAR  0DCA    CSEG
SPAOUT . . . . . . . . . . . . .          L NEAR  0DFA    CSEG
SPCLEAR. . . . . . . . . . . . .          L NEAR  1293    CSEG
SPCLEAR1 . . . . . . . . . . . .          L NEAR  DF5D    CSEG
SPCLEAR2 . . . . . . . . . . . .          L NEAR  0F73    CSEG
SPRCOM . . . . . . . . . . . . .          Number  0063
SPRDATA. . . . . . . . . . . . .          Number  0060
SPRSTAT. . . . . . . . . . . . .          Number  0061
SPWCOM . . . . . . . . . . . . .          Number  0067
SPWDATA. . . . . . . . . . . . .          Number  0064
SPWMODE. . . . . . . . . . . . .          Number  0066
SRLOUT . . . . . . . . . . . . .          L NEAR  0D83    CSEG
SRLSTAT. . . . . . . . . . . . .          L NEAR  0DA0    CSEG
SST_DISP_TBL . . . . . . . . . .          L NEAR  0D98    CSEG
ST1. . . . . . . . . . . . . . .          L NEAR  07C5    CSEG
ST12 . . . . . . . . . . . . . .          L NEAR  07CC    CSEG
ST2. . . . . . . . . . . . . . .          L NEAR  07D1    CSEG
ST21 . . . . . . . . . . . . . .          L NEAR  07D8    CSEG
ST3. . . . . . . . . . . . . . .          L NEAR  07E8    CSEG
ST31 . . . . . . . . . . . . . .          L NEAR  07F3    CSEG
ST3A . . . . . . . . . . . . . .          L NEAR  081E    CSEG
ST3B . . . . . . . . . . . . . .          L NEAR  083C    CSEG
ST3B1. . . . . . . . . . . . . .          L NEAR  0859    CSEG
ST3B2. . . . . . . . . . . . . .          L NEAR  0864    CSEG
ST3C . . . . . . . . . . . . . .          L NEAR  0806    CSEG
ST3D . . . . . . . . . . . . . .          L NEAR  0847    CSEG
ST3RET . . . . . . . . . . . . .          L NEAR  0805    CSEG
START. . . . . . . . . . . . . .          Number  006B
STATE. . . . . . . . . . . . . .          L WORD  0615    CSEG
STRATEGY . . . . . . . . . . . .          L NEAR  053C    CSEG
STRATP . . . . . . . . . . . . .          F PROC  053C    CSEG    Length =000B
```

```
STRINGF. . . . . . . . . . . . .        L BYTE  0612     CSEG
SWEDEN . . . . . . . . . . . . .        L NEAR  02E7     CSEG
SWIDTH . . . . . . . . . . . . .        Number  0050
SWISS12. . . . . . . . . . . . .        L NEAR  0331     CSEG
S_STAT1. . . . . . . . . . . . .        L NEAR  0A2B     CSEG
TEST_POS . . . . . . . . . . . .        L NEAR  11FB     CSEG
TEST_VID_OUT . . . . . . . . . .        L NEAR  0E8A     CSEG
THEX1. . . . . . . . . . . . . .        L NEAR  0CB2     CSEG
THEX_ASCII . . . . . . . . . . .        L NEAR  0CAE     CSEG
TIMDEV . . . . . . . . . . . . .        L NEAR  04AC     CSEG
TIMTBL . . . . . . . . . . . . .        L NEAR  0504     CSEG
TIM_DAYS . . . . . . . . . . . .        L NEAR  0D58     CSEG
TIM_HRS. . . . . . . . . . . . .        L NEAR  0D5B     CSEG
TIM_HSEC . . . . . . . . . . . .        L NEAR  0D5C     CSEG
TIM_INT. . . . . . . . . . . . .        L NEAR  0559     CSEG
TIM_MINS . . . . . . . . . . . .        L NEAR  0D5A     CSEG
TIM_RED. . . . . . . . . . . . .        L NEAR  0D73     CSEG
TIM_SECS . . . . . . . . . . . .        L NEAR  0D5D     CSEG
TIM_WRT. . . . . . . . . . . . .        L NEAR  0D5E     CSEG
TRERR. . . . . . . . . . . . . .        L NEAR  0DC1     CSEG
TXEMT. . . . . . . . . . . . . .        Number  0004
TXRDY. . . . . . . . . . . . . .        Number  0001
TYHIBY . . . . . . . . . . . . .        Number  0018
TYLOBY . . . . . . . . . . . . .        Number  0010
TYWORD . . . . . . . . . . . . .        Number  0000
UK . . . . . . . . . . . . . . .        L NEAR  02B8     CSEG
US . . . . . . . . . . . . . . .        L NEAR  02B5     CSEG
VAR0L. . . . . . . . . . . . . .        Number  0003
VAR10L . . . . . . . . . . . . .        Number  000F
VAR11L . . . . . . . . . . . . .        Number  0011
VAR12L . . . . . . . . . . . . .        Number  0011
VAR13L . . . . . . . . . . . . .        Number  0015
VAR1L. . . . . . . . . . . . . .        Number  0007
VAR2L. . . . . . . . . . . . . .        Number  0015
VAR3L. . . . . . . . . . . . . .        Number  0013
VAR4L. . . . . . . . . . . . . .        Number  0013
VAR5L. . . . . . . . . . . . . .        Number  0013
VAR6L. . . . . . . . . . . . . .        Number  000D
VAR7L. . . . . . . . . . . . . .        Number  0017
VAR8L. . . . . . . . . . . . . .        Number  0015
VAR9L. . . . . . . . . . . . . .        Number  0003
VCLEAR . . . . . . . . . . . . .        L NEAR  126D     CSEG
VERETR . . . . . . . . . . . . .        Number  0020
VHOME. . . . . . . . . . . . . .        L NEAR  09DF     CSEG
VSYNCM . . . . . . . . . . . . .        Number  006F
VSYNCS . . . . . . . . . . . . .        Number  006E
WDAT . . . . . . . . . . . . . .        Number  0020
WINCH_DRIVES . . . . . . . . . .        L BYTE  0016     CSEG
WI_IN_RETRIES. . . . . . . . . .        L BYTE  0018     CSEG
WI_OUT_RETRIES . . . . . . . . .        L BYTE  0017     CSEG
WPL. . . . . . . . . . . . . . .        Number  0024
WRGCHR . . . . . . . . . . . . .        L NEAR  0F08     CSEG
WRHLPOS. . . . . . . . . . . . .        L NEAR  117C     CSEG
WRITEPOS . . . . . . . . . . . .        L NEAR  1171     CSEG
WRLIN. . . . . . . . . . . . . .        L NEAR  1070     CSEG
WRLIN1 . . . . . . . . . . . . .        L NEAR  10A8     CSEG
XDSR . . . . . . . . . . . . . .        L NEAR  0C88     CSEG
XOFF . . . . . . . . . . . . . .        Number  0013
XOFFFLG. . . . . . . . . . . . .        L BYTE  0D82     CSEG
XON. . . . . . . . . . . . . . .        Number  0011
```

```
IO.SYS


XX1. . . . . . . . . . . . . . . .      L NEAR  0F0B    CSEG
XX10 . . . . . . . . . . . . . . .      L NEAR  1070    CSEG
XX11 . . . . . . . . . . . . . . .      L NEAR  1098    CSEG
XX12 . . . . . . . . . . . . . . .      L NEAR  10B6    CSEG
XX13 . . . . . . . . . . . . . . .      L NEAR  10CB    CSEG
XX14 . . . . . . . . . . . . . . .      L NEAR  1104    CSEG
XX15 . . . . . . . . . . . . . . .      L NEAR  113C    CSEG
XX16 . . . . . . . . . . . . . . .      L NEAR  0F15    CSEG
XX17 . . . . . . . . . . . . . . .      L NEAR  0F1F    CSEG
XX18 . . . . . . . . . . . . . . .      L NEAR  0F34    CSEG
XX19 . . . . . . . . . . . . . . .      L NEAR  0F3E    CSEG
XX2. . . . . . . . . . . . . . . .      L NEAR  0F2A    CSEG
XX20 . . . . . . . . . . . . . . .      L NEAR  0F81    CSEG
XX21 . . . . . . . . . . . . . . .      L NEAR  0F8B    CSEG
XX22 . . . . . . . . . . . . . . .      L NEAR  0F95    CSEG
XX23 . . . . . . . . . . . . . . .      L NEAR  0FA9    CSEG
XX24 . . . . . . . . . . . . . . .      L NEAR  0FB3    CSEG
XX25 . . . . . . . . . . . . . . .      L NEAR  0FC8    CSEG
XX26 . . . . . . . . . . . . . . .      L NEAR  0FEE    CSEG
XX27 . . . . . . . . . . . . . . .      L NEAR  0FF8    CSEG
XX28 . . . . . . . . . . . . . . .      L NEAR  1002    CSEG
XX29 . . . . . . . . . . . . . . .      L NEAR  101A    CSEG
XX3. . . . . . . . . . . . . . . .      L NEAR  0F48    CSEG
XX30 . . . . . . . . . . . . . . .      L NEAR  1024    CSEG
XX31 . . . . . . . . . . . . . . .      L NEAR  1039    CSEG
XX32 . . . . . . . . . . . . . . .      L NEAR  1043    CSEG
XX33 . . . . . . . . . . . . . . .      L NEAR  104D    CSEG
XX34 . . . . . . . . . . . . . . .      L NEAR  107A    CSEG
XX35 . . . . . . . . . . . . . . .      L NEAR  1084    CSEG
XX36 . . . . . . . . . . . . . . .      L NEAR  108E    CSEG
XX37 . . . . . . . . . . . . . . .      L NEAR  10A8    CSEG
XX38 . . . . . . . . . . . . . . .      L NEAR  10C0    CSEG
XX39 . . . . . . . . . . . . . . .      L NEAR  10D5    CSEG
XX4. . . . . . . . . . . . . . . .      L NEAR  0F77    CSEG
XX40 . . . . . . . . . . . . . . .      L NEAR  10DF    CSEG
XX41 . . . . . . . . . . . . . . .      L NEAR  10E9    CSEG
XX42 . . . . . . . . . . . . . . .      L NEAR  110E    CSEG
XX5. . . . . . . . . . . . . . . .      L NEAR  0F9F    CSEG
XX6. . . . . . . . . . . . . . . .      L NEAR  0FE4    CSEG
XX7. . . . . . . . . . . . . . . .      L NEAR  1010    CSEG
XX8. . . . . . . . . . . . . . . .      L NEAR  102F    CSEG
XX9. . . . . . . . . . . . . . . .      L NEAR  1057    CSEG
YUGOSL . . . . . . . . . . . . . .      L NEAR  037A    CSEG
ZOOM . . . . . . . . . . . . . . .      Number  0046
```

```
                         ;
                         ;       ************************************************
                         ;       **                                            **
                         ;       **              K E Y B O A R D               **
                         ;       **                                            **
                         ;       **                D R I V E R                 **
                         ;       **                                            **
                         ;       ************************************************


                         public  def_fun,kbd_out         ; callable procedures
                         public  key_init,key_in,key_nd_in,key_st        ; jump-labels
                         public  key_in_fl                               ;    "
                         public  language                                ; defined byte
                         public  flag_buf,remnum,funcoff
                         ;
                         ;
                         EXTRN   ERROR_1:NEAR,ERROR_2:NEAR,ERROR_3:NEAR  ; ERROR-EXITS TO JUMP TO
                         EXTRN   ERROR_4:NEAR,ERROR_5:NEAR,ERROR_6:NEAR  ;       "       "
                         EXTRN   ERROR_7:NEAR,ERROR_8:NEAR,ERROR_9:NEAR  ;       "       "
                         EXTRN   ERROR_10:NEAR,ERROR_11:NEAR,ERROR_12:NEAR;      "       "
                         EXTRN   ERROR_0:NEAR,EXIT:NEAR,EXIT1:NEAR       ;       "       "
                         EXTRN   BUS_EXIT:NEAR                           ;       "       "
                         EXTRN   PTRSAV:DWORD                            ; ADDRESS OF DATA BLOCK
                         EXTRN   kbd_tt:BYTE,functbl:BYTE                ; tables
                         EXTRN   clear_1:BYTE,clear_2:BYTE               ; kbd_tt: for clear key
                         EXTRN   dec_sign_1:BYTE,dec_sign_2:BYTE         ; kbd_tt; for numeric decimal sign
                         ;
0000                     CSEG    segment public 'CODE'
                                 assume ds:CSEG,cs:CSEG,ss:CSEG
                         ;
                         ;
                         ;
                         ;
                         ;
= 0001                                   no_of_units     equ     1               ; number of units using this driver
                         ;
                         ;
                         ;
= 0200                                   funclen equ     512             ; max. length of function table
                         ;
                         ;
                         ;
                         ;
                         ;
```

KBD-DRV

```
                    C ;     *********************
                    C ;     *  keyboard equates *
                    C ;     *********************
                    C ;
= 0040              C       keybase     equ   40h          ; no of controller
=                   C       wdkey       equ   keybase      ; output to keyboard
=                   C       rdkey       equ   keybase      ; input from keyboard
= 0041              C       rskey       equ   keybase+1    ; status addr of keyboard
= 0041              C       kbell       equ   keybase+1    ; addr for output a bell
= 0041              C       kcount      equ   keybase+1    ; kbd output of language number
                    C ;
                    C ;
                    C ;
= 0001              C       country     equ   01h          ; command to get country code
                    C ;
                    C ;
                    C ;
                    C ;
                    C ;
= 0080              C       lgdat86     equ   80h    ; flag for language byte ready
= 0002              C       inpbuff86   equ   02h    ; flag for output to kbd full
= 0001              C       kbdat86     equ   01h    ; flag for input from kbd ready
                    C ;
                    C ;
                    C ;
0000  00            C       language    db    00h    ; language code :
                    C                                ; OLD KBD      NEW KBD I     NEW KBD II
                    C                                ; 00 U.S.      10 U.S.       20 SWITZERLAND 1
                    C                                ; 01 U.K.      11 U.K.       21 SWITZERLAND 2
                    C                                ; 02 FRANCE    12 DENMARK    22 FRANCE
                    C                                ; 03 GERMANY   13 GERMANY    23 CANADA
                    C                                ; 04 SWED/FIN  14 SWED/FIN   24 SOUTH AFRICA
                    C                                ; 05 NORW/DENM 15 NORWAY     25 PORTUGAL
                    C                                ; 06 SPAIN     16 SPAIN      26 BRAZIL
                    C                                ; 07 ITALY     17 ITALY      27 YUGOSLAVIA
0001  F8            C       kbd_var     db    0f8h   ; variante of keyboard
```

```
0002  ??                        flag_buf      db     ?              ; keyboard input flags
                                                                    ; detailed seen at cseg:
                                                                    ; operation status record
0003  ??                        in_buffer     db     ?              ; stores data goten from keyboard
0004  0000 E                    funcoff       dw     offset functbl ; offset to function table
0006  0000                      remnum        dw     00h            ; number of remainding bytes of function
                                                                    ; that could not be transfered compleltly
                         ;
                         ;
                         ;
                         ;;     ***************************************
                         ;      **                                   **
                         ;      **      character devices            **
                         ;      **                                   **
                         ;      *       needed for  M S - D O S      **
                         ;      **                                   **
                         ;      ***************************************
                         ;
                         ;
                         ;
                                special_dev_header       struc                ; special device header is the iden
                    tifier of a driver file
0000  ????????                          point_nxt    dd       ?              ; point to next device
0004  ????                              attribute    dw       ?              ; attributes for identifying the dr
                    iver
0006  ????                              strat_entry  dw       ?              ; offset to strategy routine
0008  ????                              inter_entry  dw       ?              ; offset to interrupt routine
000A    08 [                            dev_name     db       8 dup (?)      ; name of device driver
             ??
               ]

0012                                    special_dev_header    ends
                         ;
                         ;
                         ;
                                static_request_header    struc                ; interface to D O S
0000  ??                                long         db       ?              ; length of whole data block (incl.
                    this header)
0001  ??                                subunit      db       ?              ; subunit of this driver
0002  ??                                command      db       ?              ; command code of operation
0003  ????                              status       dw       ?              ; status of operation
0005  ????????                          dos_que_addr dd       ?              ; address of D O S queue
0009  ????????                          dev_que_addr dd       ?              ; address of driver queue
000D                                    static_request_header ends
                         ;
                         ;
                         ;
                                read_write_data          struc                ; data block for read and write com
                    mands
0000    0D [                                         db       13 dup (?)     ; static request header
             ??
               ]

000D  ??                                media_desc   db       ?              ;
000E  ????                              transadd_off dw       ?              ; address of data to be transfered
0010  ????                              transadd_seg dw       ?              ; divided in segment and offset
0012  ????                              s_b_count    dw       ?              ; counter of bytes of transfer
0014                                    read_write_data ends
```

```
                                   address_struc         struc
0000 ????                                 off     dw     ?
0002 ????                                 seg     dw     ?
0004                               address_struc   ends
                            ;
                            ;
                            ;
                            ;
                            ;
                                   non_desr_read         struc                    ; non-destructive input no wait
0000    0D [                                             db     13 dup (?)        ; static request header
             ??
                 ]

0000 ??                                   read_byte      db     ?                 ; last read byte
000E                                      non_desr_read  ends
                            ;
                            ;
                            ;
                                   init_struc            struc                    ; data block for initialization
0000    0D [                                             db     13 dup (?)        ; static request header
             ??
                 ]

0000 ??                                   unit_no        db     ?                 ; number of unit to initialize
000E ????                                 breakadd_off   dw     ?                 ; end of strategy routine
0010 ????                                 breakadd_seg   dw     ?                 ; with full address: segment+offset
0012                                      init_struc     ends
                            ;
```

```
                          ;
                          ;
                          ;
                          ;       error code equates :
                          ;
= 8000                            write_protect   equ     8000h           ; write protected violation
= 8001                            unknow_unit     equ     8001h           ; unknown unit
= 8002                            drive_n_ready   equ     8002h           ; drive not ready
= 8003                            unknow_comm     equ     8003h           ; unknown command
= 8004                            crc_error       equ     8004h           ; crc error
= 8005                            bad_srh_length  equ     8005h           ; bad drive request structure lengt
                  h
= 8006                            seek_error      equ     8006h           ; seek error
= 8007                            unknow_media    equ     8007h           ; unknown media
= 8008                            sector_n_found  equ     8008h           ; sector not found
= 8009                            print_out_paper equ     8009h           ; printer out of paper
= 800A                            write_fault     equ     800ah           ; write fault
= 800B                            read_fault      equ     800bh           ; read fault
= 800C                            gen_fault       equ     800ch           ; general failure
                          ;
                          ;
                          ;       ***************************
                          ;       *   status equates        *
                          ;       ***************************
                          ;
= 0100                            ok_stat_done    equ     0100h           ; done flag set only
= 0300                            buisy_status    equ     0300h           ; buisy flag set only (needed for s
                  tatus requests)
= 0000                            not_bui_stat    equ     0000h           ; buisy flag not set  ( "     "
                   "      "  )
                          ;
                          ;
                          ; ******************************
                          ; **   data block length's    **
                          ; ******************************
                          ;
= 0012                          srh_init_l      equ     18              ; init block
= 000D                          srh_flush_l     equ     13              ; flush block
= 000D                          srh_stat_l      equ     13              ; status request block
= 000E                          srh_nd_l        equ     14              ; non destructive read no wait block
= 0014                          srh_r_w_l       equ     20              ; read or write block

                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          kbd_status_rec record lgdat:1,keyundef:3,f1:1,f2:1,ibf:1,kbdat:1
                          ;       keyboard status byte  which is returned on rskey
                          ;
                          ;       lgdat = 1 if language byte is ready
                          ;       ibf   = 1 if output to kbd is not possible
                          ;       kbdat = 1 if character ready on keyboard
                          ;
                          ;
```

KBD-DRV

```
;
;
;
;       ****************************************
;       **                                    **
;       **      S T A T U S   W O R D         **
;       **                                    **
;       **      of STATIC REQUEST HEADER      **
;       **                                    **
;       ****************************************
;
;
stat_rec record stat_err:1,stat_reserve:5,stat_buisy:1,stat_done:1,stat_code:8
;               definition of status word :
;                       stat_err = error bit (0=okay,1=error)
;                       stat-reserve  = reserved bits , not used
;                       stat_buise    = buisy flag used for status requests
;                       stat_done     = flag : operation completed
;                       stat_code     = error description
;                                       code equates see dseg (include devhdr.asm)
operation_rec record esc_flg:1,funact:1,at_beg_act:1,disfk:1,undef:2,hebr_on:1,in_buff:1
;
;               definition of operation flags :
;                       esc_flg         = ESC-function active
;                       funact          = function key active
;                       at_beg_act      = function key not completed at
;                                         last input
;                       disfk           = disable function key
;                                         if set : function key code is
;                                               given back directly without
;                                               reading function table
;                                       no translation is done for C0-F3h
;                       undef           = not used jet
;                       hebr_on         = switches display of special hebrew
;                                          characters on or off
;                       in_buff         = input buffer full (1 byte only)
;
;
;
```

```
                               ;
                               ;
0008                           key_in_fl:
0008  C4 2E 0000 E                     les     bp,PTRSAV                        ; get data block address
000C  26: 80 7E 00 0D                  cmp     es:[bp].long,srh_flush_l
0011  73 03                            jae     key_in_fl_1                      .      ; test length of data block
0013  E9 0000 E                        jmp     near ptr ERROR_5
0016                           key_in_fl_1:
0016  80 26 0002 R FE                  and     byte ptr flag_buf,not mask in_buff      ; clear input buffer full flag
001B  80 26 0002 R BF                  and     flag_buf,not mask funact         ; reset function active flags
0020  80 26 0002 R DF                  and     flag_buf,not mask at_beg_act     ;

0025  E9 0000 E                        jmp near ptr EXIT                        ; input buffer not physically clear
                          ed
```

KBD-DRV

```
                              ;
0028                          key_in_p        proc
0028  3C 0D                           cmp     al,0dh                          ; on (CR) return
002A  74 1D                           jz      key_in_p2
002C  3C 7F                           cmp     al,7fh                          ; if line shall be cleared, return
002E  74 19                           jz      key_in_p2
0030                          key_in_p6:
0030  E8 0340 R                       call    near ptr kbd_in                 ; get char.from keyboard (waiting)
                              ; ******************************
                              ; **  echo to CRT hier    ********
                              ; ******************************
                              ;
0033  3C A0                           cmp     al,0a0h                         ; was it a function key ?
0035  73 0D                           jae     key_in_p1                       ; set function flag active ,return
0037  80 3E 0000 R 32                 cmp     language,32h                    ; for HEBREW switch on or
003C  74 0C                           jz      key_in_p3                       ; off special characters
003E                          key_in_p5:
003E  AA                              stosb                                   ; store incoming char.via string
003F  E2 E7                           loop    key_in_p                        ; loop until transfer buffer full
0041  EB 06 90                        jmp     near ptr key_in_p2              ; then return
0044                          key_in_p1:
0044  80 0E 0002 R 40                 or      flag_buf,mask funact            ; set function active flag
0049                          key_in_p2:
0049  C3                              ret                                     ; and return
004A                          key_in_p3:
004A  3C 9E                           cmp     al,9Eh                          ; HEBREW: switch on
004C  75 07                           jnz     key_in_p4
004E  80 0E 0002 R 02                 or      flag_buf,mask hebr_on
0053  EB DB                           jmp     near ptr key_in_p6
0055                          key_in_p4:
0055  3C 9F                           cmp     al,9Fh                          ; or off  display of
0057  75 E5                           jnz     key_in_p5                       ; special character
0059  80 26 0002 R FD                 and     flag_buf,not mask hebr_on
005E  EB D0                           jmp     near ptr key_in_p6
                              ;
0060                          key_in_p        endp
                              ;
                              ;
                              ;
0060                          new_funct       proc
0060  BE 0000 E                       mov     si,offset functbl
0063  BB 0000                         mov     bx,00h
0066  91                              xchg    ax,cx
0067  81 E1 001F                      and     cx,01fh                         ; get number of function
006B  83 F9 00                        cmp     cx,00h                          ; if first function is
006E  74 07                           jz      new_f2                          ; requested, take it directly
0070                          new_f1:
0070  8A 1C                           mov     bl,[ds:si]
0072  03 F3                           add     si,bx
0074  E2 FA                           loop    new_f1                          ; get correct offset of special fun
                        ction
0076  91                              xchg    cx,ax
0077                          new_f2:
0077  BB 0000                         mov     bx,00h
007A  8A 1C                           mov     bl,ds:[si]                      ; get length of function in BX
007C  4B                              dec     bx
007D  46                              inc     si                              ; set source pointer to first char.
007E  C3                              ret
007F                          new_funct       endp
```

```
007F                          old_funct     proc
007F  8B 36 0004 R                   mov    si,word ptr funcoff               ; get offset of first byte
                                                                              ; to be transfered for an
                                                                              ; not completed function
0083  8B 1E 0006 R                   mov    bx,word ptr remnum                ; number of bytes to be transfered
0087  C3                             ret
0088                          old_funct     endp
                              ;
                              ;
                              ;
0088                          trans_funct   proc                              ; transfer of bytes from function k
                   ey
                                                       ; already set : dx=trans.length
                                                       ;               bx=function length
                                                       ;               [ds:si] function offset
                                                       ;               [es:di] transfer address
0088  3B D3                          cmp    dx,bx
008A  73.12                          jae    trans_f1                          ; function can be transfered comple
                   tly
008C  8B CA                          mov    cx,dx                             ; set counter
008E  2B DA                          sub    bx,dx                             ; remainding bytes of function
0090  8A 0000                        mov    dx,00h                            ; no transaction bytes are left
0093  89 1E 0006 R                   mov    word ptr remnum,bx                ; are saved
0097  80 0E 0002 R 20                or     flag_buf,mask at_beg_act          ; set flag of not compl.funct.
009C  EB 07                          jmp    short trans_f2
009E                          trans_f1:
009E  8B CB                          mov    cx,bx                             ; set counter
00A0  2B D3                          sub    dx,bx                             ; save remaind.no.of transf.bytes
00A2  BB 0000                        mov    bx,00h                            ; no function bytes are left
00A5                          trans_f2:
00A5  F3/ A4                         rep    movsb                             ; move function via string
                              ; ############################
                              ; ####   echo to CRT   ########
                              ; ############################
                              ;
00A7  89 36 0004 R                   mov    word ptr funcoff,si               ; save address of next function byt
                   e
00AB  C3                             ret
00AC                          trans_funct   endp
                              ;
```

KBD-DRV

```
00AC                           key_in:
00AC  C4 2E 0000 E                     les     bp,PTRSAV                          ; get data block address
00B0  26: 80 7E 00 14                  cmp     es:[bp].long,srh_r_w_l
00B5  73 03                            jae     key_in_1
00B7  E9 0000 E                        jmp     near ptr ERROR_5                   ; if not: return this
00BA                           key_in_1:
00BA  C4 2E 0000 E                     les     bp,PTRSAV
00BE  FC                               cld
00BF  26: 8B 56 12                     mov     dx,es:[bp].s_b_count               ; get max.transf.length
00C3  83 FA 00                         cmp     dx,00h                       ; if no transaction is requested
00C6  75 03                            jnz     key_in_7
00C8  E9 0000 E                        jmp     near ptr EXIT          ; do nothing and return with done status
00CB                           key_in_7:
00CB  B8 0000                          mov     ax,00h                       ; clear accu
00CE  26: 8B 7E 0E                     mov     di,es:[bp].transadd_off            ; set pointer to transfer addr.
00D2  26: 8E 46 10                     mov     es,es:[bp].transadd_seg
00D6  F6 06 0002 R 20                  test    flag_buf,mask at_beg_act           ; was last function key not complet
                         ed
00DB  75 62                            jnz     key_in_2                           ; so get its pointers
00DD  F6 06 0002 R 01                  test    flag_buf,mask in_buff        ; is there a char. in input buffer
00E2  74 2E                            jz      key_in_3                           ; if not get key-input
00E4  A0 0003 R                        mov     al,byte ptr in_buffer              ; else get input byte
00E7  80 26 0002 R FE                  and     flag_buf,not mask in_buff          ; clear input buffer (flag)
00EC  3C A0                            cmp     al,0a0h                            ; was it a function key
00EE  72 4B                            jb      key_in_4                           ; if not store char.
00F0                           key_in_6:
00F0  A8 10                            test    al,10h                             ;; SPAR 02314
00F2  74 07                            jz      key_in_8                           ;; for code ) defined
00F4  A8 0C                            test    al,0ch                             ;; function key codes
00F6  74 03                            jz      key_in_8                           ;; nothing is
00F8  EB 18 90                         jmp     key_in_3                           ;; returned
00FB                           key_in_8:                                          ;; end of 02314
00FB  80 26 0002 R BF                  and     flag_buf,not mask funact           ; flag not needed any more
0100  F6 06 0002 R 10                  test    flag_buf,mask disfk                ; if function keys are disabled
0105  75 34                            jnz     key_in_4                           ; return its real code
0107  E8 0060 R                        call    near ptr new_funct                 ; get pointers of function
010A                           key_in_5:
010A  E8 0088 R                        call    near ptr trans_funct               ; store function via string
010D  83 FB 00                         cmp     bx,00h                             ;  function is not completed
0110  75 18                            jnz     key_in_end                         ; return with full transf.buffer
0112                           key_in_3:
0112  83 FA 00                         cmp     dx,00h                             ; if transf.buffer is full
0115  74 0E                            jz      key_in_end2                        ; return with function complete
0117  8B CA                            mov     cx,dx                              ; set string counter
0119  E8 0028 R                        call    near ptr key_in_p                  ; get char.from keyboard
011C  8B D1                            mov     dx,cx                              ; get remaind.transfer bytes
011E  F6 06 0002 R 40                  test    flag_buf,mask funact               ; if a function is active
0123  75 CB                            jnz     key_in_6                           ; take it
0125                           key_in_end2:
0125  80 26 0002 R DF                  and     flag_buf,not mask at_beg_act       ; reset uncomplete function flag
012A                           key_in_end:
012A  C4 2E 0000 E                     les     bp,PTRSAV                          ; reset [es:bp]
012E  26: 8B 46 12                     mov     ax,es:[bp].s_b_count
0132  2B C2                            sub     ax,dx                              ; subtract remainding transfer byte
                         s
0134  26: 89 46 12                     mov     es:[bp].s_b_count,ax               ; from trans.counter and return thi
                         s
0138  E9 0000 E                        jmp     near ptr EXIT
013B                           key_in_4:
```

```
013B  AA                          stosb                               ; store input buffer on transfer bl
                  ock
013C  4A                          dec     dx
013D  EB D3                       jmp     near ptr key_in_3
013F                    key_in_2:
013F  E8 007F R                   call    near ptr old_funct      ; get inform.about uncompleted function
0142  EB C6                       jmp     near ptr key_in_5

0144                    key_nd_in:
0144  C4 2E 0000 E                les     bp,PTRSAV                   ; get data block address
0148  26: 80 7E 00 0E             cmp     es:[bp].long,srh_nd_l       ; look for correct data block lengt
                  h
014D  73 03                       jae     key_nd_in_1
014F  E9 0000 E                   jmp     near ptr ERROR_5            ; if not return this
0152                    key_nd_in_1:
0152  8B FD                       mov     di,bp                   ; set destination pointer
0154  83 C7 0D                    add     di,read_byte
0157  F6 06 0002 R 20             test    flag_buf,mask at_beg_act    ; if a function key is not complete
                  d
015C  75 2B                       jnz     key_nd_2                    ; complete it first
015E  F6 06 0002 R 01             test    flag_buf,mask in_buff       ; is a char. in put buffer
0163  75 19                       jnz     key_nd_7                    ; yes, take it first
0165                    key_nd_4:
0165  E8 033D R                   call    near ptr kbd_st             ; look for char. ready on kbd
0168  A8 01                       test    al,mask kbdat               ; if not,
016A  74 45                       jz      key_nd_end                  ; return with busy bit on
016C                    key_nd_9:
016C  E8 0340 R                   call    near ptr kbd_in             ; else get it
016F  80 3E 0000 R 32             cmp     language,32h                ; on HEBREW switch on or
0174  74 3E                       jz      key_hebrew                  ; off special characters
0176                    key_nd_10:
0176  A2 0003 R                   mov     byte ptr in_buffer,al       ; in all diff.cases safe char. in b
                  uffer
0179  80 0E 0002 R 01             or      flag_buf,mask in_buff       ; set input buffer full
017E                    key_nd_7:
017E  A0 0003 R                   mov     al,byte ptr in_buffer       ; set char. in AL
0181  3C A0                       cmp     al,0a0h                     ; if it is a function key
0183  73 0A                       jae     key_nd_5                    ; set source pointers corr.
0185                    key_nd_8:
0185  AA                          stosb                               ; save the normal char.on data bloc
                  k
0186  EB 26 90                    jmp     near ptr key_nd_end1        ; return good status
0189                    key_nd_2:
0189  E8 007F R                   call    near ptr old_funct          ; take dest.pointers of old functio
                  n
018C  EB 1A 90                    jmp     near ptr key_nd_6           ; and give it to data block
018F                    key_nd_5:
018F  A8 10                       test    al,10h                      ;; SPAR 02314
0191  74 0B                       jz      key_nd_3                    ;; codes > function
0193  A8 0C                       test    al,0ch                      ;; key codes return
0195  74 07                       jz      key_nd_3                    ;; nothing
0197  80 26 0002 R FE             and     flag_buf,not mask in_buff   ;;
019C  EB C7                       jmp     key_nd_4                    ;;
019E                    key_nd_3:                                     ;; end of  02314
019E  F6 06 0002 R 10             test    flag_buf,mask disfk         ; if function keys are disabled
01A3  75 E0                       jnz     key_nd_8                    ; return its real code
01A5  E8 0060 R                   call    near ptr new_funct          ; take dest.pointers of new functio
                  n
01A8                    key_nd_6:
01A8  A4                          movsb                               ; and transfer first byte to data b
                  lock
01A9  80 26 0002 R BF             and     flag_buf,not mask funact    ; reset function active flag
```

KBD-DRV

```
01AE                            key_nd_end1:
01AE  E9 0000 E                         jmp     near ptr EXIT                   ; return with good status
01B1                            key_nd_end:
01B1  E9 0000 E                         jmp     near ptr BUS_EXIT               ; set busy flag on
01B4                            key_hebrew:                                     ; for implementation of
01B4  3C 9E                             cmp     al,9Eh                          ; HEBREW language
01B6  75 07                             jnz     key_heb1                        ; switch on or off
01B8  80 0E 0002 R 02                   or      flag_buf,mask hebr_on           ; display of Hebrew
01BD  EB A6                             jmp     near ptr key_nd_4                       ; characters
01BF                            key_heb1:
01BF  3C 9F                             cmp     al,9fh
01C1  75 B3                             jnz     key_nd_10
01C3  80 26 0002 R FD                   and     flag_buf,not mask hebr_on
01C8  EB 9B                             jmp     near ptr key_nd_4
                                ;
                                ;
                                ;
01CA                            key_st:
01CA  C4 2E 0000 E                      les     bp,PTRSAV                       ; get data block address
01CE  26: 80 7E 00 DD                   cmp     es:[bp].long,srh_stat_l
01D3  73 03                             jae     key_st_3                                ; test length of data block
01D5  E9 0000 E                         jmp     near ptr ERROR_5
01D8                            key_st_3:
01D8  F6 06 0002 R 20                   test    flag_buf,mask at_beg_act                ; look at active function k
                    eys
01DD  75 11                             jnz     key_st_1.
01DF  F6 06 0002 R 01                   test flag_buf,mask in_buff                 ; when input buffer not empty
01E4  75 0A                             jnz     key_st_1                        ; return this
01E6  E8 033D R                         call near ptr kbd_st                    ; get keyboard status
01E9  A8 01                             test    al,mask kbdat                   ; when character ready on keyboard
01EB  75 03                             jnz     key_st_1                        ; return this equ. to no empty inp.
                    buffer
01ED  E9 0000 E                         jmp     near ptr BUS_EXIT               ; else return buisy flag on
                                ;
01F0                            key_st_1:
01F0  E9 0000 E                         jmp     near ptr EXIT                   ; set buisy flag off
                                ;
```

KBD-DRV

```
                        ;
                        ;
01F3                    key_init:
01F3  C4 2E 0000 E              les     bp,PTRSAV                       ; get dat block address
01F7  26: 80 7E 00 12           cmp     es:[bp].long,srh_init_l         ; test length of data block
01FC  73 03                     jae     key_init_1                                          ; return status "ba
                  d drive request str. length
01FE  E9 0000 E                 jmp     near ptr ERROR_5
0201                    key_init_1:
0201  C6 06 0002 R 00           mov     byte ptr flag_buf,00h           ; set all operation flags reture
0206  E8 02B8 R                 call    near ptr kbd_init               ; get language code from keyboard
                        ; this procedure stores the gotten code on byte LANGUAGE
                        ;
0209  26: C6 46 00 01           mov     es:[bp].unit_no,no_of_units     ; return number of units belonging
                  to this driver
020E  80 3E 0001 R C8           cmp     kbd_var,0C8h            ; for undefined keyboard
0213  73 08                     jae     key_init_2              ; variantes
0215  80 26 0000 R 07           and     language,07h            ; set old one,without
021A  E9 0000 E                 jmp     near ptr ERROR_11       ; changing the language ÷
                                                                ; and return a read error
021D                    key_init_2:
021D  E9 0000 E                 jmp     near ptr EXIT
                        ;
                        ;
                        ;
                        ;
                        ;       ************************************************
                        ;       ************************************************
                        ;       ***                                          ***
                        ;       ***           D E F _ F U N                  ***
                        ;       ***                                          ***
                        ;       ***            Subroutine                    ***
                        ;       ***                                          ***
                        ;       ************************************************
                        ;       ************************************************
                        ;
0220  0000                      remlen  dw      00h                 ; number of bytes of all functions behind
                                                                    ; the changing one
```

```
0222                    def_fun        proc
0222  56                       push    si
0223  06                       push    es
0224  53                       push    bx
0225  51                       push    cx                      ; save all incoming data
0226  FC                       cld
                                                               ; deleted for CONFIG
                                                               ; see SPAR G2344
0227  B8 0000                  mov     ax,00h
022A  26: 8A 07                mov     al,es:[bx]              ; AL = # of function that is
                                                               ; to be changed
022D  3C 00                    cmp     al,00h                  ; function # 0 :
022F  74 14                    jz      set_fun_dis             ;   disable function keys
0231  3C 14                    cmp     al,20d                  ; function # 1 - 20 :
0233  76 1E                    jbe     reset_fun               ;   change the function contents
0235  3C 63                    cmp     al,99d                  ; function # 99 :
0237  74 13                    jz      set_fun_en              ;   enable function keys
0239                    ret_fail:
0239  B0 0C                    mov     al,0Ch                  ; for unallowed function numbers
023B  EB 03 90                 jmp     near ptr ret_end        ; a failure "general failure"
                                                               ; is returned
                                                               ; also diff. deficulties end in this way

023E                    ret_okay:
023E  B0 00                    mov     al,00h                  ; END: the complete sequence is stored
0240                    ret_end:
0240  59                       pop     cx
0241  5B                       pop     bx
0242  07                       pop     es                      ; return with string address
0243  5E                       pop     si
0244  C3                       ret
                        ;
                        ;
                        ;
                        ;
0245                    set_fun_dis:
0245  80 0E 0002 R 10          or      flag_buf,mask disfk     ; set flag : getting contents of
024A  EB F2                    jmp     near ptr ret_okay       ; function keys is disabled
                        ;
                        ;
024C                    set_fun_en:
024C  80 26 0002 R EF          and     flag_buf,not mask disfk ; reset flag : getting contents
0251  EB EB                    jmp     near ptr ret_okay       ; of function keys is enabled
                        ;
                        ;
                        ;
                        ;
                        ;
0253                    reset_fun:
0253  FE C8                    dec     al
0255  50                       push    ax
0256  B8 0014                  mov     ax,20d                  ; save number of function
0259  E8 0060 R                call    near ptr new_funct      ; get complete function length
025C  89 36 0220 R             mov     remlen,si               ; length includes offset
0260  01 1E 0220 R             add     -remlen,bx
0264  58                       pop     ax                      ; get number of requested function
0265  E8 0060 R                call    near ptr new_funct      ; get possition of req.function
                                                               ; and its old length
0268  43                       inc     bx                      ; old length (complete)
0269  B8 0000 E                mov     ax,offset functbl
```

```
026C  05 0200                  add   ax,funclen
026F  2B 06 0220 R             sub   ax,remlen
0273  03 C3                    add   ax,bx                      ; max.length new contents can be
0275  4E                       dec   si                         ; address of 1.byte (length)
0276  29 36 0220 R             sub   remlen,si                  ; length of functions ) req.one
027A  29 1E 0220 R             sub   remlen,bx
027E  59                       pop   cx
027F  51                       push  cx                         ;
                                                                ; get correct length of new
                                                                ; string (incl.length byte)
0280  3B C8                    cmp   cx,ax                      ; if whole string is too long
0282  77 B5                    ja    ret_fail                   ; return with an error
0284  56                       push  si                         ; save begin of function string
0285  3B CB                    cmp   cx,bx
0287  77 28                    ja    res4                       ; if new one is longer
                                                                ; than old one
0289  74 12                    jz    res2                       ; new contents as long as old one
                                                  ; no move of remainding functions is needed
028B                     res1:
028B  8B FE                    mov   di,si                      ; set pointers to move the
028D  03 F9                    add   di,cx
028F  8C C8                    mov   ax,cs                      ; remainding functions
0291  8E C0                    mov   es,ax
0293  03 F3                    add   si,bx
0295  8B 0E 0220 R             mov   cx,remlen
0299  41                       inc   cx
029A  F3/ A4                   rep movsb
                         ;
029C  FC                       cld                              ; direction now up
029D                     res2:
029D  8C C8                    mov   ax,cs
029F  8E C0                    mov   es,ax                      ; set pointers to move the
02A1  5F                       pop   di                         ; new contents
02A2  59                       pop   cx
02A3  5E                       pop   si
02A4  46                       inc   si
02A5  1F                       pop   ds
02A6  83 EC 06                 sub   sp,6                       ; reset stack pointer
02A9  8B C1                    mov   ax,cx                      ; first write length of string
02AB  AA                       stosb                            ; of the new function
02AC  49                       dec   cx                         ; cx= string length
02AD  F3/ A4                   rep   movsb                      ; store whole contents of new
                                                                ; function
02AF  EB 8D                    jmp   near ptr ret_okay          ; return with good status
02B1                     res4:
02B1  03 36 0220 R             add   si,remlen                  ; set pointers to end of table
02B5  FD                       std                              ; direction : down
02B6  EB D3                    jmp   near ptr res1              ; continue
02B8                     def_fun endp
                         ;
                         ;
                         ;
```

```
C  include kbdpimc.seg
C  ;
C  ;
C  ;       **************************************************
C  ;       **                                              **
C  ;       **              K E Y B O A R D                 **
C  ;       **                                              **
C  ;       **                  P I M                       **
C  ;       **                                              **
C  ;       **************************************************
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       KBD_INIT
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           INITIALIZE THE KEYBOARD AND GET ITS LANGUAGE CODE
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   MUST BE FIRST ROUTINE ON KEYBOARD AFTER THE POWER UP
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    AL = LANGUAGE CODE  (00H - 07H)
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
02B8                         C  kbd_init:
02B8  B0 01                  C          mov    al,country                   ; load command to get language code
02BA  E6 41                  C          out    byte ptr kcount,al           ; send this command
02BC                         C  kbd_init_1:
02BC  E4 41                  C          in     al,byte ptr rskey            ; get keyboard status
02BE  A8 01                  C          test   al,kbdat86                   ; when data not ready
02C0  74 FA                  C          jz     kbd_init_1                   ; try again (loop)
02C2  E4 41                  C          in     al,byte ptr rskey            ;
02C4  A8 80                  C          test   al,lgdat86                   ; when language code ready
02C6  75 04                  C          jnz    kbd_init_2                   ; get it
02C8  E4 40                  C          in     al,byte ptr rdkey            ; dummy read neede for 8741 control
                          ler
02CA  EB F0                  C          jmp    kbd_init_1                   ; try again
02CC                         C  kbd_init_2:
02CC  E4 40                  C          in     al,byte ptr rdkey            ; get language code
02CE  C6 06 0000 R 07        C          mov    language,07h
02D3  20 06 0000 R           C          and    language,al                  ; clear bits:7,...,3
02D7  24 F8                  C          and    al,not 07h                   ; clear lower bits
02D9  B9 0003                C          mov    cx,03h                       ; look for the 3 variantes
02DC                         C  kbd_init_4:
02DC  3A 06 0001 R           C          cmp    al,kbd_var                   ; get # of
02E0  74 0C                  C          jz     kbd_init_5                   ; keyboard variante
02E2  80 06 0000 R 10        C          add    language,10h                 ; and change
02E7  80 2E 0001 R 10        C          sub    kbd_var,10h                  ; language code
02EC  E2 EE                  C          loop   kbd_init_4                   ; accordingly
02EE                         C  kbd_init_5:
02EE  80 3E 0000 R 01        C          cmp    language,01h                 ; if language is
02F3  76 3D                  C          jbe    kbd_init_6
02F5  80 3E 0000 R 10        C          cmp    language,10h
02FA  74 36                  C          jz     kbd_init_6
02FC  80 3E 0000 R 11        C          cmp    language,11h
0301  74 2F                  C          jz     kbd_init_6
0303  80 3E 0000 R 23        C          cmp    language,23h                 ; CANADA
0308  74 28                  C          jz     kbd_init_6
030A  80 3E 0000 R 32        C          cmp    language,32h                 ; HEBREW
030F  74 17                  C          jz     kbd_init_7
0311  C6 06 0000 E 2C        C          mov    byte ptr dec_sign_1,2ch      ;; SPAR 02332
0316  C6 06 0000 E 2C        C          mov    byte ptr dec_sign_2,2ch      ;;
031B  BF 001E E              C          mov    di,offset kbd_tt +1eh
031E  C6 05 1E               C          mov    byte ptr [di],1eh            ; for Hebrew the codes
0321  47                     C          inc    di                           ; 9Eh and 9Fh switch on

0322  C6 05 1F               C          mov    byte ptr [di],1fh            ; and off display of
0325  EB 15 90               C          jmp    kbd_init_3
0328                         C  kbd_init_7:
0328  BF 001E E              C          mov    di,offset kbd_tt +1eh
032B  C6 05 9E               C          mov    byte ptr [di],9eh            ; for Hebrew the codes
032E  47                     C          inc    di                           ; 9Eh and 9Fh switch on
032F  C6 05 9F               C          mov    byte ptr [di],9fh            ; and off display of
                             C                                              ; hebrew characters
0332                         C  kbd_init_6:
0332  C6 06 0000 E 2E        C          mov    byte ptr dec_sign_1,2eh      ; 00 = us or  01 = uk
0337  C6 06 0000 E 2E        C          mov    byte ptr dec_sign_2,2eh      ; use decimal point
                             C                                              ; instead of comma
033C                         C  kbd_init_3:
033C  C3                     C          ret
```

```
                        C ;
                        C ;***************************************************************************
                        C ;***************************************************************************
                        C ;***************************************************************************
                        C
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;   ROUTINE NAME:       KBD_ST
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;   FUNCTION:           GET STATUS OF KEYBOARD CONTROLLER
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;   ENTRY VIA:          CALL
                        C ;
                        C ;
                        C ;   ENTRY CONDITIONS:   NON
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;   EXIT VIA:           RETURN
                        C ;
                        C ;
                        C ;   EXIT CONDITIONS:    AL = STATUS OF KEYBOARD CONTROLLER
                        C ;
                        C ;
                        C ;
                        C ;***************************************************************************
                        C ;***************************************************************************
                        C ;***************************************************************************
                        C ;
                        C ;
                        C ;
033D                    C kbd_st:
033D  E4 41             C       in      al,byte ptr rskey                 ; get status of keyboard controller
033F  C3                C       ret
```

```
                        C  ;**********************************************************************
                        C  ;**********************************************************************
                        C  ;**********************************************************************
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;    ROUTINE NAME:       KBD_IN
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;    FUNCTION:           GET AN INPUT FROM KEYBOARD
                        C  ;                        (AND WAIT UNTIL ONE IS COMING
                        C  ;
                        C  ;
                        C  ;
                        C  ;    ENTRY VIA:          CALL
                        C  ;
                        C  ;
                        C  ;    ENTRY CONDITIONS:   NON
                        C  ;
                        C  ;
                        C  ;
                        C  ;
                        C  ;    EXIT VIA:           RETURN
                        C  ;
                        C  ;
                        C  ;    EXIT CONDITIONS:    AL = CHARACTER FROM KEYBOARD INPUT
                        C  ;
                        C  ;
                        C  ;
                        C  ;**********************************************************************
                        C  ;**********************************************************************
                        C  ;**********************************************************************
                        C  ;
                        C  ;
                        C  ;
0340                    C  kbd_in:
0340  E4 41             C          in      al,byte ptr rskey              ; wait for character ready
0342  A8 01             C          test    al,kbdat86
0344  74 FA             C          jz      kbd_in                         ;  (loop)
0346  E4 40             C          in      al,byte ptr rdkey              ; get character for keyboard
0348  3C 80             C          cmp     al,80h                         ; if char is a ASCII one
034A  72 0A             C          jb      kbd_in_2                       ; okay  return
034C  3C A0             C          cmp     al,0a0h                        ; also function keys are returned
034E  73 06             C          jae     kbd_in_2
0350  24 1F             C          and     al,1fh                         ; all char. ) 80h and ( a0h
0352  BB 0000 E         C          mov     bx,offset kbd_tt               ; are translated
0355  D7                C          xlat                                   ; by the keyboard translation table
                        C                                                 ; the character ) 80h

0356                    C  kbd_in_2:
0356  C3                C          ret
```

KBD-DRV

```
                   C  ;**********************************************************************
                   C  ;**********************************************************************
                   C  ;**********************************************************************
                   C  ;
                   C  ;
                   C  ;
                   C  ;    DATE: 83/02/25      AUTHOR: H.M}ller
                   C  ;
                   C  ;
                   C  ;
                   C  ;    ROUTINE NAME:       KBD_OUT
                   C  ;
                   C  ;
                   C  ;
                   C  ;
                   C  ;
                   C  ;    FUNCTION:           OUTPUT TO KEYBOARD
                   C  ;
                   C  ;
                   C  ;
                   C  ;
                   C  ;    ENTRY VIA:          CALL
                   C  ;
                   C  ;
                   C  ;    ENTRY CONDITIONS:   CL = CHARACTER FOR RETREIVE ON KEYBOARD
                   C  ;                        (WAITING UNTIL KEYBOARD CAN TAKE IT)
                   C  ;
                   C  ;
                   C  ;
                   C  ;    EXIT VIA:           RETURN
                   C  ;
                   C  ;
                   C  ;    EXIT CONDITIONS:    NON
                   C  ;
                   C  ;
                   C  ;
                   C  ;**********************************************************************
                   C  ;**********************************************************************
                   C  ;**********************************************************************
                   C  ;
                   C  ;
                   C  ;
0357               C  kbd_out:
0357               C  kbd_out_2:
                   C                                                  ; output character in CL
0357  E4 41        C          in      al,byte ptr rskey               ; get keyboard status
0359  A8 01        C          test    al,kbdat86                      ; when a character is ready
035B  74 02        C          jz      kbd_out_1                       ;
035D  E4 40        C          in      al,byte ptr rdkey               ; do a dummy read (needed for 8741
              controller)
035F               C  kbd_out_1:
035F  E4 41        C          in      al,byte ptr rskey               ; get keyboard status
0361  A8 02        C          test    al,inpbuff86                    ; and check whether output to kbd c

              an be done
0363  75 F2        C          jnz      kbd_out_2                      ; if not, try again
0365  8A C1        C          mov      al,cl                          ; get character for output
0367  E6 41        C          out      byte ptr kbell,al              ; and send it
0369  C3           C          ret
                   C  ;
                   C  ;
                   C  ;
                   C  ;
                      ;
                      ;
036A                  CSEG    ends
                      end
```

```
Structures and records:

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

ADDRESS_STRUC. . . . . . . . . .        0004    0002
  OFF. . . . . . . . . . . . . . .      0000
  SEG. . . . . . . . . . . . . . .      0002
INIT_STRUC . . . . . . . . . . .        0012    0004
  UNIT_NO. . . . . . . . . . . . .      000D
  BREAKADD_OFF . . . . . . . . . .      000E
  BREAKADD_SEG . . . . . . . . . .      0010
KBD_STATUS_REC . . . . . . . . .        0008    0006
  LGDAT. . . . . . . . . . . . . .      0007    0001    0080    0000
  KEYUNDEF . . . . . . . . . . . .      0004    0003    0070    0000
  F1 . . . . . . . . . . . . . . .      0003    0001    0008    0000
  F2 . . . . . . . . . . . . . . .      0002    0001    0004    0000
  IBF. . . . . . . . . . . . . . .      0001    0001    0002    0000
  KBDAT. . . . . . . . . . . . . .      0000    0001    0001    0000
NON_DESR_READ. . . . . . . . . .        000E    0002
  READ_BYTE. . . . . . . . . . . .      000D
OPERATION_REC. . . . . . . . . .        0008    0007
  ESC_FLG. . . . . . . . . . . . .      0007    0001    0080    0000
  FUNACT . . . . . . . . . . . . .      0006    0001    0040    0000
  AT_BEG_ACT . . . . . . . . . . .      0005    0001    0020    0000
  DISFK. . . . . . . . . . . . . .      0004    0001    0010    0000
  UNDEF. . . . . . . . . . . . . .      0002    0002    000C    0000
  HEBR_ON. . . . . . . . . . . . .      0001    0001    0002    0000
  IN_BUFF. . . . . . . . . . . . .      0000    0001    0001    0000
READ_WRITE_DATA. . . . . . . . .        0014    0005
  MEDIA_DESC . . . . . . . . . . .      000D
  TRANSADD_OFF . . . . . . . . . .      000E
  TRANSADD_SEG . . . . . . . . . .      0010
  S_B_COUNT. . . . . . . . . . . .      0012
SPECIAL_DEV_HEADER . . . . . . .        0012    0005
  POINT_NXT. . . . . . . . . . . .      0000
  ATTRIBUTE. . . . . . . . . . . .      0004
  STRAT_ENTRY. . . . . . . . . . .      0006
  INTER_ENTRY. . . . . . . . . . .      0008
  DEV_NAME . . . . . . . . . . . .      000A
STATIC_REQUEST_HEADER. . . . . .        000D    0006
  LONG . . . . . . . . . . . . . .      0000
  SUBUNIT. . . . . . . . . . . . .      0001
  COMMAND. . . . . . . . . . . . .      0002
  STATUS . . . . . . . . . . . . .      0003
  DOS_QUE_ADDR . . . . . . . . . .      0005
  DEV_QUE_ADDR . . . . . . . . . .      0009
STAT_REC . . . . . . . . . . . .        0010    0005
  STAT_ERR . . . . . . . . . . . .      000F    0001    8000    0000
  STAT_RESERVE . . . . . . . . . .      000A    0005    7C00    0000
  STAT_BUISY . . . . . . . . . . .      0009    0001    0200    0000
  STAT_DONE. . . . . . . . . . . .      0008    0001    0100    0000
  STAT_CODE. . . . . . . . . . . .      0000    0008    00FF    0000

Segments and groups:

                N a m e                 Size    align   combine class

CSEG . . . . . . . . . . . . . .        036A    PARA    PUBLIC  'CODE'
```

KBD-DRV

```
Symbols:

                N a m e             Type    Value   Attr

BAD_SRH_LENGTH . . . . . . . . . .  Number  8005
BUISY_STATUS . . . . . . . . . . .  Number  0300
BUS_EXIT . . . . . . . . . . . . .  L NEAR  0000            External
CLEAR_1. . . . . . . . . . . . . .  V BYTE  0000            External
CLEAR_2. . . . . . . . . . . . . .  V BYTE  0000            External
COUNTRY. . . . . . . . . . . . . .  Number  0001
CRC_ERROR. . . . . . . . . . . . .  Number  8004
DEC_SIGN_1 . . . . . . . . . . . .  V BYTE  0000            External
DEC_SIGN_2 . . . . . . . . . . . .  V BYTE  0000            External
DEF_FUN. . . . . . . . . . . . . .  N PROC  0222    CSEG    Global  Length =0096
DRIVE_N_READY. . . . . . . . . . .  Number  8002
ERROR_0. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_1. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_10 . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_11 . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_12 . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_2. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_3. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_4. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_5. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_6. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_7. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_8. . . . . . . . . . . . . .  L NEAR  0000            External
ERROR_9. . . . . . . . . . . . . .  L NEAR  0000            External
EXIT . . . . . . . . . . . . . . .  L NEAR  0000            External
EXIT1. . . . . . . . . . . . . . .  L NEAR  0000            External
FLAG_BUF . . . . . . . . . . . . .  L BYTE  0002    CSEG    Global
FUNCLEN. . . . . . . . . . . . . .  Number  0200
FUNCOFF. . . . . . . . . . . . . .  L WORD  0004    CSEG    Global
FUNCTBL. . . . . . . . . . . . . .  V BYTE  0000            External
GEN_FAULT. . . . . . . . . . . . .  Number  800C
INPBUFF86. . . . . . . . . . . . .  Number  0002
IN_BUFFER. . . . . . . . . . . . .  L BYTE  0003    CSEG
KBDAT86. . . . . . . . . . . . . .  Number  0001
KBD_IN . . . . . . . . . . . . . .  L NEAR  0340    CSEG
KBD_INIT . . . . . . . . . . . . .  L NEAR  02B8    CSEG
KBD_INIT_1 . . . . . . . . . . . .  L NEAR  02BC    CSEG
KBD_INIT_2 . . . . . . . . . . . .  L NEAR  02CC    CSEG
KBD_INIT_3 . . . . . . . . . . . .  L NEAR  033C    CSEG
KBD_INIT_4 . . . . . . . . . . . .  L NEAR  02DC    CSEG
KBD_INIT_5 . . . . . . . . . . . .  L NEAR  02EE    CSEG
KBD_INIT_6 . . . . . . . . . . . .  L NEAR  0332    CSEG
KBD_INIT_7 . . . . . . . . . . . .  L NEAR  0328    CSEG
KBD_IN_2 . . . . . . . . . . . . .  L NEAR  0356    CSEG
KBD_OUT. . . . . . . . . . . . . .  L NEAR  0357    CSEG    Global
KBD_CUT_1. . . . . . . . . . . . .  L NEAR  035F    CSEG
KBD_OUT_2. . . . . . . . . . . . .  L NEAR  0357    CSEG
KBD_ST . . . . . . . . . . . . . .  L NEAR  033D    CSEG
KBD_TT . . . . . . . . . . . . . .  V BYTE  0000            External
KBD_VAR. . . . . . . . . . . . . .  L BYTE  0001    CSEG
KBELL. . . . . . . . . . . . . . .  Number  0041
KCOUNT . . . . . . . . . . . . . .  Number  0041
KEYBASE. . . . . . . . . . . . . .  Number  0040
KEY_HEB1 . . . . . . . . . . . . .  L NEAR  01BF    CSEG
KEY_HEBREW . . . . . . . . . . . .  L NEAR  01B4    CSEG
KEY_IN . . . . . . . . . . . . . .  L NEAR  00AC    CSEG    Global
KEY_INIT . . . . . . . . . . . . .  L NEAR  01F3    CSEG    Global
```

```
KEY_INIT_1 . . . . . . . . . .    L NEAR  0201    CSEG
KEY_INIT_2 . . . . . . . . . .    L NEAR  021D    CSEG
KEY_IN_1 . . . . . . . . . . .    L NEAR  00BA    CSEG
KEY_IN_2 . . . . . . . . . . .    L NEAR  013F    CSEG
KEY_IN_3 . . . . . . . . . . .    L NEAR  0112    CSEG
KEY_IN_4 . . . . . . . . . . .    L NEAR  013B    CSEG
KEY_IN_5 . . . . . . . . . . .    L NEAR  010A    CSEG
KEY_IN_6 . . . . . . . . . . .    L NEAR  00FD    CSEG
KEY_IN_7 . . . . . . . . . . .    L NEAR  00CB    CSEG
KEY_IN_8 . . . . . . . . . . .    L NEAR  00FB    CSEG
KEY_IN_END . . . . . . . . . .    L NEAR  012A    CSEG
KEY_IN_END2. . . . . . . . . .    L NEAR  0125    CSEG
KEY_IN_FL. . . . . . . . . . .    L NEAR  0008    CSEG    Global
KEY_IN_FL_1. . . . . . . . . .    L NEAR  0016    CSEG
KEY_IN_P . . . . . . . . . . .    N PROC  0028    CSEG    Length =0038
KEY_IN_P1. . . . . . . . . . .    L NEAR  0044    CSEG
KEY_IN_P2. . . . . . . . . . .    L NEAR  0049    CSEG
KEY_IN_P3. . . . . . . . . . .    L NEAR  004A    CSEG
KEY_IN_P4. . . . . . . . . . .    L NEAR  0055    CSEG
KEY_IN_P5. . . . . . . . . . .    L NEAR  003E    CSEG
KEY_IN_P6. . . . . . . . . . .    L NEAR  0030    CSEG
KEY_ND_10. . . . . . . . . . .    L NEAR  0176    CSEG
KEY_ND_2 . . . . . . . . . . .    L NEAR  0189    CSEG
KEY_ND_3 . . . . . . . . . . .    L NEAR  019E    CSEG
KEY_ND_4 . . . . . . . . . . .    L NEAR  0165    CSEG
KEY_ND_5 . . . . . . . . . . .    L NEAR  018F    CSEG
KEY_ND_6 . . . . . . . . . . .    L NEAR  01A8    CSEG
KEY_ND_7 . . . . . . . . . . .    L NEAR  017E    CSEG
KEY_ND_8 . . . . . . . . . . .    L NEAR  0185    CSEG
KEY_ND_9 . . . . . . . . . . .    L NEAR  016C    CSEG
KEY_ND_END . . . . . . . . . .    L NEAR  01B1    CSEG
KEY_ND_END1. . . . . . . . . .    L NEAR  01AE    CSEG
KEY_ND_IN. . . . . . . . . . .    L NEAR  0144    CSEG    Global
KEY_ND_IN_1. . . . . . . . . .    L NEAR  0152    CSEG
KEY_ST . . . . . . . . . . . .    L NEAR  01CA    CSEG    Global
KEY_ST_1 . . . . . . . . . . .    L NEAR  01F0    CSEG
KEY_ST_3 . . . . . . . . . . .    L NEAR  01D8    CSEG
LANGUAGE . . . . . . . . . . .    L BYTE  0000    CSEG    Global
LGDAT86. . . . . . . . . . . .    Number  0080
NEW_F1 . . . . . . . . . . . .    L NEAR  0070    CSEG
NEW_F2 . . . . . . . . . . . .    L NEAR  0077    CSEG
NEW_FUNCT. . . . . . . . . . .    N PROC  0060    CSEG    Length =001F
NOT_BUI_STAT . . . . . . . . .    Number  0000
NO_OF_UNITS. . . . . . . . . .    Number  0001
OK_STAT_DONE . . . . . . . . .    Number  0100
OLD_FUNCT. . . . . . . . . . .    N PROC  007F    CSEG    Length =0009
PRINT_OUT_PAPER. . . . . . . .    Number  8009
PTRSAV . . . . . . . . . . . .    V DWORD 0000            External
RDKEY. . . . . . . . . . . . .    Alias   KEYBASE
READ_FAULT . . . . . . . . . .    Number  800B
REMLEN . . . . . . . . . . . .    L WORD  0220    CSEG
REMNUM . . . . . . . . . . . .    L WORD  0006    CSEG    Global
RES1 . . . . . . . . . . . . .    L NEAR  0288    CSEG
RES2 . . . . . . . . . . . . .    L NEAR  0290    CSEG
RES4 . . . . . . . . . . . . .    L NEAR  02B1    CSEG
RESET_FUN. . . . . . . . . . .    L NEAR  0253    CSEG
RET_END. . . . . . . . . . . .    L NEAR  0240    CSEG
RET_FAIL . . . . . . . . . . .    L NEAR  0239    CSEG
RET_OKAY . . . . . . . . . . .    L NEAR  023E    CSEG
RSKEY. . . . . . . . . . . . .    Number  0041
```

KBD-DRV

```
SECTOR_N_FOUND . . . . . . . . .        Number  8008
SEEK_ERROR . . . . . . . . . . .        Number  8006
SET_FUN_DIS. . . . . . . . . . .        L NEAR  0245    CSEG
SET_FUN_EN . . . . . . . . . . .        L NEAR  024C    CSEG
SRH_FLUSH_L. . . . . . . . . . .        Number  000D
SRH_INIT_L . . . . . . . . . . .        Number  0012
SRH_ND_L . . . . . . . . . . . .        Number  000E
SRH_R_W_L. . . . . . . . . . . .        Number  0014
SRH_STAT_L . . . . . . . . . . .        Number  000D
TRANS_F1 . . . . . . . . . . . .        L NEAR  009E    CSEG
TRANS_F2 . . . . . . . . . . . .        L NEAR  00A5    CSEG
TRANS_FUNCT. . . . . . . . . . .        N PROC  0088    CSEG    Length =0024
UNKNOW_COMM. . . . . . . . . . .        Number  8003
UNKNOW_MEDIA . . . . . . . . . .        Number  8007
UNKNOW_UNIT. . . . . . . . . . .        Number  8001
WOKEY. . . . . . . . . . . . . .        Alias   KEYBASE
WRITE_FAULT. . . . . . . . . . .        Number  800A
WRITE_PROTECT. . . . . . . . . .        Number  8000
```

```
0000                           CSEG    SEGMENT        PUBLIC 'CODE'

                               ASSUME  CS:CSEG,DS:CSEG,SS:CSEG,ES:CSEG

                               PUBLIC  DSK_INT,DREND

                               EXTRN   ERROR_0:NEAR,ERROR_1:NEAR,ERROR_2:NEAR,ERROR_3:NEAR
                               EXTRN   ERROR_4:NEAR,ERROR_5:NEAR,ERROR_6:NEAR,ERROR_7:NEAR
                               EXTRN   ERROR_8:NEAR,ERROR_9:NEAR,ERROR_10:NEAR,ERROR_11:NEAR
                               EXTRN   ERROR_12:NEAR,EXIT:NEAR,PTRSAV:DWORD,DRVMAX:BYTE
                               EXTRN   FL_OUT_RETRIES:BYTE,FLTAB:WORD

                               ;
                               ; I/O data packet structure
                               ;
                               IODAT   STRUC
0000  ??                       CMDLEN  DB      ?               ; COMMAND LENGTH
0001  ??                       UNIT    DB      ?               ; UNIT NUMBER
0002  ??                       CMD     DB      ?               ; COMMAND CODE
0003  ????                     STATUS  DW      ?               ; RETURN STATUS
                                                               ;
0005     08 [                          DB      8 DUP (?)       ;
               ??
                   ]

000D  ??                       MEDIA   DB      ?               ; MEDIA DESCRIPTOR
000E  ????                     TRANS   DW      ?               ; TRANSFER ADDRESS (OFFSET)
0010  ????                             DW      ?               ;                 (SEGMENT)
0012  ????                     COUNT   DW      ?               ; COUNT OF BLOCKS
0014  ????                     START   DW      ?               ; FIRST BLOCK TO TRANSFER
0016                           IODAT   ENDS

                               ;
                               ; BIOS PARAMETER BLOCK structure
                               ;
                               DPB     STRUC
0000  ????                     SECSIZE DW      ?               ; Sector size in bytes
0002  ??                       ALLOC   DB      ?               ; Number of sectors per allocation unit
0003  ????                     RESSEC  DW      ?               ; Reserved sectors
0005  ??                       FATS    DB      ?               ; Number of FAT's
0006  ????                     MAXDIR  DW      ?               ; Number of root directory entries
0008  ????                     SECTORS DW      ?               ; Number of sectors per diskette
000A  ??                       MEDIAID DB      ?               ; Media byte ID
000B  ????                     FATSEC  DW      ?               ; Number of FAT sectors
000D                           DPB     ENDS
```

```
                         C         INCLUDE FLXPIMO.ASM
                         C ;       TITLE   FLEX DISK DRIVER PIM (DATA SEGMENT)
                         C
                         C
                         C ;
                         C ;
                         C ;
                         C ;*******************
                         C ;***  I/O PORTS  ***
                         C ;*******************
                         C ;
                         C ;
                         C ; FDC
                         C ; ---
                         C ;
= 0051                   C DCOMD   EQU     51H             ; DISK COMMAND PORT
= 0050                   C DSTAT   EQU     50H             ; DISK STATUS PORT
= 0051                   C FDCRA   EQU     51H             ; READ DMA FROM FDC PORT
                         C ;
                         C ;
                         C ;
                         C ; DMA
                         C ; ---
                         C ;
= 002A                   C DMAMB   EQU     2AH             ; WRITE SINGLE MASK REGISTER BIT
= 002B                   C DMAMO   EQU     2BH             ; DMA MODE PORT
= 0026                   C COAD    EQU     26H             ; DMA ADDR PORT
= 0027                   C COTC    EQU     27H             ; DMA LENGTH PORT
                         C ;
                         C ;
                         C ;
                         C ; SYSTEM STATUS
                         C ; -------------
                         C ;
= 0013                   C SYSSTA  EQU     13H             ; SYSTEM STATUS PORT
= 0014                   C MOTORON EQU     14H             ; MOTOR ON PORT
                         C ;
                         C ;
                         C ;
                         C ; BANK SELECT
                         C ; -----------
                         C ;
= 00E0                   C BANK    EQU     0E0H            ; BANK SELECT E0 :    0K -  64K
                         C                                 ;             E1 :  64K - 128K
                         C                                 ;             E2 : 128K - 196K
                         C                                 ;             E3 : 196K - 256K
                         C ;
                         C ;
                         C ;
```

```
                        C
                        C ;
                        C ;
                        C ;
                        C ;*********************
                        C ;***  FDC COMMANDS  ***
                        C ;*********************
                        C ;
                        C ;
= 0002                  C READTRK EQU     02H          ; READ TRACK COMMAND
= 0005                  C WRITDAT EQU     05H          ; WRITE DATA COMMAND
= 0006                  C READDAT EQU     06H          ; READ DATA COMMAND
= 0007                  C RESTORE EQU     07H          ; RESTORE COMMAND
= 0008                  C FDCSIS  EQU     08H          ; SENSE INTERRUPT STATUS
= 000A                  C IDREAD  EQU     0AH          ; READ ID COMMAND
= 000D                  C WRITFMT EQU     0DH          ; FORMAT A TRACK
= 000F                  C SEEKTRK EQU     0FH          ; SEEK A TRACK
                        C ;
                        C ;
                        C ;
                        C ;
                        C ;**********************
                        C ;***  FDC VARIABLES  ***
                        C ;**********************
                        C ;
                        C ;
0000  00                C CYLMODE DB      00           ; 0 = CYLINDER MODE, 1 = not CYLINDER MODE
0001  00                C DRV     DB      00           ; DRIVE NUMBER
0002  00                C HEAD    DB      00           ; HEAD NUMBER
0003  00                C TRACK   DB      00           ; TRACK NUMBER
0004  00                C SECTOR  DB      00           ; SECTOR NUMBER
                        C                              ;
0005  0000              C SECCNT  DW      0000         ; Number of sectors for I/O
                        C ;
                        C ;
0007  00                C COMSTR  DB      00           ; COMMAND STRING LENGTH
0008  00                C         DB      00           ; COMMAND STRING (max. 9 bytes)
0009  00                C         DB      00           ;
000A  00                C         DB      00           ;
000B  00                C         DB      00           ;
000C  00                C         DB      00           ;
000D  00                C         DB      00           ;
000E  00                C         DB      00           ;
000F  00                C         DB      00           ;
0010  00                C         DB      00           ;
                        C                              ;
0011  00                C ERRBUF  DB      00           ; STATUS BYTE 0
0012  00                C         DB      00           ; STATUS BYTE 1
0013  00                C         DB      00           ; STATUS BYTE 2
0014  00                C         DB      00           ; CYLINDER/TRACK
0015  00                C         DB      00           ; HEAD 0 or HEAD 1
0016  00                C         DB      00           ; SECTOR
0017  00                C         DB      00           ; SECTOR SIZE
```

DSKDRV

```
                         C ;**********************
                         C ;***  DMA COMMANDS  ***
                         C ;**********************
                         C ;
                         C ;
= 0047                   C DMAWRT  EQU    47H          ; WRITE DMA COMMAND
= 004B                   C DMAREAD EQU    4BH          ; READ DMA COMMAND
                         C ;
                         C ;
                         C ;
                         C ;
                         C ;***********************
                         C ;***  DMA VARIABLES  ***
                         C ;***********************
                         C ;
                         C ;
0018  0000               C DMAADDR DW     0000         ; DMA ADDR OFFSET
001A  0000               C         DW     0000         ;          SEGMENT
                         C                             ;
001C  0000               C DMALENG DW     0000         ; DMA LENGTH
001E  00                 C DMAFUNC DB     00           ; DMA FUNCTION
                         C ;
                         C ;
                         C ;
                         C ;************************
                         C ;***  DISK VARIABLES  ***
                         C ;************************
                         C ;
                         C ;
001F  08                 C SECTRK  DB     08           ; SECTORS PER TRACK
0020  40                 C DENSITY DB     40H          ; DOUBLE DENSITY BIT (MFM)
0021  02                 C BYTSEC  DB     02           ; BYTES PER SECTOR (N): 00 - 128 bytes
                         C                             ;                       01 - 256 bytes
                         C                             ;                       02 - 512 bytes
                         C                             ;                          .
                         C                             ;                          .
                         C                             ;                          .
0022  1B                 C GPL     DB     1BH          ; GAP LENGTH
                         C                             ;
0023  F6                 C PATTERN DB     0F6H         ; FORMAT PATTERN
                         C                             ;
0024  05                 C RETRIES DB     05           ; Number of retries
                         C ;
                         C ;
                         C ;
0025  0000               C SSB     DW     0000         ; Special Sector Buffer for BANK conflict
                         C                             ; (not expanded for some CP/M O.S.)
0027  01FF [                       DW     511 DUP (0)  ; Max. possible sector size of FDC controller
            0000
                ]
```

```
C          INCLUDE FLXPIMC.ASM
C  ;       TITLE   FLEX DISK DRIVER PIM (CODE SEGMENT)
C  ;

C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   ROUTINE NAME:      DREAD
C  ;                      DWRITE
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   FUNCTION:          DREAD - low level READ DATA
C  ;                      DWRITE - low level WRITE DATA
C  ;
C  ;
C  ;
C  ;
C  ;   ENTRY VIA:         CALL
C  ;
C  ;
C  ;   ENTRY CONDITIONS:  Following variables are set:
C  ;                      CYLMODE, DRV, HEAD, TRACK, SECTOR,
C  ;                      SECCNT (Number of sectors),
C  ;                      and DMAADDR (SEGMENT and OFFSET)
C  ;
C  ;
C  ;
C  ;   EXIT VIA:          RETURN
C  ;
C  ;
C  ;   EXIT CONDITIONS:   STATUS (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
```

```
                         C
0425                     C  DREAD:                               ;
0425  B1 06              C          MOV    CL,READDAT            ; CL <-- READ DATA COMMAND
0427  C6 06 001E R 47    C          MOV    DMAFUNC,DMAWRT        ; DMAFUNC <-- WRITE DMA COMMAND
042C  EB 08 90           C          JMP    I01                   ;
042F                     C  DWRITE:                              ;
042F  B1 05              C          MOV    CL,WRITDAT            ; CL <-- WRITE DATA COMMAND
0431  C6 06 001E R 4B    C          MOV    DMAFUNC,DMAREAD       ; DMAFUNC <-- READ DMA COMMAND
0436                     C  I01:                                 ;
0436  83 3E 0005 R 00    C          CMP    SECCNT,0              ; Check if an I/O is necessary
043B  75 01              C          JNZ    I02                   ; Jump if necessary
043D  C3                 C          RET                          ; Return if not necessary
043E                     C  I02:                                 ;
                         C                                       ; Check TRACK conflict
043E  B7 00              C          MOV    BH,00                 ; -------------------
0440  8A 1E 001F R       C          MOV    BL,SECTRK             ; BX <-- SECTORS PER TRACK
0444  FE C3              C          INC    BL                    ;
0446  2A 1E 0004 R       C          SUB    BL,SECTOR             ; BX - remaining sectors in track
                         C                                       ;
044A  A0 0000 R          C          MOV    AL,CYLMODE            ; If CYLINDER MODE
044D  0A 06 0002 R       C          OR     AL,HEAD               ; and HEAD 0
0451  75 04              C          JNZ    I03                   ;
0453  02 1E 001F R       C          ADD    BL,SECTRK             ; then add sectors of corresponding track
0457                     C  I03:                                 ;
0457  3B 1E 0005 R       C          CMP    BX,SECCNT             ; Compare remainding sectors with SECCNT
045B  72 04              C          JB     I04                   ; Jump if more than one I/O
045D  8B 1E 0005 R       C          MOV    BX,SECCNT             ;
0461                     C  I04:                                 ; BX - number of sectors fitting in TRACK
                         C                                       ;
                         C                                       ; Check BANK conflict
                         C                                       ; -----------------
0461  A1 001A R          C          MOV    AX,DMAADDR+2          ; AX <-- DMA SEGMENT
0464  D1 E0              C          SHL    AX,1                  ;
0466  D1 E0              C          SHL    AX,1                  ;
0468  D1 E0              C          SHL    AX,1                  ;
046A  D1 E0              C          SHL    AX,1                  ;
046C  03 06 0018 R       C          ADD    AX,DMAADDR            ; AX <-- absolute addr within BANK
0470  F7 D8              C          NEG    AX                    ; AX <-- remaining bytes within BANK
0472  8A 36 0021 R       C          MOV    DH,BYTSEC             ;
0476  B2 00              C          MOV    DL,00                 ; DX <-- sector size
0478  80 FE 00           C          CMP    DH,00                 ;
047B  75 02              C          JNZ    I05                   ;
                         C                                       ;
047D  B2 80              C          MOV    DL,128                ;
047F                     C  I05:                                 ;
047F  8B F2              C          MOV    SI,DX                 ; SI <-- sector size
0481  BA 0000            C          MOV    DX,0000               ; DX <-- 0000
0484  F7 F6              C          DIV    SI                    ; AX <-- number of sectors fitting in BANK
                         C                                       ;
0486  3B C3              C          CMP    AX,BX                 ; Check if we must do Special Sector Handling
0488  72 03              C          JB     I06                   ; Jump if we must
                         C                                       ;
048A  E9 0513 R          C          JMP    I015                  ; Jump around if not
048D                     C  I06:                                 ;
048D  93                 C          XCHG   BX,AX                 ; BX <-- number of sectors fitting in BANK
048E  83 FB 00           C          CMP    BX,00                 ; Check if we must do now Special Sector Handling
0491  74 03              C          JZ     I07                   ; Jump if we must     ---
                         C                                       ;
0493  EB 7E 90           C          JMP    I015                  ; Jump around if not
                         C
```

```
                              C
                              C
0496                          C  I07:                          ;** Special Sector Handling
                              C                                ;** --------------------
0496  83 2E 0005 R 01         C         SUB    SECCNT,01       ;** SECCNT <-- remainding sectors for next I/O
                              C                                ;**
049B  8A 26 0021 R            C         MOV    AH,BYTSEC       ;**
049F  B0 00                   C         MOV    AL,00           ;** AX <-- sector size
04A1  80 FC 00                C         CMP    AH,00           ;**
04A4  75 02                   C         JNZ    I08             ;**
                              C                                ;**
04A6  B0 80                   C         MOV    AL,128          ;**
04A8                          C  I08:                          ;**
04A8  A3 001C R               C         MOV    DMALENG,AX      ;** DMALENG <-- sector size
                              C                                ;**
04AB  80 E1 0F                C         AND    CL,0FH          ;** Clear upper bits
04AE  80 F9 05                C         CMP    CL,WRITDAT      ;** Check if WRITE DATA COMMAND
04B1  75 1B                   C         JNZ    I09             ;** Jump around if not
                              C                                ;*
                              C                                ;*
                              C                                ;*
04B3  51                      C         PUSH   CX              ;* Save CX
04B4  8B 36 0018 R            C         MOV    SI,DMAADDR      ;* SI <-- source offset
04B8  BF 0025 R               C         MOV    DI,OFFSET SSB   ;* DI <-- destination offset
04BB  8B 0E 001C R            C         MOV    CX,DMALENG      ;* CX <-- sector size
04BF  D1 E9                   C         SHR    CX,1            ;* We move WORDS
04C1  FC                      C         CLD                    ;* incrementing
04C2  1E                      C         PUSH   DS              ;* Save DS
04C3  A1 001A R               C         MOV    AX,DMAADDR+2    ;*
04C6  8E D8                   C         MOV    DS,AX           ;* DS <-- SEGMENT of TRANSFER ADDR
04C8  07                      C         POP    ES              ;*
04C9  06                      C         PUSH   ES              ;* ES <-- our SEGMENT of Special Sector Buffer
                              C                                ;*
                              C                                ;* W R I T E  D A T A  C O M M A N D:
04CA  F3/ A5                  C  REP    MOVSW                  ;* Move BANK into Special Sector Buffer
                              C                                ;* ---------------------------------
04CC  1F                      C         POP    DS              ;* Restore DS
04CD  59                      C         POP    CX              ;* Restore CX
                              C                                ;*
                              C                                ;*
                              C                                ;*
04CE                          C  I09:                          ;**
04CE  A1 0018 R               C         MOV    AX,DMAADDR      ;**
04D1  50                      C         PUSH   AX              ;** Save DMA OFFSET
04D2  A1 001A R               C         MOV    AX,DMAADDR+2    ;**
04D5  50                      C         PUSH   AX              ;** Save DMA SEGMENT
                              C                                ;**
04D6  B8 0025 R               C         MOV    AX,OFFSET SSB   ;**
04D9  A3 0018 R               C         MOV    DMAADDR,AX      ;** new OFFSET <-- Special Sector Buffer
04DC  8C D8                   C         MOV    AX,DS           ;**
04DE  A3 001A R               C         MOV    DMAADDR+2,AX    ;** new SEGMENT <-- our SEGMENT
                              C                                ;**
04E1  E8 0535 R               C         CALL   IO              ;** Do I/O
                              C                                ;** ------
04E4  72 03                   C         JC     IO10            ;** Jump if normal termination
04E6  58                      C         POP    AX              ;** else
04E7  58                      C         POP    AX              ;** flush STACK
04E8  C3                      C         RET                    ;** and return with bad status in ERRBUF
04E9                          C  IO10:                         ;**
04E9  58                      C         POP    AX              ;**
```

```
04EA  A3 001A R           C          MOV    DMAADDR+2,AX   ;** Restore DMA SEGMENT
04ED  8E C0               C          MOV    ES,AX          ;**
04EF  58                  C          POP    AX             ;**
04F0  A3 0018 R           C          MOV    DMAADDR,AX     ;** Restore DMA OFFSET
                          C                                ;**
                          C                                ;**
                          C                                ;**
04F3  80 E1 0F            C          AND    CL,0FH         ;** Clear upper bits
04F6  80 F9 06            C          CMP    CL,READDAT     ;** Check if READ DATA COMMAND
04F9  75 12               C          JNZ    I011           ;** Jump around if not
                          C                                ;*
04FB  51                  C          PUSH   CX             ;* Save CX
04FC  BE 0025 R           C          MOV    SI,OFFSET SSB  ;* SI <-- source offset
04FF  8B 3E 0018 R        C          MOV    DI,DMAADDR     ;* DI <-- destination offset
0503  8B 0E 001C R        C          MOV    CX,DMALENG     ;* CX.<-- sector size
0507  D1 E9               C          SHR    CX,1           ;* We move WORDS
0509  FC                  C          CLD                   ;* incrementing
                          C                                ;* R E A D  D A T A  C O M M A N D:
050A  F3/ A5              C  REP     MOVSW                 ;* Move Special Sector Buffer into BANK
                          C                                ;* ----------------------------------
050C  59                  C          POP    CX             ;* Restore CX
                          C                                ;*
                          C                                ;*
050D                      C  I011:                         ;**
050D  BB 0001             C          MOV    BX,0001        ;** BX - number of sectors of previous I/O
0510  EB 62 90            C          JMP    I030           ;** Jump to update variables for next I/O
                          C
```

```
                         C
                         C
0513                     C  I015:                    ; BX - number of sectors for I/O
0513  53                 C         PUSH   BX         ; ----------------------------
0514  29 1E 0005 R       C         SUB    SECCNT,BX  ; SECCNT <-- remainding sectors for next I/O
                         C                           ;
0518  8A 26 0021 R       C         MOV    AH,BYTSEC  ;
051C  B0 00              C         MOV    AL,00      ; AX <-- sector size
051E  80 FC 00           C         CMP    AH,00      ;
0521  75 02              C         JNZ    I016       ;
                         C                           ;
0523  B0 80              C         MOV    AL,128     ;
0525                     C  I016:                    ;
0525  F7 E3              C         MUL    BX         ; * sectors for I/O gives DMA LENGTH
0527  A3 001C R          C         MOV    DMALENG,AX ; DMALENG <-- DMA LENGTH
                         C                           ;
052A  E8 0535 R          C         CALL   IO         ; Dc I/O
                         C                           ; ------
052D  72 02              C         JC     I017       ; Jump if normal termination
052F  58                 C         POP    AX         ; else flush STACK
0530  C3                 C         RET               ; and return with bad status in ERRBUF
0531                     C  I017:                    ;
0531  5B                 C         POP    BX         ; BX - number of sectors of previous I/O
0532  EB 40 90           C         JMP    I030       ; Jump to update variables for next I/O
                         C
```

```
                          C
                          C
0535                      C  IO:                          ; Disk I/O
                          C                               ; --------
0535  A0 0024 R           C        MOV    AL,RETRIES      ; AL <-- retry counter
0538                      C  IO20:                        ;
0538  50                  C        PUSH   AX              ; Save retry counter
0539  E8 06A4 R           C        CALL   SETUP9          ; Set up COMMAND STRING and DMA
053C  E8 0732 R           C        CALL   XWAIT           ; Send COMMAND STRING to FDC
053F  E8 0750 R           C        CALL   GETBYT          ; Get STATUS BYTES
0542  58                  C        POP    AX              ; Restore retry counter
                          C                               ;
0543  F6 06 0011 R C0     C        TEST   ERRBUF,0C0H     ; Test for normal termination
0548  75 02               C        JNZ    IO21            ; Jump on error
054A  F9                  C        STC                    ; Set status flag
054B  C3                  C        RET                    ; Return with good status
                          C                               ;
                          C                               ;
                          C                               ;
054C                      C  IO21:                        ;
054C  F6 06 0011 R 08     C        TEST   ERRBUF,08H      ; Test for 'NOT READY'
0551  74 02               C        JZ     IO22            ;
0553  F8                  C        CLC                    ; Set status flag
0554  C3                  C        RET                    ; Return immediately if disk 'NOT READY'
0555                      C  IO22:                        ;
0555  F6 06 0012 R 02     C        TEST   ERRBUF+1,02H    ; Test for 'WRITE PROTECTED'
055A  74 02               C        JZ     IO23            ;
055C  F8                  C        CLC                    ; Set status flag
055D  C3                  C        RET                    ; Return immediately if 'WRITE PROTECTED'
055E                      C  IO23:                        ;
055E  F6 06 0011 R 80     C        TEST   ERRBUF,80H      ; Test for 'INVALID COMMAND'
0563  74 02               C        JZ     IO24            ;
0565  F8                  C        CLC                    ; Set status flag
0566  C3                  C        RET                    ; Return immediately if 'INVALID COMMAND'
0567                      C  IO24:                        ;
0567  FE C8               C        DEC    AL              ; Decrement retry counter
0569  74 07               C        JZ     IO25            ; Jump to exit with bad status
                          C                               ;
056B  50                  C        PUSH   AX              ; Save retry counter
056C  E8 05FA R           C        CALL   DREST           ; Do a low level RESTORE
056F  58                  C        POP    AX              ; Restore retry counter
0570  EB C6               C        JMP    IO20            ; Do retries
                          C                               ;
                          C                               ;
                          C                               ;
0572                      C  IO25:                        ;
0572  F8                  C        CLC                    ; Set status flag
0573  C3                  C        RET                    ; Return with bad status
```

```
                        C
                        C
0574                    C  I030:                        ; Update variables for next I/O
                        C                               ; -------------------------------
                        C                               ; BX - number of sectors of previous I/O
                        C                               ;
0574  83 3E 0005 R 00   C        CMP    SECCNT,0        ; Check if another I/O is necessary
0579  75 01             C        JNZ    I031            ; Jump if necessary
057B  C3                C        RET                    ; Return if not necessary
057C                    C  I031:                        ;
057C  8B 16 001C R      C        MOV    DX,DMALENG      ; DX <-- previous DMA LENGTH
0580  D1 EA             C        SHR    DX,1            ;
0582  D1 EA             C        SHR    DX,1            ;
0584  D1 EA             C        SHR    DX,1            ;
0586  D1 EA             C        SHR    DX,1            ; DX - previous DMA LENGTH in paragraphs
0588  01 16 001A R      C        ADD    WORD PTR DMAADDR+2,DX ; Update DMAADDR (SEGMENT)
                        C                               ;
058C  00 1E 0004 R      C        ADD    SECTOR,BL       ; Update SECTOR variable
0590  A0 001F R         C        MOV    AL,SECTRK       ; AL <-- sectors per track
                        C                               ;
0593  80 3E 0000 R 00   C        CMP    CYLMODE,00      ; Check if CYLINDER MODE
0598  74 29             C        JZ     I034            ; Jump if CYLINDER MODE
                        C                               ;
                        C                               ; Not CYLINDER MODE
                        C                               ; -----------------
059A  3A 06 0004 R      C        CMP    AL,SECTOR       ; Check for legal SECTOR variable
059E  72 03             C        JB     I032            ; Jump if not legal
                        C                               ;
05A0  E9 0436 R         C        JMP    I01             ; Do next I/O
05A3                    C  I032:                        ;
05A3  C6 06 0004 R 01   C        MOV    SECTOR,1        ; Set SECTOR to begin of track
05A8  80 3E 0003 R 27   C        CMP    TRACK,39        ; Check if side 1 is full
05AD  74 07             C        JZ     I033            ; Jump if full
                        C                               ;
05AF  FE 06 0003 R      C        INC    TRACK           ; Increment TRACK
05B3  E9 0436 R         C        JMP    I01             ; Do next I/O
05B6                    C  I033:                        ;
05B6  C6 06 0002 R 01   C        MOV    HEAD,1          ; If side 1 is full
05BB  C6 06 0003 R 00   C        MOV    TRACK,0         ; then initialize for side 2
05C0  E9 0436 R         C        JMP    I01             ; Do next I/O
05C3                    C  I034:                        ;
                        C                               ; CYLINDER MODE
                        C                               ; -------------
05C3  3A 06 0004 R      C        CMP    AL,SECTOR       ; Check for legal SECTOR variable
05C7  72 03             C        JB     I035            ; Jump if not legal
                        C                               ;
05C9  E9 0436 R         C        JMP    I01             ; Do next I/O
05CC                    C  I035:                        ;
05CC  80 3E 0002 R 01   C        CMP    HEAD,1          ; Check if cylinder is full
05D1  74 16             C        JZ     I036            ; Jump if full
                        C                               ;
05D3  D0 E0             C        SHL    AL,1            ; AL <-- sectors per cylinder
05D5  3A 06 0004 R      C        CMP    AL,SECTOR       ; Check if cylinder is full
05D9  72 0E             C        JB     I036            ; Jump if full
                        C                               ;
05DB  D0 E8             C        SHR    AL,1            ; AL <-- sectors per track
05DD  28 06 0004 R      C        SUB    SECTOR,AL       ; Set SECTOR variable within
05E1  C6 06 0002 R 01   C        MOV    HEAD,1          ; corresponding track with HEAD 1
05E6  E9 0436 R         C        JMP    I01             ; Do next I/O
05E9                    C  I036:                        ;
05E9  FE 06 0003 R      C        INC    TRACK           ; Increment TRACK
05ED  C6 06 0002 R 00   C        MOV    HEAD,0          ; Set HEAD 0
05F2  C6 06 0004 R 01   C        MOV    SECTOR,1        ; Set SECTOR to begin of cylinder
05F7  E9 0436 R         C        JMP    I01             ; Do next I/O
                        C
```

DSKDRV

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       DREST
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Low level RESTORE
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   DRV variable is set
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    CL - preserved
C  ;                        STATUS (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                           C
05FA                       C DREST:                         ;
05FA B4 02                 C         MOV     AH,02          ; Special retry for CP/M
                           C                                ;
05FC                       C DREST1:                        ; Set up COMMAND STRING
                           C                                ; ---------------------
05FC C6 06 0007 R 02       C         MOV     COMSTR,2       ; COMMAND STRING <-- LENGTH 2
0601 C6 06 0008 R 07       C         MOV     COMSTR+1,RESTORE;               <-- RESTORE COMMAND
0606 A0 0001 R             C         MOV     AL,DRV         ;
0609 A2 0009 R             C         MOV     COMSTR+2,AL    ;               <-- DRIVE NUMBER
                           C                                ;
060C 50                    C         PUSH    AX             ; Save retry counter
060D E8 0732 R             C         CALL    XWAIT          ; Send COMMAND STRING to FDC
0610                       C DREST2:                        ;
0610 E4 13                 C         IN      AL,SYSSTA      ; Wait on interrupt
0612 24 08                 C         AND     AL,08          ; Test DISK INTERRUPT BIT
0614 74 FA                 C         JZ      DREST2         ; Jump if no interrupt
                           C                                ;
0616 E8 0674 R             C         CALL    DSIS           ; Reset interrupt via low level SENSE
                           C                                ; INTERRUPT STATUS
                           C                                ;
0619 58                    C         POP     AX             ; Restore retry counter
061A F6 06 0011 R C0       C         TEST    ERRBUF,0C0H    ; Test for normal termination
061F 74 04                 C         JZ      DREST3         ; Jump if normal termination
                           C                                ;
0621 FE CC                 C         DEC     AH             ; Decrement retry counter
0623 75 D7                 C         JNZ     DREST1         ; Do special retry !
0625                       C DREST3:                        ; Reason: MOTOR OFF & RESTORE in CP/M
0625 C3                    C         RET                    ;
```

DSKDRV

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       DSEEK
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Low level SEEK A TRACK
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   Following variables are set:
C  ;                        DRV, HEAD, and TRACK
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    CL - preserved
C  ;                        STATUS (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                          C
0626                      C  DSEEK:                      ; Set up COMMAND STRING
                          C                              ; ------------------
0626  C6 06 0007 R 03     C        MOV   CONSTR,3        ; COMMAND STRING <-- LENGTH 3
062B  C6 06 0008 R 0F     C        MOV   CONSTR+1,SEEKTRK;                <-- SEEK COMMAND
0630  A0 0002 R           C        MOV   AL,HEAD         ;
0633  D0 E0               C        SHL   AL,1            ;
0635  D0 E0               C        SHL   AL,1            ;
0637  0A 06 0001 R        C        OR    AL,DRV          ;
063B  A2 0009 R           C        MOV   CONSTR+2,AL     ;                <-- DRIVE & HEAD
063E  A0 0003 R           C        MOV   AL,TRACK        ;
0641  A2 000A R           C        MOV   CONSTR+3,AL     ;                <-- TRACK
                          C                              ;
0644  E8 0732 R           C        CALL  XWAIT           ; Send COMMAND STRING to FDC
0647                      C  DSEEK1:                     ;
0647  E4 13               C        IN    AL,SYSSTA       ; Wait on interrupt
0649  24 08               C        AND   AL,08           ; Test DISK INTERRUPT BIT
064B  74 FA               C        JZ    DSEEK1          ; jump if no interrupt
                          C                              ;
064D  E8 0674 R           C        CALL  DSIS            ; Reset interrupt via low level SENSE
                          C.                             ; INTERRUPT STATUS
0650  C3                  C        RET                   ;
```

```
C
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   ROUTINE NAME:       DREADID
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   FUNCTION:           Low level READ ID
C  ;                       (Used to get SECTOR SIZE)
C  ;
C  ;
C  ;
C  ;
C  ;   ENTRY VIA:          CALL
C  ;
C  ;
C  ;   ENTRY CONDITIONS:   Following variables are set:
C  ;                       DRV and HEAD
C  ;
C  ;
C  ;
C  ;
C  ;   EXIT VIA:           RETURN
C  ;
C  ;
C  ;   EXIT CONDITIONS:    STATUS and BYTES PER SECTOR (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
```

```
                          C
0651                      C  DREADID:                         ; Set up COMMAND STRING
                          C                                   ; -------------------
 0651  C6 06 0007 R 02    C          MOV     CONSTR,2         ; COMMAND STRING <-- LENGTH 2
 0656  B0 0A              C          MOV     AL,IDREAD        ;
 0658  0A 06 0020 R       C          OR      AL,DENSITY       ;
 065C  A2 0008 R          C          MOV     CONSTR+1,AL      ;              <-- READ ID COMMAND & DENSITY
 065F  A0 0002 R          C          MOV     AL,HEAD          ;
 0662  D0 E0              C          SHL     AL,1             ;
 0664  D0 E0              C          SHL     AL,1             ;
 0666  0A 06 0001 R       C          OR      AL,DRV           ;
 066A  A2 0009 R          C          MOV     CONSTR+2,AL      ;              <-- DRIVE & HEAD
                          C                                   ;
 066D  E8 0732 R          C          CALL    XWAIT            ; Send COMMAND STRING to FCB
 0670  E8 0750 R          C          CALL    GETBYT           ; Get STATUS BYTES (sector size)
 0673  C3                 C          RET                      ;
```

DSKDRV

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       DSIS
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Low level SENSE INTERRUPT STATUS
C  ;                        (used to reset interrupt)
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   NONE
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    STATUS (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                          C
0674                      C  DSIS:                         ; Set up COMMAND STRING
                          C                                ; ---------------------
0674  C6 06 0007 R 01     C        MOV    CONSTR,1         ; COMMAND STRING <-- LENGTH 1
0679  C6 06 0008 R 08     C        MOV    CONSTR+1,FDCSIS  ;                <-- FDCSIS COMMAND
                          C                                ;
067E  E8 0732 R           C        CALL   XWAIT            ; Send COMMAND STRING to FDC
0681  E8 0750 R           C        CALL   GETBYT           ; Get STATUS BYTES
0684  C3                  C        RET                     ;
```

DSKDRV

```
C
C  ;********************************************************************************
C  ;********************************************************************************
C  ;********************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       DFORMAT
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Low level FORMAT A TRACK
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   Following variables are set:
C  ;                        DRV, HEAD, TRACK, PATTERN
C  ;                        and DMAADDR (SEGMENT and OFFSET)
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    STATUS (returned in ERRBUF)
C  ;
C  ;
C  ;
C  ;********************************************************************************
C  ;********************************************************************************
C  ;********************************************************************************
```

```
                        C
0685                    C  DFORMAT:                        ;
0685  B1 0D             C          MOV     CL,WRITFMT      ; CL <-- FORMAT COMMAND
0687  C6 06 001E R 4B   C          MOV     DMAFUNC,DMAREAD ; DMAFUNC <-- READ DMA COMMAND
068C  B7 00             C          MOV     BH,00           ;
068E  8A 1E 001F R      C          MOV     BL,SECTRK       ;
0692  D1 E3             C          SHL     BX,1            ;
0694  D1 E3             C          SHL     BX,1            ;
0696  89 1E 001C R      C          MOV     DMALENG,BX      ; DMALENG <-- DMA LENGTH (SECTRK*4)
                        C                                  ;
069A  E8 06F9 R         C          CALL    SETUP6          ; Set up COMMAND STRING and DMA
069D  E8 0732 R         C          CALL    XWAIT           ; Send COMMAND STRING to FDC
06A0  E8 0750 R         C          CALL    GETBYT          ; Get STATUS BYTES
06A3  C3                C          RET                     ;
```

DSKDRV

```
C
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       SETUP9
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Set up (9 byte) COMMAND STRING and DMA
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   CL - COMMAND
C  ;                        Following variables are set:
C  ;                        DMAADDR (SEGMENT and OFFSET)
C  ;                        DMALENG and DMAFUNC
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    NONE
C  ;
C  ;
C  ;
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
```

```
                          C
06A4                      C  SETUP9:                          ;
06A4  E8 0626 R           C          CALL    DSEEK            ; First do low level SEEK A TRACK
                          C                                   ;
06A7  C6 06 0007 R 09     C          MOV     COMSTR,9         ; COMMAND STRING <-- LENGTH 9
06AC  0A 0E 0020 R        C          OR      CL,DENSITY       ;
06B0  80 3E 0000 R 00     C          CMP     CYLMODE,00       ;
06B5  75 03               C          JNZ     SET1             ;
                          C                                   ;
06B7  80 C9 80            C          OR      CL,80H           ;
06BA                      C  SET1:                            ;
06BA  88 0E 0008 R        C          MOV     COMSTR+1,CL      ;                <-- FUNCTION & DENSITY & MT
06BE  A0 0002 R           C          MOV     AL,HEAD          ;
06C1  D0 E0               C          SHL     AL,1             ;
06C3  D0 E0               C          SHL     AL,1             ;
06C5  0A 06 0001 R        C          OR      AL,DRV           ;
06C9  A2 0009 R           C          MOV     COMSTR+2,AL      ;                <-- DRIVE & HEAD
06CC  A0 0003 R           C          MOV     AL,TRACK         ;
06CF  A2 000A R           C          MOV     COMSTR+3,AL      ;                <-- TRACK
06D2  A0 0002 R           C          MOV     AL,HEAD          ;
06D5  A2 000B R           C          MOV     COMSTR+4,AL      ;                <-- HEAD
06D8  A0 0004 R           C          MOV     AL,SECTOR        ;
06DB  A2 000C R           C          MOV     COMSTR+5,AL      ;                <-- SECTOR
06DE  A0 0021 R           C          MOV     AL,BYTSEC        ;
06E1  A2 000D R           C          MOV     COMSTR+6,AL      ;                <-- BYTES PER SECTOR
06E4  A0 001F R           C          MOV     AL,SECTRK        ;
06E7  A2 000E R           C          MOV     COMSTR+7,AL      ;                <-- SECTORS PER TRACK
06EA  A0 0022 R           C          MOV     AL,GPL           ;
06ED  A2 000F R           C          MOV     COMSTR+8,AL      ;                <-- GAP LENGTH
06F0  C6 06 0010 R FF     C          MOV     COMSTR+9,0FFH    ;                <-- DTL
                          C                                   ;
06F5  E8 0779 R           C          CALL    DMA              ; Initialize DMA
06F8  C3                  C          RET                      ;
```

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   ROUTINE NAME:       SETUP6
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;   FUNCTION:           Set up (6 byte) COMMAND STRING and DMA
C  ;
C  ;
C  ;
C  ;
C  ;   ENTRY VIA:          CALL
C  ;
C  ;
C  ;   ENTRY CONDITIONS:   CL - (FORMAT) COMMAND
C  ;                       Following variables are set:
C  ;                       DMAADDR (SEGMENT and OFFSET)
C  ;                       DMALENG and DMAFUNC
C  ;
C  ;
C  ;
C  ;
C  ;   EXIT VIA:           RETURN
C  ;
C  ;
C  ;   EXIT CONDITIONS:    NONE
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                              C
06F9                          C  SETUP6:                         ;
06F9  E8 0626 R               C          CALL    DSEEK           ; First do low level SEEK A TRACK
                              C                                  ;
06FC  C6 06 0007 R 06         C          MOV     COMSTR,6        ; COMMAND STRING <-- LENGTH 6
0701  0A 0E 0020 R            C          OR      CL,DENSITY      ;
0705  88 0E 0008 R            C          MOV     COMSTR+1,CL     ;                <-- FUNCTION & DENSITY
0709  A0 0002 R               C          MOV     AL,HEAD         ;
070C  D0 E0                   C          SHL     AL,1            ;
070E  D0 E0                   C          SHL     AL,1            ;
0710  0A 06 0001 R            C          OR      AL,DRV          ;
0714  A2 0009 R               C          MOV     COMSTR+2,AL     ;                <-- DRIVE & HEAD
0717  A0 0021 R               C          MOV     AL,BYTSEC       ;
071A  A2 000A R               C          MOV     COMSTR+3,AL     ;                <-- BYTES PER SECTOR
071D  A0 001F R               C          MOV     AL,SECTRK       ;
0720  A2 000B R               C          MOV     COMSTR+4,AL     ;                <-- SECTORS PER TRACK
0723  C6 06 000C R 50         C          MOV     COMSTR+5,50H    ;                <-- GAP LENGTH
0728  A0 0023 R               C          MOV     AL,PATTERN      ;
072B  A2 000D R               C          MOV     COMSTR+6,AL     ;                <-- PATTERN
                              C                                  ;
072E  E8 0779 R               C          CALL    DMA             ; Initialize DMA
0731  C3                      C          RET                     ;
```

DSKDRV

```
C
C  ;****************************************************************************
C  ;****************************************************************************
C  ;****************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:      XWAIT
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:          Send COMMAND STRING to FDC
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:         CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:  NONE
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:          RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:   CL - preserved
C  ;
C  ;
C  ;
C  ;****************************************************************************
C  ;****************************************************************************
C  ;****************************************************************************
```

```
                         C
0732                     C  XWAIT:                       ;
0732  E8 0767 R          C         CALL   MOTORCK        ; SWITCH MOTOR ON
                         C                               ;
0735  8A 2E 0007 R       C         MOV    CH,COMSTR      ; CH <-- COMMAND STRING LENGTH
0739  BB 0007 R          C         MOV    BX,OFFSET COMSTR; BX <-- Addr of COMMAND STRING
073C                     C  XWAIT1:                      ;
073C  43                 C         INC    BX             ;
073D  E8 0760 R          C         CALL   FDCRDY         ; Wait until FDC is ready
0740  8A 07              C         MOV    AL,BYTE PTR [BX]; AL <-- next COMMAND STRING byte
0742  E6 51              C         OUT    DCOMD,AL       ; Send byte to FDC
0744  FE CD              C         DEC    CH             ; Decrement counter
0746  75 F4              C         JNZ    XWAIT1         ; Loop until last byte
                         C                               ;
0748  E8 0760 R          C         CALL   FDCRDY         ; Wait until FDC is ready
                         C                               ;
074B  B0 07              C         MOV    AL,07          ;
074D  E6 2A              C         OUT    DMAMB,AL       ; Disable DMA CHANNEL
074F  C3                 C         RET                   ;
                         C
```

```
C
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       GETBYT
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Get STATUS BYTES into ERRBUF
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   NONE
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    NONE
C  ;
C  ;
C  ;
C  ;******************************************************************************
C  ;******************************************************************************
C  ;******************************************************************************
```

```
                          C
0750                      C  GETBYT:                          ;
0750  BB 0011 R           C          MOV    BX,OFFSET ERRBUF; BX <-- Addr of ERROR BUFFER
0753                      C  GETBYT1:                         ;
0753  E4 51               C          IN     AL,FDCRA        ; Read STATUS BYTE from FDC
0755  88 07               C          MOV    BYTE PTR [BX],AL; into ERROR BUFFER
0757  43                  C          INC    BX              ;
0758  E8 076D R           C          CALL   FDCRDY          ; Wait until FDC is ready
075B  A8 40               C          TEST   AL,40H          ; Check if FDC has another byte
075D  75 F4               C          JNZ    GETBYT1         ; Jump to fetch next byte
075F  C3                  C          RET                    ;
```

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       FDCRDY
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Wait until FDC is ready
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   NONE
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    NONE
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                        C
0760                    C  FDCRDY:                          ;
0760  E4 50             C          IN      AL,DSTAT         ; AL <-- DISK STATUS
0762  A8 80             C          TEST    AL,80H           ; Test MASTER REQUEST BIT
0764  74 FA             C          JZ      FDCRDY           ; Jump if no MASTER REQUEST (means: in execution)
                        C                                   ;
0766  C3                C          RET                      ; Return if FDC is ready
```

DSKDRV

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       MOTORCK
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           Check if motor is on
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   NONE
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    Motor is on
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                        C
0767                    C  MOTORCK:                            ;
0767  E4 13             C          IN      AL,SYSSTA           ; AL <-- SYSTEM STATUS
0769  24 01             C          AND     AL,01               ; Test DISK MOTOR ON BIT
076B  E6 14             C          OUT     MOTORON,AL          ; Switch motor on
076D  75 01             C          JNZ     MOTORCK1            ;
076F  C3                C          RET                         ; Return if motor was on
0770                    C  MOTORCK1:                           ;
0770  BB FFFF           C          MOV     BX,0FFFFH           ; Wait some time if motor was off
0773                    C  MOTORCK2:                           ;
0773  D4 0A             C          AAM                         ; (83)
0775  4B                C          DEC     BX                  ; ( 2)
0776  75 FB             C          JNZ     MOTORCK2            ; ( 8)  = 93 CLOCKS * FFFF = 1 sec
                        C                                      ;
0778  C3                C          RET                         ;
```

```
C
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    ROUTINE NAME:       DMA
C  ;
C  ;
C  ;
C  ;
C  ;
C  ;    FUNCTION:           DMA routines
C  ;
C  ;
C  ;
C  ;
C  ;    ENTRY VIA:          CALL
C  ;
C  ;
C  ;    ENTRY CONDITIONS:   Following variables are set:
C  ;                        DMAADDR (SEGMENT and OFFSET)
C  ;                        DMALENG and DMAFUNC
C  ;
C  ;
C  ;
C  ;
C  ;    EXIT VIA:           RETURN
C  ;
C  ;
C  ;    EXIT CONDITIONS:    NONE
C  ;
C  ;
C  ;
C  ;*******************************************************************************
C  ;*******************************************************************************
C  ;*******************************************************************************
```

```
                           C
0779                       C  DMA:                          ;
0779  A0 001E R            C         MOV    AL,DMAFUNC      ; DMAFUNC <-- DMA FUNCTION
077C  E6 2B                C         OUT    DMAMO,AL        ; OUT MODE
                           C                                ;
077E  A1 001A R            C         MOV    AX,DMAADDR+2    ; AX <-- DMA SEGMENT
0781  D1 E0                C         SHL    AX,1            ;
0783  D1 E0                C         SHL    AX,1            ;
0785  D1 E0                C         SHL    AX,1            ;
0787  D1 E0                C         SHL    AX,1            ;
0789  03 06 0018 R         C         ADD    AX,DMAADDR      ; AX <-- absolute addr within BANK
078D  E6 26                C         OUT    COAD,AL         ; OUT DMA ADDR low
078F  8A C4                C         MOV    AL,AH           ;
0791  E6 26                C         OUT    COAD,AL         ; OUT DMA ADDR high
                           C                                ;
0793  A1 001C R            C         MOV    AX,DMALENG      ; AX <-- DMA LENGTH
0796  48                   C         DEC    AX              ;
0797  E6 27                C         OUT    COTC,AL         ; OUT DMA LENGTH low
0799  8A C4                C         MOV    AL,AH           ;
079B  E6 27                C         OUT    COTC,AL         ; OUT DMA LENGTH high
                           C                                7?
079D  B6 00                C         MOV    DH,00           ;
079F  B2 E0                C         MOV    DL,BANK         ; DX - BANK 0 initialisation
07A1  80 D2 00             C         ADC    DL,00           ; DX - next BANK if SEGMENT + OFFSET > 64K
                           C                                ;
07A4  A1 001A R            C         MOV    AX,DMAADDR+2    ; AX <-- DMA SEGMENT
07A7  D0 EC                C         SHR    AH,1            ;
07A9  D0 EC                C         SHR    AH,1            ;
07AB  D0 EC                C         SHR    AH,1            ;
07AD  D0 EC                C         SHR    AH,1            ;
07AF  02 D4                C         ADD    DL,AH           ; DX <-- BANK SELECT PORT
                           C                                ;
07B1  EE                   C         OUT    DX,AL           ; SELECT BANK
                           C                                ; -----------
07B2  B0 03                C         MOV    AL,03           ;
07B4  E6 2A                C         OUT    DMAMB,AL        ; Enable FDC CHANNEL
07B6  C3                   C         RET                    ;
```

```
07B7                          DSKTBL:
07B7  093C R                          DW      DSK_INIT        ; 0  - INIT
07B9  1A                              DB      26              ; Length of drive request structure
                                                              ;
07BA  0965 R                          DW      MEDIAC          ; 1  - MEDIA CHECK
07BC  0F                              DB      15              ; Length of drive request structure
                                                              ;
07BD  0995 R                          DW      GET_BPB         ; 2  - Build BPB
07BF  16                              DB      22              ; Length of drive request structure
                                                              ;
07C0  08B0 R                          DW      CMDERR          ; 3  - IOCTL INPUT (currently returns error)
07C2  16                              DB      22              ; Length of drive request structure
                                                              ;
07C3  0425 R                          DW      DREAD           ; 4  - READ
07C5  16                              DB      22              ; Length of drive request structure
                                                              ;
07C6  0000                            DW      0000            ; 5  - NON DESTRUCTIVE INPUT (char. devices)
07C8  00                              DB      00              ; Length of drive request structure
                                                              ;
07C9  0000                            DW      0000            ; 6  - INPUT STATUS (char. devices)
07CB  00                              DB      00              ; Length of drive request structure
                                                              ;
07CC  0000                            DW      0000            ; 7  - INPUT FLUSH (char. devices)
07CE  00                              DB      00              ; Length of drive request structure
                                                              ;
07CF  042F R                          DW      DWRITE          ; 8  - WRITE
07D1  16                              DB      22              ; Length of drive request structure
                                                              ;
07D2  0A36 R                          DW      DVERIFY         ; 9  - WRITE WITH VERIFY
07D4  16                              DB      22              ; Length of drive request structure
                                                              ;
07D5  0000                            DW      0000            ; 10 - OUTPUT STATUS (char. devices)
07D7  00                              DB      00              ; Length of drive request structure
                                                              ;
07D8  0000                            DW      0000            ; 11 - OUTPUT FLUSH (char. devices)
07DA  00                              DB      00              ; Length of drive request structure
                                                              ;
07DB  08B0 R                          DW      CMDERR          ; 12 - IOCTL OUTPUT (currently returns error)
07DD  16                              DB      22              ; Length of drive request structure
```

```
                           ;
                           ; Disk interrupt routine for processing I/O packets.
                           ;

07DE                       DSK_INT:

07DE  56                          PUSH   SI
07DF  50                          PUSH   AX              ;Save all nessacary registers.
07E0  51                          PUSH   CX
07E1  52                          PUSH   DX
07E2  57                          PUSH   DI
07E3  55                          PUSH   BP
07E4  1E                          PUSH   DS
07E5  06                          PUSH   ES
07E6  53                          PUSH   BX

07E7  0E                          PUSH   CS              ;
07E8  1F                          POP    DS              ; Set DATA SEGMENT to CODE SEGMENT
                                                         ;
07E9  C4 1E 0000 E                LES    BX,[PTRSAV]     ; Retrieve pointer to I/O data packet

07ED  26: 8B 47 12                MOV    AX,ES:[BX.COUNT]; AX <-- Sector count
07F1  A3 0005 R                   MOV    SECCNT,AX       ; Fill PIM.SECCNT variable
                                                         ;
07F4  26: 8B 47 0E                MOV    AX,ES:[BX.TRANS]; AX <-- Transfer address (OFFSET)
07F8  A3 0018 R                   MOV    DMAADDR,AX      ; Fill PIM.DMAADDR variable
07FB  26: 8B 47 10                MOV    AX,ES:[BX.TRANS+2] ; AX <-- Transfer address (SEGMENT)
07FF  A3 001A R                   MOV    DMAADDR+2,AX    ; Fill PIM.DMAADDR variable
                                                         ;
0802  26: 8B 47 14                MOV    AX,ES:[BX.START]; AX <-- Start sector
                                                         ;
0806  C6 06 001F R 08             MOV    SECTRK,8        ; SECTRK <-- 8 for 'FE' and 'FF' disks
080B  26: 80 7F 0D FE             CMP    ES:[BX.MEDIA],0FEH ;
0810  73 05                       JAE    INT0            ; Jump if 'FE' or 'FF' disk
                                                         ;
0812  C6 06 001F R 09             MOV    SECTRK,9        ; SECTRK <-- 9 for 'FC' and 'FD' disks
0817                       INT0:                         ;
0817  B5 00                       MOV    CH,00           ;
0819  8A 0E 001F R                MOV    CL,SECTRK       ; CX <-- SECTORS PER TRACK
081D  C6 06 0000 R 01             MOV    CYLMODE,01      ; Fill PIM.CYLINDER MODE variable
0822  26: 80 7F 0D FE             CMP    ES:[BX.MEDIA],0FEH ;
0827  74 0E                       JZ     INT1            ; Jump if single sided 'FE' disk
0829  26: 80 7F 0D FC             CMP    ES:[BX.MEDIA],0FCH ;
082E  74 07                       JZ     INT1            ; Jump if single sided 'FC' disk
                                                         ;
0830  03 C9                       ADD    CX,CX           ; CX <-- SECTORS PER CYLINDER
0832  C6 06 0000 R 00             MOV    CYLMODE,00      ; Fill PIM.CYLINDER MODE variable
                                                         ; for double sided 'FD' and 'FF' disks
0837                       INT1:                         ;
0837  BA 0000                     MOV    DX,0000         ; DX <-- 0000
083A  F7 F1                       DIV    CX              ; AX - track
083C  42                          INC    DX              ; DX - sector (MS-DOS starts with log sector 0)
083D  A2 0003 R                   MOV    TRACK,AL        ; Fill PIM.TRACK variable
                                                         ;
0840  3A 16 001F R                CMP    DL,SECTRK       ; Test for side 0 or side 1
0844  C6 06 0002 R 00             MOV    HEAD,00         ; Fill PIM.HEAD variable
0849  88 16 0004 R                MOV    SECTOR,DL       ; Fill PIM.SECTOR variable
084D  76 00                       JBE    INT2            ; Jump if side 0
```

```
                                                        ;
084F  2A 16 001F R              SUB    DL,SECTRK        ; DL - Sector within track
0853  C6 06 0002 R 01           MOV    HEAD,01          ; Fill PIM.HEAD variable
0858  88 16 0004 R              MOV    SECTOR,DL        ; Fill PIM.SECTOR variable
085C                      INT2:                         ;
085C  26: 8A 4F 01              MOV    CL,ES:[BX.UNIT]  ; CL <-- Unit code
0860  3A 0E 0000 E              CMP    CL,DRVMAX        ; Test if unit is available
0864  76 03                     JBE    INT3             ; Jump if unit is available
0866  E9 0000 E                 JMP    ERROR_1          ; Jump if 'UNKNOWN UNIT'
0869                      INT3:                         ;
0869  88 0E 0001 R              MOV    DRV,CL           ; Fill PIM.DRV variable
                                                        ;
086D  26: 8A 47 02              MOV    AL,ES:[BX.CMD]   ; AL <-- Command code
0871  3C 00                     CMP    AL,00            ; Test if DISK INIT
0873  74 0C                     JZ     INT4             ; If so, jump around next test
                                                        ;
0875  26: 8A 6F 00              MOV    CH,ES:[BX.MEDIA]; CH <-- Media descriptor
0879  80 FD FC                  CMP    CH,0FCH          ; Test if media is well-known
087C  73 03                     JAE    INT4             ; Jump if media is well-known
087E  E9 0000 E                 JMP    ERROR_7          ; Jump if 'UNKNOWN MEDIA'
0881                      INT4:                         ;
0881  BE 07B7 R                 MOV    SI,OFFSET DSKTBL; SI <-- addr of disk-table
0884  B4 00                     MOV    AH,00            ; AX <-- Command code (see above)
0886  03 F0                     ADD    SI,AX            ;
0888  03 F0                     ADD    SI,AX            ; Compute entry in disk-table
088A  03 F0                     ADD    SI,AX            ;
                                                        ;
088C  3C 0C                     CMP    AL,12            ; Verify that not more than 12 commands
088E  76 03                     JBE    INT5             ; Jump if not more than 12 commands
0890  E9 0000 E                 JMP    ERROR_3          ; Jump if 'UNKNOWN COMMAND'
0893                      INT5:                         ;
0893  26: 8A 07                 MOV    AL,ES:[BX.CMDLEN] ; AL <-- Length of drive request structure
0896  80 7C 02 00               CMP    BYTE PTR [SI+2],00 ;
089A  74 11                     JZ     INT7             ; Skip char. device commands
089C  3A 44 02                  CMP    AL,[SI+2]        ; Compare with requested length
089F  73 03                     JAE    INT6             ; Jump if equal
08A1  E9 0000 E                 JMP    ERROR_5          ; Jump if 'BAD DRIVE REQUEST STRUCTURE LENGTH'
08A4                      INT6:                         ;
08A4  FF 14                     CALL   [SI]             ; Perform I/O packet command
                                                        ;
08A6  F6 06 0011 R C0           TEST   ERRBUF,0C0H      ; Test for normal termination
08AB  75 07                     JNZ    DSKERR           ; Jump to disk error routine
08AD                      INT7:                         ;
08AD  E9 0000 E                 JMP    EXIT             ; Jump to EXIT
                                                        ;
08B0                      CMDERR:                       ;
08B0  58                        POP    AX               ; Flush return address from stack
08B1  E9 0000 E                 JMP    ERROR_3          ; Generate 'UNKNOWN COMMAND' error
```

```
08B4                            DSKERR:                          ;
08B4  C4 1E 0000 E                        LES     BX,[PTRSAV]     ; Retrieve pointer to I/O data packet
08B8  26: 8A 47 02                        MOV     AL,ES:[BX.CMD]  ; AL <-- Command code
08BC  3C 04                               CMP     AL,04           ; Test if READ
08BE  74 0A                               JZ      DSKERR0         ; Jump if READ
08C0  3C 08                               CMP     AL,08           ; Test if WRITE
08C2  74 06                               JZ      DSKERR0         ; Jump if WRITE
08C4  3C 09                               CMP     AL,09           ; Test if VERIFY
08C6  74 02                               JZ      DSKERR0         ; Jump if VERIFY
                                                                  ;
08C8  EB 06                               JMP     SHORT DSKERR1   ; Jump if not READ, WRITE or VERIFY
08CA                            DSKERR0:                          ;
08CA  26: C7 47 12 0000                   MOV     ES:[BX.COUNT],0 ; Transfer counter <-- 0000
08D0                            DSKERR1:                          ;
08D0  F6 06 0012 R 02                     TEST    ERRBUF+1,02     ; Test for 'WRITE PROTECTED'
08D5  74 03                               JZ      DSKERR2         ;
08D7  E9 0000 E                           JMP     ERROR_0         ; Jump if 'WRITE PROTECTED'
08DA                            DSKERR2:                          ;
08DA  F6 06 0011 R 08                     TEST    ERRBUF,08       ; Test for 'DRIVE NOT READY'
08DF  74 03                               JZ      DSKERR3         ;
08E1  E9 0000 E                           JMP     ERROR_2         ; Jump if 'DRIVE NOT READY'
08E4                            DSKERR3:                          ;
08E4  F6 06 0011 R 80                     TEST    ERRBUF,80H      ; Test for 'UNKNOWN COMMAND'
08E9  74 03                               JZ      DSKERR4         ;
08EB  E9 0000 E                           JMP     ERROR_3         ; Jump if 'UNKNOWN COMMAND'
08EE                            DSKERR4:                          ;
08EE  F6 06 0012 R 20                     TEST    ERRBUF+1,20H    ; Test for 'CRC ERROR'
08F3  74 03                               JZ      DSKERR5         ;
08F5  E9 0000 E                           JMP     ERROR_4         ; Jump if 'CRC ERROR'
08F8                            DSKERR5:                          ;
08F8  F6 06 0011 R 20                     TEST    ERRBUF,20H      ; Test for 'SEEK ERROR'
08FD  75 0A                               JNZ     DSKERR6         ;
08FF  F6 06 0013 R 10                     TEST    ERRBUF+2,10H    ; Test for 'SEEK ERROR'
0904  74 03                               JZ      DSKERR6         ;
0906  E9 0000 E                           JMP     ERROR_6         ; Jump if 'SEEK ERROR'
0909                            DSKERR6:                          ;
0909  F6 06 0013 R 01                     TEST    ERRBUF+2,01     ; Test for 'UNKNOWN MEDIA'
090E  74 03                               JZ      DSKERR7         ;
0910  E9 0000 E                           JMP     ERROR_7         ; Jump if 'UNKNOWN MEDIA'
0913                            DSKERR7:                          ;
0913  F6 06 0012 R 04                     TEST    ERRBUF+1,04     ; Test for 'SECTOR NOT FOUND'
0918  74 03                               JZ      DSKERR8         ;
091A  E9 0000 E                           JMP     ERROR_8         ; Jump if 'SECTOR NOT FOUND'
091D                            DSKERR8:                          ;
091D  B4 05                               MOV     AH,WRITDAT      ;
091F  22 26 0008 R                        AND     AH,COMSTR+1     ;
0923  80 FC 05                            CMP     AH,WRITDAT      ;
0926  75 03                               JNZ     DSKERR9         ;
0928  E9 0000 E                           JMP     ERROR_10        ; Jump if error after WRITE COMMAND
092B                            DSKERR9:                          ;
092B  B4 06                               MOV     AH,READDAT      ;
092D  22 26 0008 R                        AND     AH,COMSTR+1     ;
0931  80 FC 06                            CMP     AH,READDAT      ;
0934  75 03                               JNZ     DSKERR10        ;
0936  E9 0000 E                           JMP     ERROR_11        ; Jump if error after READ COMMAND
0939                            DSKERR10:                         ;
0939  E9 0000 E                           JMP     ERROR_12        ; Rest becomes 'GENERAL FAILURE'
```

DSKDRV

```
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;
;
;
;
;
;
;    ROUTINE NAME:      DSK_INIT
;
;
;
;
;
;    FUNCTION:          DISK INITIALIZE
;
;
;
;
;    ENTRY VIA:         CALL
;
;
;    ENTRY CONDITIONS:  NONE
;
;
;
;
;    EXIT VIA:          RETURN
;
;
;    EXIT CONDITIONS:   NONE
;
;
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
```

```
0930                     DSK_INIT:                     ;
093C  A0 0000 E                  MOV    AL,DRVMAX      ;
093F  26: 88 47 0D               MOV    ES:[BX.MEDIA],AL; I/O data packet <-- max. number of units
                                                       ;
0943  BE 0AC7 R                  MOV    SI,OFFSET DREND ;
0946  26: 89 77 0E               MOV    ES:[BX.TRANS],SI; I/O data packet <-- BREAK ADDR (OFFSET)
094A  26: 8C 4F 10               MOV    ES:[BX.TRANS+2],CS ;              <--            (SEGMENT)
                                                       ;
094E  BE 09FE R                  MOV    SI,OFFSET INITTAB ;
0951  26: 89 77 12               MOV    ES:[BX.COUNT],SI; I/O data packet <-- Pointer to
0955  26: 8C 4F 14               MOV    ES:[BX.START],CS;                     BPB array
                                                       ;
0959  A0 0000 E                  MOV    AL,FL_OUT_RETRIES ;
095C  A2 0024 R                  MOV    RETRIES,AL     ; Set retry counter
                                                       ;
095F  C6 06 0011 R 00            MOV    ERRBUF,00      ; Set normal termination
0964  C3                         RET                   ;
```

```
;****************************************************************************
;****************************************************************************
;****************************************************************************
;
;
;
;
;
;
;
;    ROUTINE NAME:      MEDIAC
;
;
;
;
;
;    FUNCTION:          MEDIA CHECK
;
;
;
;
;    ENTRY VIA:         CALL
;
;
;    ENTRY CONDITIONS:  CL - UNIT CODE
;                       CH - MEDIA DESCRIPTOR
;
;
;
;
;    EXIT VIA:          RETURN
;
;
;    EXIT CONDITIONS:
;
;
;
;****************************************************************************
;****************************************************************************
;****************************************************************************
```

```
0965                        MEDIAC:                         ;
0965  E4 13                        IN     AL,SYSSTA         ;
0967  A8 08                        TEST   AL,08             ; Test DISK INTERRUPT BIT
0969  74 06                        JZ     MEDIAC1           ; Jump if no interrupt
                                                            ;
096B  C7 06 0000 E 0101            MOV    FLTAB,0101H       ; Set interrupt flags for both units
0971                        MEDIAC1:                        ;
0971  C6 06 0011 R 00              MOV    ERRBUF,00         ; Set normal termination
                                                            ;
0976  BE 0000 E                    MOV    SI,OFFSET FLTAB   ; SI <-- Addr of FLAG TABLE
0979  B4 00                        MOV    AH,00             ;
097B  26: 8A 47 01                 MOV    AL,ES:[BX.UNIT]   ; AX <-- Unit code
097F  03 F0                        ADD    SI,AX             ; Compute entry in FLAG TABLE
                                                            ;
0981  80 3C 00                     CMP    BYTE PTR [SI],0   ; Check if interrupt flag is set for that unit
0984  74 09                        JZ     MEDIAC2           ; Jump if not set
                                                            ;
0986  C6 04 00                     MOV    BYTE PTR [SI],0   ; Reset interrupt flag
0989  26: C6 47 0E 00              MOV    BYTE PTR ES:[BX.TRANS],00 ; I/O data packet <-- don't know
098E  C3                           RET                      ;
098F                        MEDIAC2:                        ;
098F  26: C6 47 0E 01              MOV    BYTE PTR ES:[BX.TRANS],01 ; I/O data packet <-- not changed
0994  C3                           RET                      ;
```

```
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
;
;
;
;
;
;
;
;    ROUTINE NAME:       GET_BPB
;
;
;
;
;
;    FUNCTION:           Get BPB (BIOS PARAMETER BLOCK)
;
;
;
;
;    ENTRY VIA:          CALL
;
;
;    ENTRY CONDITIONS:   DRV variable is set
;
;
;
;
;    EXIT VIA:           RETURN
;
;
;    EXIT CONDITIONS:    STATUS (returned in ERRBUF)
;
;
;
;*******************************************************************************
;*******************************************************************************
;*******************************************************************************
```

```
0995                          GET_BPB:                          ; Fill PIM variables to read 1st FAT
                                                                ; ---------------------------------
0995  C6 06 0000 R 00                  MOV    CYLMODE,00        ; PIM.CYLINDER MODE <-- 00
                                                                ; PIM.DRV already set
099A  C6 06 0002 R 00                  MOV    HEAD,00           ; PIM.HEAD <-- 00
099F  C6 06 0003 R 00                  MOV    TRACK,00          ; PIM.TRACK <-- 00
09A4  C6 06 0004 R 02                  MOV    SECTOR,02         ; PIM.SECTOR <-- 2nd phy. = 1st log
09A9  C7 06 0005 R 0001                MOV    SECCNT,01         ; PIM.SECCNT <-- one sector
                                                                ; DMA ADDRESS of scratch buffer already set
                                                                ;
09AF  E8 0425 R                        CALL   DREAD             ; Read 1st FAT sector
                                                                ;
09B2  F6 06 0011 R C0                  TEST   ERRBUF,0C0H       ; Test for normal termination
09B7  74 01                            JZ     GETBPB1           ; Jump if normal termination
09B9  C3                               RET                      ; else return
09BA                          GETBPB1:                          ;
09BA  8B 3E 0018 R                     MOV    DI,DMAADDR        ;
09BE  8E 06 001A R                     MOV    ES,DMAADDR+2      ; ES:DI <-- scratch buffer address
09C2  26: 8A 05                        MOV    AL,ES:[DI]        ; AL <-- media descriptor byte from FAT
                                                                ;
09C5  BE 0A02 R                        MOV    SI,OFFSET FE160   ; SI <-- Pointer to BPB (single sided)
09C8  3A 44 0A                         CMP    AL,[SI.MEDIAID]   ; Compare media descriptors in FAT & BPB
09CB  74 18                            JZ     GETBPB2           ; Jump if equal
                                                                ;
09CD  BE 0A0F R                        MOV    SI,OFFSET FF320   ; SI <-- Pointer to BPB (double sided)
09D0  3A 44 0A                         CMP    AL.[SI.MEDIAID]   ; Compare media descriptors in FAT & BPB
09D3  74 10                            JZ     GETBPB2           ; Jump if equal
                                                                ;
09D5  BE 0A1C R                        MOV    SI,OFFSET FC180   ; SI <-- Pointer to BPB (single sided)
09D8  3A 44 0A                         CMP    AL,[SI.MEDIAID]   ; Compare media descriptors in FAT & BPB
09DB  74 08                            JZ     GETBPB2           ; Jump if equal
                                                                ;
09DD  BE 0A29 R                        MOV    SI,OFFSET FD360   ; SI <-- Pointer to BPB (double sided)
09E0  3A 44 0A                         CMP    AL,[SI.MEDIAID]   ; Compare media descriptors in FAT & BPB
09E3  74 00                            JZ     GETBPB2           ; Jump if equal
                                                                ;
                                                                ;
                                                                ; No match, assume 'FE' disk
                                                                ;
                                                                ;
09E5                           GETBPB2:                         ;
09E5  8A 44 0A                         MOV    AL,[SI.MEDIAID]   ; AL <-- new media descriptor
                                                                ;
09E8  C4 1E 0000 E                     LES    BX,[PTRSAV]       ; Update I/O data packet
09EC  26: 88 47 0D                     MOV    ES:[BX.MEDIA],AL;        <-- media descriptor
09F0  26: 89 77 12                     MOV    ES:[BX.COUNT],SI;        <-- BPB pointer
09F4  26: 8C 4F 14                     MOV    ES:[BX.COUNT+2],CS ;     <-- CODE SEGMENT
                                                                ;
09F8  C6 06 0011 R 00                  MOV    ERRBUF,00         ; Set normal termination
09FD  C3                               RET                      ;
```

```
DSKDRV

                        ;
                        ; ************************************
                        ; ***  BPB's (BIOS PARAMETER BLOCKS)  ***
                        ; ************************************
                        ;
                        ;
                        ;
09FE                    INITTAB:                    ;
09FE  0A29 R                    DW      FD360       ;
0A00  0A29 R                    DW      FD360       ;
                                                    ;
                                                    ;
0A02                    FE160:                      ; Single sided drive (5 1/4 ")
0A02  0200                      DW      512         ; Sector size
0A04  01                        DB      1           ; Sectors per allocation unit
0A05  0001                      DW      1           ; Number of reserverd sectors
0A07  02                        DB      2           ; Number of FATs
0A08  0040                      DW      4*16        ; Number of directory entries
0A0A  0140                      DW      40*8        ; Total number of sectors
0A0C  FE                        DB      0FEH        ; Media byte
0A0D  0001                      DW      1           ; Sectors for one FAT
                                                    ;
                                                    ;
0A0F                    FF320:                      ; Double sided drive (5 1/4 ")
0A0F  0200                      DW      512         ; Sector size
0A11  02                        DB      2           ; Sectors per allocation unit
0A12  0001                      DW      1           ; Number of reserverd sectors
0A14  02                        DB      2           ; Number of FATs
0A15  0070                      DW      7*16        ; Number of directory entries
0A17  0280                      DW      2*40*8      ; Total number of sectors
0A19  FF                        DB      0FFH        ; Media byte
0A1A  0001                      DW      1           ; Sectors for one FAT
                                                    ;
                                                    ;
0A1C                    FC180:                      ; Single sided drive (5 1/4 ")
0A1C  0200                      DW      512         ; Sector size
0A1E  01                        DB      1           ; Sectors per allocation unit
0A1F  0001                      DW      1           ; Number of reserverd sectors
0A21  02                        DB      2           ; Number of FATs
0A22  0040                      DW      4*16        ; Number of directory entries
0A24  0168                      DW      40*9        ; Total number of sectors
0A26  FC                        DB      0FCH        ; Media byte
0A27  0002                      DW      2           ; Sectors for one FAT
                                                    ;
                                                    ;
0A29                    FD360:                      ; Double sided drive (5 1/4 ")
0A29  0200                      DW      512         ; Sector size
0A2B  02                        DB      2           ; Sectors per allocation unit
0A2C  0001                      DW      1           ; Number of reserverd sectors
0A2E  02                        DB      2           ; Number of FATs
0A2F  0070                      DW      7*16        ; Number of directory entries
0A31  0200                      DW      2*40*9      ; Total number of sectors
0A33  FD                        DB      0FDH        ; Media byte
0A34  0002                      DW      2           ; Sectors for one FAT
```

```
;******************************************************************************
;******************************************************************************
;******************************************************************************
;
;
;
;
;
;
;
;   ROUTINE NAME:      DVERIFY
;
;
;
;
;
;   FUNCTION:          WRITE with verify
;
;
;
;
;   ENTRY VIA:         CALL
;
;
;   ENTRY CONDITIONS:  Following  variables are set:
;                      CYLMODE, DRV, HEAD, TRACK, SECTOR,
;                      SECCNT (Number of sectors),
;                      and DMAADDR (SEGMENT and OFFSET)
;
;
;
;   EXIT VIA:          RETURN
;
;
;   EXIT CONDITIONS:   STATUS (returned in ERRBUF)
;
;
;
;******************************************************************************
;******************************************************************************
;******************************************************************************
```

```
0A36                        DVERIFY:                    ;
0A36  8A 26 0002 R                   MOV    AH,HEAD     ; Save PIM.HEAD
0A3A  A0 0003 R                      MOV    AL,TRACK    ;      PIM.TRACK
0A3D  50                             PUSH   AX          ;
0A3E  8A 26 0004 R                   MOV    AH,SECTOR   ;      PIM.SECTOR
0A42  50                             PUSH   AX          ;
0A43  A1 0005 R                      MOV    AX,SECCNT   ;      PIM.SECCNT variables to READ
0A46  50                             PUSH   AX          ;                        after WRITE
                                                        ;
0A47  E8 042F R                      CALL   DWRITE      ; Do low level WRITE DATA
                                                        ;
0A4A  58                             POP    AX          ;
0A4B  A3 0005 R                      MOV    SECCNT,AX   ; Restore PIM.SECCNT
0A4E  58                             POP    AX          ;
0A4F  88 26 0004 R                   MOV    SECTOR,AH   ;         PIM.SECTOR
0A53  58                             POP    AX          ;
0A54  A2 0003 R                      MOV    TRACK,AL    ;         PIM.TRACK
0A57  88 26 0002 R                   MOV    HEAD,AH     ;         PIM.HEAD variables to READ
                                                        ;                        after WRITE
                                                        ;
                                                        ; Fill PIM variables to READ after WRITE
                                                        ; ---------------------------------------
0A5B  C7 06 0018 R 0AC6 R            MOV    DMAADDR,OFFSET BYTEBUF ; DMAADDR <-- OFFSET
0A61  8C 0E 001A R                   MOV    DMAADDR+2,CS   ;               <-- SEGMENT
0A65  C7 06 001C R 0001              MOV    DMALENG,0001   ; DMALENG       <-- 1 byte transfer
0A6B  C6 06 001E R 47                MOV    DMAFUNC,DMAWRT ; DMAFUNC       <-- WRITE DMA COMMAND
0A70                        DVER1:                      ;
0A70  B1 06                          MOV    CL,READDAT  ; CL <-- READ DATA COMMAND
0A72  E8 06A4 R                      CALL   SETUP9      ; Set up COMMAND STRING and DMA
0A75  E8 0732 R                      CALL   XWAIT       ; Send COMMAND STRING to FDC
0A78  E8 0750 R                      CALL   GETBYT      ; Get STATUS BYTES
                                                        ;
0A7B  F6 06 0011 R C0                TEST   ERRBUF,0C0H ; Test for normal termination
0A80  74 04                          JZ     DVER2       ; Jump if normal termination
                                                        ;
0A82  58                             POP    AX          ; Flush return addr from stack
0A83  E9 0000 E                      JMP    ERROR_10    ; Generate 'WRITE FAULT' error
0A86                        DVER2:                      ;
0A86  FF 0E 0005 R                   DEC    SECCNT      ; Decrement sector counter
0A8A  75 01                          JNZ    DVER3       ; Jump if another I/O is necessary
0A8C  C3                             RET                ; else return if I/O is complete
0A8D                        DVER3:                      ;
0A8D  A0 0004 R                      MOV    AL,SECTOR   ;
0A90  3A 06 001F R                   CMP    AL,SECTRK   ; Check if next sector fits in track
0A94  74 06                          JZ     DVER4       ; Jump if not
                                                        ;
0A96  FE 06 0004 R                   INC    SECTOR      ; Increment SECTOR (next sector fits in track)
0A9A  EB D4                          JMP    DVER1       ;
0A9C                        DVER4:                      ;
0A9C  C6 06 0004 R 01                MOV    SECTOR,1    ; Set SECTOR to begin of track
                                                        ;
0AA1  80 3E 0000 R 00                CMP    CYLMODE,0   ; Check if double sided disk
0AA6  74 06                          JZ     DVER5       ; Jump if double sided
                                                        ;
                                                        ; Single sided
                                                        ; -----------
0AA8  FE 06 0003 R                   INC    TRACK       ; Increment TRACK
0AAC  EB C2                          JMP    DVER1       ;
```

```
0AAE                        DVER5:                              ; Double sided
                                                                ; -----------
0AAE  80 3E 0002 R 00              CMP     HEAD,0               ;
0AB3  75 06                        JNZ     DVER6                ; If HEAD 0
0AB5  FE 06 0002 R                 INC     HEAD                 ; then set HEAD 1
0AB9  EB B5                        JMP     DVER1                ;
0ABB                        DVER6:                              ; else
0ABB  C6 06 0002 R 00              MOV     HEAD,0               ; set HEAD 0
0AC0  FE 06 0003 R                 INC     TRACK                ; and increment TRACK
0AC4  EB AA                        JMP     DVER1                ;
                                                                ;
                                                                ;
0AC6                        BYTEBUF:                            ;
0AC6  ??                           DB      ?                    ; BYTE BUFFER to detect CRC errors

0AC7                        DREND:                              ; End of driver

0AC7                        CSEG   ENDS
                            END
```

```
DSKDRV


Structures and records:

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

DPB. . . . . . . . . . . . . . .        000D    0008
  SECSIZE. . . . . . . . . . . . . .    0000
  ALLOC. . . . . . . . . . . . . . .    0002
  RESSEC . . . . . . . . . . . . . .    0003
  FATS . . . . . . . . . . . . . . .    0005
  MAXDIR . . . . . . . . . . . . . .    0006
  SECTORS. . . . . . . . . . . . . .    0008
  MEDIAID. . . . . . . . . . . . . .    000A
  FATSEC . . . . . . . . . . . . . .    000B
IODAT. . . . . . . . . . . . . .        0016    000A
  CMDLEN . . . . . . . . . . . . . .    0000
  UNIT . . . . . . . . . . . . . . .    0001
  CMD. . . . . . . . . . . . . . . .    0002
  STATUS . . . . . . . . . . . . . .    0003
  MEDIA. . . . . . . . . . . . . . .    000D
  TRANS. . . . . . . . . . . . . . .    000E
  COUNT. . . . . . . . . . . . . . .    0012
  START. . . . . . . . . . . . . . .    0014

Segments and groups:

                N a m e                 Size    align   combine class

CSEG . . . . . . . . . . . . . .        0AC7    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                 Type    Value   Attr

BANK . . . . . . . . . . . . . .        Number  00E0
BYTEBUF. . . . . . . . . . . . .        L NEAR  0AC6    CSEG
BYTSEC . . . . . . . . . . . . .        L BYTE  0021    CSEG
CMDERR . . . . . . . . . . . . .        L NEAR  0880    CSEG
COAD . . . . . . . . . . . . . .        Number  0026
COMSTR . . . . . . . . . . . . .        L BYTE  0007    CSEG
COTC . . . . . . . . . . . . . .        Number  0027
CYLMODE. . . . . . . . . . . . .        L BYTE  0000    CSEG
DCOMD. . . . . . . . . . . . . .        Number  0051
DENSITY. . . . . . . . . . . . .        L BYTE  0020    CSEG
DFORMAT. . . . . . . . . . . . .        L NEAR  0685    CSEG
DMA. . . . . . . . . . . . . . .        L NEAR  0779    CSEG
DMAADDR. . . . . . . . . . . . .        L WORD  0018    CSEG
DMAFUNC. . . . . . . . . . . . .        L BYTE  001E    CSEG
DMALENG. . . . . . . . . . . . .        L WORD  001C    CSEG
DMAMB. . . . . . . . . . . . . .        Number  002A
DMAMO. . . . . . . . . . . . . .        Number  002B
DMAREAD. . . . . . . . . . . . .        Number  0048
DMAWRT . . . . . . . . . . . . .        Number  0047
DREAD. . . . . . . . . . . . . .        L NEAR  0425    CSEG
DREADID. . . . . . . . . . . . .        L NEAR  0651    CSEG
DREND. . . . . . . . . . . . . .        L NEAR  0AC7    CSEG    Global
DREST. . . . . . . . . . . . . .        L NEAR  05FA    CSEG
DREST1 . . . . . . . . . . . . .        L NEAR  05FC    CSEG
DREST2 . . . . . . . . . . . . .        L NEAR  0610    CSEG
DREST3 . . . . . . . . . . . . .        L NEAR  0625    CSEG
DRV. . . . . . . . . . . . . . .        L BYTE  0001    CSEG
```

```
DRVMAX . . . . . . . . . . . . .   V BYTE  0000   CSEG    External
DSEEK. . . . . . . . . . . . . .   L NEAR  0626   CSEG
DSEEK1 . . . . . . . . . . . . .   L NEAR  0647   CSEG
DSIS . . . . . . . . . . . . . .   L NEAR  0674   CSEG
DSKERR . . . . . . . . . . . . .   L NEAR  08B4   CSEG
DSKERR0. . . . . . . . . . . . .   L NEAR  08CA   CSEG
DSKERR1. . . . . . . . . . . . .   L NEAR  08D0   CSEG
DSKERR10 . . . . . . . . . . . .   L NEAR  0939   CSEG
DSKERR2. . . . . . . . . . . . .   L NEAR  08DA   CSEG
DSKERR3. . . . . . . . . . . . .   L NEAR  08E4   CSEG
DSKERR4. . . . . . . . . . . . .   L NEAR  08EE   CSEG
DSKERR5. . . . . . . . . . . . .   L NEAR  08F8   CSEG
DSKERR6. . . . . . . . . . . . .   L NEAR  0909   CSEG
DSKERR7. . . . . . . . . . . . .   L NEAR  0913   CSEG
DSKERR8. . . . . . . . . . . . .   L NEAR  091D   CSEG
DSKERR9. . . . . . . . . . . . .   L NEAR  092B   CSEG
DSKTBL . . . . . . . . . . . . .   L NEAR  07B7   CSEG
DSK_INIT . . . . . . . . . . . .   L NEAR  093C   CSEG
DSK_INT. . . . . . . . . . . . .   L NEAR  070E   CSEG    Global
DSTAT. . . . . . . . . . . . . .   Number  0050
DVER1. . . . . . . . . . . . . .   L NEAR  0A70   CSEG
DVER2. . . . . . . . . . . . . .   L NEAR  0A86   CSEG
DVER3. . . . . . . . . . . . . .   L NEAR  0A8D   CSEG
DVER4. . . . . . . . . . . . . .   L NEAR  0A9C   CSEG
DVER5. . . . . . . . . . . . . .   L NEAR  0AAE   CSEG
DVER6. . . . . . . . . . . . . .   L NEAR  0ABB   CSEG
DVERIFY. . . . . . . . . . . . .   L NEAR  0A36   CSEG
DWRITE . . . . . . . . . . . . .   L NEAR  042F   CSEG
ERRBUF . . . . . . . . . . . . .   L BYTE  0011   CSEG
ERROR_0. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_1. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_10 . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_11 . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_12 . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_2. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_3. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_4. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_5. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_6. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_7. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_8. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
ERROR_9. . . . . . . . . . . . .   L NEAR  0000   CSEG    External
EXIT . . . . . . . . . . . . . .   L NEAR  0000   CSEG    External
FC180. . . . . . . . . . . . . .   L NEAR  0A1C   CSEG
FD360. . . . . . . . . . . . . .   L NEAR  0A29   CSEG
FDCRA. . . . . . . . . . . . . .   Number  0051
FDCRDY . . . . . . . . . . . . .   L NEAR  0760   CSEG
FDCSIS . . . . . . . . . . . . .   Number  0008
FE160. . . . . . . . . . . . . .   L NEAR  0A02   CSEG
FF320. . . . . . . . . . . . . .   L NEAR  0A0F   CSEG
FLTAB. . . . . . . . . . . . . .   V WORD  0000   CSEG    External
FL_OUT_RETRIES . . . . . . . . .   V BYTE  0000   CSEG    External
GETBPB1. . . . . . . . . . . . .   L NEAR  09BA   CSEG
GETBPB2. . . . . . . . . . . . .   L NEAR  09E5   CSEG
GETBYT . . . . . . . . . . . . .   L NEAR  0750   CSEG
GETBYT1. . . . . . . . . . . . .   L NEAR  0753   CSEG
GET_BPB. . . . . . . . . . . . .   L NEAR  0995   CSEG
GPL. . . . . . . . . . . . . . .   L BYTE  0022   CSEG
HEAD . . . . . . . . . . . . . .   L BYTE  0002   CSEG
IDREAD . . . . . . . . . . . . .   Number  000A

INITTAB. . . . . . . . . . . . .   L NEAR  09FE   CSEG
INT0 . . . . . . . . . . . . . .   L NEAR  0817   CSEG
INT1 . . . . . . . . . . . . . .   L NEAR  0837   CSEG
INT2 . . . . . . . . . . . . . .   L NEAR  085C   CSEG
INT3 . . . . . . . . . . . . . .   L NEAR  0869   CSEG
INT4 . . . . . . . . . . . . . .   L NEAR  0881   CSEG
```

```
DSKDRV
INT5 . . . . . . . . . . . . . .      L NEAR  0893   CSEG
INT6 . . . . . . . . . . . . . .      L NEAR  08A4   CSEG
INT7 . . . . . . . . . . . . . .      L NEAR  08AD   CSEG
IO . . . . . . . . . . . . . . .      L NEAR  0535   CSEG
IO1. . . . . . . . . . . . . . .      L NEAR  0436   CSEG
IO10 . . . . . . . . . . . . . .      L NEAR  04E9   CSEG
IO11 . . . . . . . . . . . . . .      L NEAR  050D   CSEG
IO15 . . . . . . . . . . . . . .      L NEAR  0513   CSEG
IO16 . . . . . . . . . . . . . .      L NEAR  0525   CSEG
IO17 . . . . . . . . . . . . . .      L NEAR  0531   CSEG
IO2. . . . . . . . . . . . . . .      L NEAR  043E   CSEG
IO20 . . . . . . . . . . . . . .      L NEAR  0538   CSEG
IO21 . . . . . . . . . . . . . .      L NEAR  054C   CSEG
IO22 . . . . . . . . . . . . . .      L NEAR  0555   CSEG
IO23 . . . . . . . . . . . . . .      L NEAR  055E   CSEG
IO24 . . . . . . . . . . . . . .      L NEAR  0567   CSEG
IO25 . . . . . . . . . . . . . .      L NEAR  0572   CSEG
IO3. . . . . . . . . . . . . . .      L NEAR  0457   CSEG
IO30 . . . . . . . . . . . . . .      L NEAR  0574   CSEG
IO31 . . . . . . . . . . . . . .      L NEAR  057C   CSEG
IO32 . . . . . . . . . . . . . .      L NEAR  05A3   CSEG
IO33 . . . . . . . . . . . . . .      L NEAR  05B6   CSEG
IO34 . . . . . . . . . . . . . .      L NEAR  05C3   CSEG
IO35 . . . . . . . . . . . . . .      L NEAR  05CC   CSEG
IO36 . . . . . . . . . . . . . .      L NEAR  05E9   CSEG
IO4. . . . . . . . . . . . . . .      L NEAR  0461   CSEG
IO5. . . . . . . . . . . . . . .      L NEAR  047F   CSEG
IO6. . . . . . . . . . . . . . .      L NEAR  048D   CSEG
IO7. . . . . . . . . . . . . . .      L NEAR  0496   CSEG
IO8. . . . . . . . . . . . . . .      L NEAR  04A8   CSEG
IO9. . . . . . . . . . . . . . .      L NEAR  04CE   CSEG
MEDIAC . . . . . . . . . . . . .      L NEAR  0965   CSEG
MEDIAC1. . . . . . . . . . . . .      L NEAR  0971   CSEG
MEDIAC2. . . . . . . . . . . . .      L NEAR  098F   CSEG
MOTORCK. . . . . . . . . . . . .      L NEAR  0767   CSEG
MOTORCK1 . . . . . . . . . . . .      L NEAR  0770   CSEG
MOTORCK2 . . . . . . . . . . . .      L NEAR  0773   CSEG
MOTORON. . . . . . . . . . . . .      Number  0014
PATTERN. . . . . . . . . . . . .      L BYTE  0023   CSEG
PTRSAV . . . . . . . . . . . . .      V DWORD 0000   CSEG    External
READDAT. . . . . . . . . . . . .      Number  0006
READTRK. . . . . . . . . . . . .      Number  0002
RESTORE. . . . . . . . . . . . .      Number  0007
RETRIES. . . . . . . . . . . . .      L BYTE  0024   CSEG
SECCNT . . . . . . . . . . . . .      L WORD  0005   CSEG
SECTOR . . . . . . . . . . . . .      L BYTE  0004   CSEG
SECTRK . . . . . . . . . . . . .      L BYTE  001F   CSEG
SEEKTRK. . . . . . . . . . . . .      Number  000F
SET1 . . . . . . . . . . . . . .      L NEAR  068A   CSEG
SETUP6 . . . . . . . . . . . . .      L NEAR  06F9   CSEG
SETUP9 . . . . . . . . . . . . .      L NEAR  06A4   CSEG
SSB. . . . . . . . . . . . . . .      L WORD  0025   CSEG
SYSSTA . . . . . . . . . . . . .      Number  0013
TRACK. . . . . . . . . . . . . .      L BYTE  0003   CSEG

WRITDAT. . . . . . . . . . . . .      Number  0005
WRITFMT. . . . . . . . . . . . .      Number  000D
XWAIT. . . . . . . . . . . . . .      L NEAR  0732   CSEG
XWAIT1 . . . . . . . . . . . . .      L NEAR  073C   CSEG
```

```
                        ;**************************************************
                        ;*                                                *
                        ;*      W I N C H E S T E R   D I S K             *
                        ;*                                                *
                        ;*              D R I V E R                       *
                        ;*                                                *
                        ;**************************************************
                        ;
                        ;
                        ;
                        ;     DEFINE OFFSETS FOR IO DATA PACKET
                        ;
= 0000                  CMDLEN  EQU     0               ;LENGTH OF THIS BLOCK
= 0001                  UNIT    EQU     1               ;SUB UNIT SPECIFIER
= 0002                  CMD     EQU     2               ;COMMAND CODE
= 0003                  STATUS  EQU     3               ;STATUS
= 000D                  MEDIA   EQU     13              ;MEDIA DESCRIPTOR
= 000E                  TRANS   EQU     14              ;TRANSFER ADDRESS
= 0012                  COUNT   EQU     18              ;COUNT OF SECTORS
= 0014                  START   EQU     20              ;FIRST BLOCK TO TRANSFER
                        ;
                        ;     WINCHESTER DISK SYSTEM BYTE ADDRESSES
                        ;
= 0416                  WINCHDRIVES     EQU     0416H   ;NO. OF WINCHESTER DRIVES
= 0417                  WIOUTRETRIES    EQU     0417H   ;NO. OF RETRIES (OUT)
= 0418                  WIINRETRIES     EQU     0418H   ;NO. OF RETRIES (IN)
                        ;
                        ;
                        ;
                        ;          WINCHESTER DISK DEFINITIONS
                        ;          ===========================
                        ;
                        ;**********************************************
                        ;*                                            *
                        ;*      PORT DEFINITIONS                      *
                        ;*                                            *
                        ;**********************************************
                        ;
= 00C0                  HBASE   EQU     0C0H            ;     CONTROLLER BASE ADDR.
=                       DATA    EQU     HBASE           ; R/W DATA REGISTER
= 00C1                  ERROR   EQU     HBASE+1         ; R   ERROR REGISTER
= 00C1                  WPC     EQU     HBASE+1         ;   W WRITE PRECOMP. REGISTER
= 00C2                  SECNT   EQU     HBASE+2         ; R/W SECTOR COUNT REGISTER
= 00C3                  SECNO   EQU     HBASE+3         ; R/W SECTOR NUMBER REGISTER
= 00C4                  CYLLO   EQU     HBASE+4         ; R/W CYLINDER LOW REGISTER
= 00C5                  CYLHI   EQU     HBASE+5         ; R/W CYLINDER HIGH REGISTER
= 00C6                  SDH     EQU     HBASE+6         ; R/W ECC/CRC-BYTES PER SECTOR-DRIVE-HEAD
= 00C7                  STAT    EQU     HBASE+7         ; R   STATUS REGISTER
= 00C7                  COMMD   EQU     HBASE+7         ;   W COMMAND REGISTER
```

```
                                 ;
                                 ;**********************************************
                                 ;*                                            *
                                 ;*     DISK FUNCTIONS                         *
                                 ;*                                            *
                                 ;**********************************************

                                 ;
 = 0000                          STRATE  EQU    0               ;STEPING RATE TRACK TO TRACK = BUFFERED STEP
 = 0010                          REST    EQU    10H OR STRATE   ;RESTORE COMMAND WITH STRATE
 = 0070                          SEEK    EQU    70H OR STRATE   ;SEEK COMMAND WITH STRATE
 = 0020                          READ    EQU    20H             ;READ COMMAND
 = 0030                          WRITE   EQU    30H             ;WRITE COMMAND
 = 0050                          FORMAT  EQU    50H             ;FORMAT COMMAND
                                 ;
                                 ;
                                 ;**********************************************
                                 ;*                                            *
                                 ;*     ERRROR  REGISTER  EQUATES              *
                                 ;*                                            *
                                 ;**********************************************
                                 ;
 = 0001                          DAMNFD  EQU    01H             ; ADDR. MARK NOT FOUND
 = 0002                          TR0     EQU    02H             ; TRACK 0 ERROR
 = 0004                          ABC     EQU    04H             ; ABORTED COMMAND
 = 0010                          IDNFD   EQU    10H             ; ID NOT FOUND
 = 0020                          CRCID   EQU    20H             ; CRC-ERROR  ID-FIELD
 = 0040                          UNCOR   EQU    40H             ; UNCORRECTED DATA IN DATA FIELD
 = 0080                          BBD     EQU    80H             ; BAD BLOCK DETECTED
                                 ;
                                 ;**********************************************
                                 ;*                                            *
                                 ;*     STATUS REGISTER EQUATES                *
                                 ;*                                            *
                                 ;**********************************************
                                 ;
 = 0001                          CERR    EQU    01H             ; CONTROLLER ERROR
 = 0004                          CORRD   EQU    04H             ; DATA CORRECTED IN DATA FIELD (ECC)
 = 0008                          CDRQ    EQU    08H             ; CONTROLLER DATA REQUEST
 = 0010                          DSEEC   EQU    10H             ; DRIVE SEEK COMPLETE
 = 0020                          DWRFA   EQU    20H             ; DRIVE WRITE FAULT
 = 0040                          DREADY  EQU    40H             ; DRIVE READY
 = 0080                          CBUSY   EQU    80H             ; CONTROLLER BUSY
                                 ;
                                 ;
                                 ;**********************************************
                                 ;*                                            *
                                 ;*     SPECIALS                               *
                                 ;*                                            *
                                 ;**********************************************
                                 ;
 = 00A0                          SDHREG  EQU    0A0H            ;ECC/512 BYTES PER SECTOR
```

```
                              ;
0000                          CSEG    SEGMENT
                              ASSUME  CS:CSEG,DS:CSEG
                              ;
0000                          BEGIN:
                              ;
                              ;**********************************************************
                              ;*                                                        *
                              ;*   S P E C I A L   D E V I C E   H E A D E R            *
                              ;*                                                        *
                              ;**********************************************************
                              ;
                              ;---------------------------------------------+
                              ;     DWORD pointer to next device            ! 1 word offset.
                              ;        (-1,-1 if last device)               ! 1 word segement.
                              ;---------------------------------------------+
                              ;     Device attribute WORD                   ; 1 word.
                              ;       Bit 15 = 1 for character devices.     ;
                              ;                0 for block devices.         ;
                              ;                                             ;
                              ;       Charcter devices. (Bit 15=1)          ;
                              ;         Bit 0 = 1  current sti device.      ;
                              ;         Bit 1 = 1  current sto device.      ;
                              ;         Bit 2 = 1  current NUL device.      ;
                              ;         Bit 3 = 1  current Clock device.    ;
                              ;                                             ;
                              ;         Bit 14 = 1 IOCTL control bit.       ;
                              ;---------------------------------------------+
                              ;     Device strategy pointer.                ; 1 word offset.
                              ;---------------------------------------------+
                              ;     Device interrupt pointer.               ; 1 word offset.
                              ;---------------------------------------------+
                              ;     Device name field.                      ; 8 bytes.
                              ;       Character devices are any valid name  ;
                              ;         left justified, in a space filled   ;
                              ;         field.                              ;
                              ;       Block devices contain # of units in   ;
                              ;         the first byte.                     ;
                              ;---------------------------------------------+
                              ;
                              ;
0000  FFFF FFFF               WINDEV  DW      -1,-1           ;LINK TO NEXT DEVICE
0004  0000                            DW      0000H           ;ATTRIBUTES (BLOCK DEVICE)
0006  002B R                          DW      STRATEGY        ;STRATEGY ENTRY POINT
0008  003A R                          DW      INTERRUPT       ;INTERRUPT ENTRY POINT
000A  02                              DB      2               ;NUMBER OF UNITS
                              ;
                              ;       RELEASE ID
000B  01                              DB      01              ;ISSUE
000C  03                              DB      03              ;SUB ISSUE
000D  00                              DB      00              ;PATCH LEVEL
```

```
                                  ;
                                  ;     BIOS PARAMETER BLOCK ARRAY
                                  ;
000E  001E R                      WIBPB   DW      WIBPB1
0010  001E R                              DW      WIBPB1
0012  001E R                              DW      WIBPB1
0014  001E R                              DW      WIBPB1
0016  001E R                              DW      WIBPB1
0018  001E R                              DW      WIBPB1
001A  001E R                              DW      WIBPB1
001C  001E R                              DW      WIBPB1
                                  ;
                                  ;
                                  ;     BIOS PARAMETER BLOCK (BPB)
                                  ;
001E  0200                        WIBPB1  DW      512             ;BYTES PER SECTOR
0020  10                                  DB      16              ;SECTOR PER ALLOCATION UNIT
0021  0000                                DW      0               ;RESERVED SECTORS
0023  01                                  DB      1               ;NUMBER OF FAT'S
0024  0200                                DW      512             ;NUMBER OF ROOT DIRECT. ENTRIES
0026  2882                                DW      10370           ;NUMBER OF SECTORS PER DISK
0028  FA                                  DB      0FAH            ;MEDIA DESCRIPTOR
0029  0002                                DW      2               ;NUMBER OF FAT SECTORS
                                  ;
                                  ;
                                  ;
                                  ;
                                  ;   SIMPLISTIC STRATEGY ROUTINE FOR NON-MULTI-TASKING SYSTEM
                                  ;
                                  ;       CURRENTLY JUST SAVES I/O PACKET POINTER IN PTRSAV
                                  ;       FOR LATER PROCESSING BY THE INTERRUPT ROUTINE.
                                  ;
002B                              STRATP  PROC    FAR
                                  ;
002B                              STRATEGY:
002B  2E: 89 1E 0036 R                    MOV     CS:WORD PTR PTRSAV,BX
0030  2E: 8C 06 0038 R                    MOV     CS:WORD PTR PTRSAV+2,ES
0035  CB                                  RET
                                  ;
0036                              STRATP  ENDP
                                  ;
0036  0000 0000                   PTRSAV  DW      0,0             ;STRATEGY POINTER SAVE
                                  :
                                  ;
003A                              INTERRUPT:
003A  56                                  PUSH    SI
003B  BE 0095 R                           MOV     SI,OFFSET WITBL
                                  ;
003E  50                          ENTRY:  PUSH    AX                      ;SAVE ALL NECESSARY REGISTERS.
003F  51                                  PUSH    CX
0040  52                                  PUSH    DX
0041  57                                  PUSH    DI
0042  55                                  PUSH    BP
0043  1E                                  PUSH    DS
0044  06                                  PUSH    ES
0045  53                                  PUSH    BX
0046  0E                                  PUSH    CS
0047  1F                                  POP     DS                      ;SET DATA SEG. TO CODE SEG.
```

```
0048  8B 1E 0036 R              MOV   BX,WORD PTR PTRSAV      ;Retrieve pointer to I/O Packet.
004C  8E 06 0038 R              MOV   ES,WORD PTR PTRSAV+2
0050  26: 8A 47 01              MOV   AL,ES:UNIT[BX]          ;AL = Unit code.
0054  A2 0316 R                 MOV   BYTE PTR WIPAR,AL
0057  26: 8A 67 0D              MOV   AH,ES:MEDIA[BX]         ;AH = Media descriptor.
005B  26: 80 7F 02 00           CMP   ES:BYTE PTR CMD[BX],0
0060  74 05                     JZ    ENTRY1                  ;SKIP MEDIA CHECK IF INIT FUNCT.
0062  80 FC FA                  CMP   AH,0FAH
0065  75 52                     JNZ   MEDERR                  ;Unknown Media descriptor
0067                    ENTRY1:
0067  26: 8B 4F 12              MOV   CX,ES:COUNT[BX]         ;CX = Contains byte/sector count.
006B  26: 8B 57 14              MOV   DX,ES:START[BX]         ;DX = Starting Logical sector.
006F  89 16 0318 R              MOV   WORD PTR WIPAR+2,DX
0073  97                        XCHG  DI,AX                   ;Move Unit/Media into DI temporarily.
0074  26: 8A 47 02              MOV   AL,ES:CMD[BX]           ;Retrieve Command type.
0078  32 E4                     XOR   AH,AH                   ;Clear upper half of AX for calc.
007A  03 F0                     ADD   SI,AX                   ;Comp. entry point in funct. table.
007C  03 F0                     ADD   SI,AX
007E  3C 0B                     CMP   AL,11                   ;Not more than 11 commands.
0080  77 3B                     JA    CMDERR                  ;Ah, well, error out.
0082  97                        XCHG  AX,DI                   ;Move Unit & Media back.
0083  26: 8B 7F 10              MOV   DI,ES:TRANS+2 [BX]      ;DI = addess of Transfer address.
0087  89 3E 031C R              MOV   WORD PTR WIPAR+6,DI     ;Buffer Addr. to PIM table
008B  26: 8B 7F 0E              MOV   DI,ES:TRANS [BX]
008F  89 3E 031E R              MOV   WORD PTR WIPAR+8,DI
0093  FF 24                     JMP   WORD PTR[SI]            ;Perform I/O packet command.
                        ;
                        ;
                        ;    WINCHESTER DISK FUNCTION TABLE
                        ;
0095  03F0 R            WITBL   DW    WIINIT        ;0  - Initialize Driver.
0097  00E8 R                    DW    MEDIAC        ;1  - Return current media code.
0099  010C R                    DW    GETBPB        ;2  - Get Bios Parameter Block.
009B  00BD R                    DW    CMDERR        ;3  - Reserved. (currently returns error)
009D  014B R                    DW    WIREAD        ;4  - Block read.
009F  00AD R                    DW    BUSEXIT       ;5  - (Not used, return busy flag)
00A1  00C4 R                    DW    EXIT          ;6  - Return status. (Not used)
00A3  00C4 R                    DW    EXIT          ;7  - Flush input buffer. (Not used.)
00A5  0218 R                    DW    WIWRT         ;8  - Block write.
00A7  02FD R                    DW    WIWRTV        ;9  - Block write with verify.
00A9  00C4 R                    DW    EXIT          ;10 - Return output status.
00AB  00C4 R                    DW    EXIT          ;11 - Flush output buffer. (Not used.)
                        ;
                        ;
                        ;  COMMON ERROR PROCESSING ROUTINE.
                        ;   AL = ERROR CODE.
                        ;
                        ;   Error # 0 = Write Protect violation.
                        ;           1 = Unknown unit.
                        ;           2 = Drive not ready.
                        ;           3 = Unknown command in I/O packet.
                        ;           4 = CRC error.
                        ;           5 = Bad drive request structure length.
                        ;           6 = Seek error.
                        ;           7 = Unknown media discovered.
                        ;           8 = Sector not found.
                        ;           9 = Printer out of paper.
                        ;          10 = Write fault.
                        ;          11 = Read fault.
                        ;          12 = General failure.
```

```
                                 ;
00AD                             BUSEXIT:                        ;Device busy exit.
00AD  B4 03                             MOV    AH,00000011B      ;Set busy and done bits.
00AF  EB 15                             JMP    SHORT   EXIT1
00B1                             UNITERR:
00B1  B0 01                             MOV    AL,1
00B3  EB 0A                             JMP    SHORT   ERREXIT
00B5  B0 05                      LENERR: MOV   AL,5              ;Bad drive request struct.length
00B7  EB 06                             JMP    SHORT   ERREXIT
00B9  B0 07                      MEDERR: MOV   AL,7              ;Unknown Media discovered.
00BB  EB 02                             JMP    SHORT   ERREXIT
00BD  B0 03                      CMDERR: MOV   AL,3              ;Set unknown command error #.
00BF                             ERREXIT:
00BF  B4 81                             MOV    AH,10000001B      ;Set error and done bits.
00C1  F9                                STC                      ;Set Carry bit.
00C2  EB 02                             JMP    SHORT   EXIT1     ;Quick way out.
                                 ;
00C4                             EXITP  PROC   FAR
                                 ;
00C4  B4 01                      EXIT:  MOV    AH,00000001B      ;Set done bit for MSDOS.
00C6  8B 1E 0036 R               EXIT1: MOV    BX,WORD PTR PTRSAV
00CA  8E 06 0038 R                      MOV    ES,WORD PTR PTRSAV+2
00CE  26: 89 47 03                      MOV    ES:STATUS[BX],AX  ;Save operation complete and status.
00D2  5B                                POP    BX                ;Restore registers.
00D3  07                                POP    ES
00D4  1F                                POP    DS
00D5  5D                                POP    BP
00D6  5F                                POP    DI
00D7  5A                                POP    DX
00D8  59                                POP    CX
00D9  58                                POP    AX
00DA  5E                                POP    SI
00DB  CB                                RET                      ;RESTORE REGS AND RETURN
                                 ;
00DC                             EXITP  ENDP
                                 ;
                                 ;
                                 ;    MOVE LENGTH OF DRIVE REQU. STRUCT. TO AL REG.
                                 ;
00DC  8B 1E 0036 R               GETLEN: MOV   BX,WORD PTR PTRSAV
00E0  8E 06 0038 R                      MOV    ES,WORD PTR PTRSAV+2
00E4  26: 8A 07                         MOV    AL,ES:BYTE PTR CMDLEN[BX]
00E7  C3                                RET
                                 ;
                                 ;
00E8                             MEDIAC:
00E8  E8 00DC R                         CALL   GETLEN            ;GET DRIVE STRUCT. LENGTH
00EB  3C 0F                             CMP    AL,15
00ED  72 C6                             JB     LENERR            ;BAD STRUCTURE LENGTH
00EF  BF 0143 R                         MOV    DI,OFFSET UNITTAB
00F2  A0 0316 R                         MOV    AL,BYTE PTR WIPAR
00F5  98                                CBW
00F6  03 F8                             ADD    DI,AX
00F8  8A 05                             MOV    AL,BYTE PTR [DI]
00FA  3C 00                             CMP    AL,0
00FC  75 07                             JNZ    MEDIAC1
00FE  26: C6 47 0E 01                   MOV    ES:BYTE PTR TRANS[BX],1
0103  EB BF                             JMP    EXIT
```

```
0105                            MEDIAC1:
0105  26: C6 47 0E 00                   MOV     ES:BYTE PTR TRANS[BX],0 ;SET MEDIA NOT CHANGED
010A  EB 08                             JMP     EXIT
                                ;
010C                            GETBPB:
010C  E8 00DC R                         CALL    GETLEN                  ;GET DRIVE STRUCT. LENGTH
010F  3C 16                             CMP     AL,22
0111  72 1E                             JB      GETBPB2                 ;BAD STRUCT. LENGTH
0113  06                                PUSH    ES
0114  26: 8B 47 10                      MOV     AX,ES:TRANS+2[BX]
0118  8E C0                             MOV     ES,AX
011A  26: 8A 0D                         MOV     CL,BYTE PTR ES:[DI]
011D  07                                POP     ES
011E  BF 0143 R                         MOV     DI,OFFSET UNITTAB
0121  A0 0316 R                         MOV     AL,BYTE PTR WIPAR
0124  98                                CBW
0125  03 F8                             ADD     DI,AX
0127  80 F9 FA                          CMP     CL,0FAH
012A  74 07                             JZ      GETBPB1
012C  C6 05 FF                          MOV     BYTE PTR[DI],0FFH
012F  EB 08                             JMP     MEDERR
0131                            GETBPB2:
0131  EB 02                             JMP     LENERR
0133                            GETBPB1:
0133  C6 05 00                          MOV     BYTE PTR[DI],0
0136  BE 001E R                         MOV     SI,OFFSET WIBPB1
0139  26: 89 77 12                      MOV     ES:WORD PTR COUNT[BX],SI
013D  26: 8C 4F 14                      MOV     ES:WORD PTR COUNT+2[BX],CS
0141  EB 01                             JMP     EXIT
                                ;
0143      08 [                  UNITTAB DB      8 DUP(?)
                 ??
                    ]

                                ;
                                ;       READ DATA
                                ;
014B                            WIREAD:
014B  E8 00DC R                         CALL    GETLEN                  ;GET DRIVE REQU.STRUCT.LENGTH
014E  3C 16                             CMP     AL,22
0150  72 DF                             JB      GETBPB2                 ;BAD STRUCT. LENGTH
0152  E8 0305 R                         CALL    CHKDRIVE                ;CHECK IF UNIT AVAILABLE
0155  72 5F                             JC      WIREAD4                 ;UNIT ERROR
0157  E8 0321 R                         CALL    FIXREADY                ;CHECK IF DRIVE READY
015A  75 40                             JNZ     WIREAD3
015C  A1 0213 R                         MOV     AX,WORD PTR RETRYDEF    ;GET NO. OF RETRIES
015F  A3 0215 R                         MOV     WORD PTR RETRYC,AX
0162                            WIREAD1:
0162  C6 06 0317 R 20                   MOV     BYTE PTR WIPAR+1,READ   ;SET READ FUNCTION
0167  E8 0334 R                         CALL    FIXDR                   ;READ ONE SECTOR
016A  A0 031A R                         MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
016D  24 F9                             AND     AL,0F9H
016F  3C 50                             CMP     AL,50H
0171  75 16                             JNZ     WIREAD2                 ;GO PERFORM RETRIES
0173  A1 0213 R                         MOV     AX,WORD PTR RETRYDEF
0176  A3 0215 R                         MOV     WORD PTR RETRYC,AX      ;SET RETRY DEFAULT VALUE
0179  81 06 031E R 0200                 ADD     WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
017F  FF 06 0318 R                      INC     WORD PTR WIPAR+2        ;SECTOR # +1
0183  49                                DEC     CX                      ;SECTOR COUNT -1
0184  75 DC                             JNZ     WIREAD1                 ;GO READ NEXT SECTOR
0186  E9 00C4 R                         JMP     EXIT
```

```
                               ;
0189                           WIREAD2:
0189  24 80                            AND     AL,CBUSY
018B  75 1C                            JNZ     WIREAD3                 ;CHECK DRIVE NOT READY
018D  FE 0E 0215 R                     DEC     BYTE PTR RETRYC
0191  79 CF                            JNS     WIREAD1                 ;PERFORM RETRY
0193  A0 0213 R                        MOV     AL,BYTE PTR RETRYDEF
0196  A2 0215 R                        MOV     BYTE PTR RETRYC,AL      ;SET RETRY DEFAULT VALUE
0199  FE 0E 0216 R                     DEC     BYTE PTR RETRYC+1
019D  78 1A                            JS      WIERR                   ;CHECK ERROR TYPE
019F  C6 06 0317 R 10                  MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
01A4  E8 0334 R                        CALL    FIXDR                   ;RESTORE DRIVE
01A7  EB B9                            JMP     WIREAD1                 ;GO READ AGAIN
                               ;
01A9                           WIREAD3:
01A9  E8 00DC R                        CALL    GETLEN
01AC  26: C7 47 12 0000                MOV     ES:WORD PTR COUNT[BX],00        ;SET NO. OF SECT.PROCESSED
01B2  B0 02                            MOV     AL,2                    ;SET DRIVE NOT READY STATUS
01B4  EB 1D                            JMP     SHORT   WIERR2
0:B6                           WIREAD4:
01B6  E9 00B1 R                        JMP     UNITERR
                               ;
                               ;
                               ;       ERROR ROUTINE
                               ;
01B9                           WIERR:
01B9  E8 00DC R                        CALL    GETLEN
01BC  26: 29 4F 12                     SUB     ES:WORD PTR COUNT[BX],CX        ;SET NO. OF PROCESSED SECT.
01C0  A0 031A R                        MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS REG.
01C3  D0 D8                            RCR     AL,1
01C5  72 23                            JC      WIERR8                  ;CONTROLLER ERROR
01C7  B1 04                            MOV     CL,4
01C9  D2 D8                            RCR     AL,CL
01CB  73 09                            JNC     WIERR3                  ;SEEK ERROR
01CD  D0 D8                            RCR     AL,1
01CF  72 09                            JC      WIERR4                  ;WRITE FAULT
01D1  B0 0C                    WIERR1: MOV     AL,12                   ;GENERAL FAILURE
01D3  E9 00BF R                WIERR2: JMP     ERREXIT                 ;STORE ERROR AND EXIT
                               ;
01D6  B0 06                    WIERR3: MOV     AL,6                    ;SEEK ERROR
01D8  EB F9                            JMP     SHORT   WIERR2
01DA  B0 0A                    WIERR4: MOV     AL,10                   ;WRITE FAULT
01DC  EB F5                            JMP     SHORT   WIERR2
01DE  B0 0B                    WIERR5: MOV     AL,11                   ;READ FAULT
01E0  EB F1                            JMP     SHORT   WIERR2
01E2  B0 08                    WIERR6: MOV     AL,8                    ;SECTOR NOT FOUND
01E4  EB ED                            JMP     SHORT   WIERR2
01E6  B0 04                    WIERR7: MOV     AL,4                    ;CRC ERROR
01E8  EB E9                            JMP     SHORT   WIERR2
01EA  A0 031B R                WIERR8: MOV     AL,BYTE PTR WIPAR+5     ;GET ERROR REGISTER
01ED  D0 D8                            RCR     AL,1
01EF  72 ED                            JC      WIERR5                  ;ADDR. MARK NOT FOUND
01F1  D0 D8                            RCR     AL,1
01F3  72 DC                            JC      WIERR1                  ;TRACK 0 ERROR
01F5  D0 D8                            RCR     AL,1
01F7  72 D8                            JC      WIERR1                  ;ABORTED COMMAND
01F9  D0 D8                            RCR     AL,1
01FB  D0 D8                            RCR     AL,1
01FD  72 E3                            JC      WIERR6                  ;ID NOT FOUND
01FF  D0 D8                            RCR     AL,1
0201  72 E3                            JC      WIERR7                  ;CRC ERROR IN ID FIELD
```

```
0203  D0 D8                       RCR     AL,1
0205  72 D7                       JC      WIERR5                  ;UNCORRECTABLE DATA
0207  D0 D8                       RCR     AL,1
0209  72 D7                       JC      WIERR6                  ;BAD BLOCK DETECTED
020B  EB C4                       JMP     SHORT   WIERR1
                          ;
                          ;
                          ;
                          ;
020D                      WIWRTC:
020D  E9 00B5 R                   JMP     LENERR
0210                      WIWRTD:
0210  E9 00B1 R                   JMP     UNITERR
                          ;
                          ;
0213  0000                RETRYDEF        DW      0               ;RETRY DEFAULT VALUE
0215  0000                RETRYC          DW      0               ;RETRY COUNT
0217  00                  WRTFLG          DB      0               ; 00=WRITE, FF=WRITE/VERIFY
                          ;
                          ;
                          ;       WRITE DATA
                          ;
0218                      WIWRT:
0218  C6 06 0217 R 00             MOV     BYTE PTR WRTFLG,0       ;SET WRITE DATA FLAG
021D                      WIWRT1:
021D  E8 00DC R                   CALL    GETLEN                  ;GET DRIVE REQU. STRUCT. LENGTH
0220  3C 16                       CMP     AL,22
0222  72 E9                       JB      WIWRTC                  ;BAD STRUCT. LENGTH
0224  E8 0305 R                   CALL    CHKDRIVE                ;CHECK IF UNIT AVAILABLE
0227  72 E7                       JC      WIWRTD                  ;UNIT ERROR
0229  E8 0321 R                   CALL    FIXREADY                ;CHECK IF DRIVE READY
022C  75 34                       JNZ     WIWRT3A
022E  A1 0213 R                   MOV     AX,WORD PTR RETRYDEF
0231  A3 0215 R                   MOV     WORD PTR RETRYC,AX      ;SET RETRY COUNT
0234                      WIWRT2:
0234  C6 06 0317 R 30             MOV     BYTE PTR WIPAR+1,WRITE  ;SET WRITE FUNCTION
0239  E8 0334 R                   CALL    FIXDR                   ;WRITE DATA
023C  A0 031A R                   MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
023F  24 F9                       AND     AL,0F9H
0241  3C 50                       CMP     AL,50H                  ;CHECK FOR ERROR
0243  75 2E                       JNZ     WIWRT4                  ;PERFORM RETRIES
0245  A1 0213 R                   MOV     AX,WORD PTR RETRYDEF
0248  A3 0215 R                   MOV     WORD PTR RETRYC,AX      ;SET RETRY DEFAULT VALUES
024B  80 3E 0217 R 00             CMP     BYTE PTR WRTFLG,0
0250  75 41                       JNZ     WIWRT5                  ;GO VERIFY DATA
0252                      WIWRT3:
0252  81 06 031E R 0200           ADD     WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
0258  FF 06 0318 R                INC     WORD PTR WIPAR+2        ;SECTOR NUMBER +1
025C  49                          DEC     CX                      ;SECTOR COUNT -1
025D  75 D5                       JNZ     WIWRT2                  ;GO WRITE NEXT SECTOR
025F  E9 00C4 R                   JMP     EXIT
0262                      WIWRT3A:
0262  E8 00DC R                   CALL    GETLEN
0265  26: C7 47 12 0000           MOV     ES:WORD PTR COUNT[BX],00        ;SET NO. OF PROCESSED SECT.
026B  B0 02                       MOV     AL,2                    ;DRIVE NOT READY
026D  E9 00BF R                   JMP     ERREXIT
0270                      WIWRT3B:
0270  E9 01B9 R                   JMP     WIERR                   ;CHECK STATUS TYPE
0273                      WIWRT4:
0273  24 80                       AND     AL,CBUSY                ;CHECK IF DRIVE NOT READY
0275  75 EB                       JNZ     WIWRT3A
```

```
0277  FE 0E 0215 R                      DEC    BYTE PTR RETRYC
027B  79 B7                             JNS    WIWRT2                  ;WRITE AGAIN
027D  A0 0213 R                         MOV    AL,BYTE PTR RETRYDEF
0280  A2 0215 R                         MOV    BYTE PTR RETRYC,AL      ;SET RETRY DEFAULT VALUE
0283  FE 0E 0216 R                      DEC    BYTE PTR RETRYC+1
0287  78 E7                             JS     WIWRT3B                 ;WRITE ERROR
0289  C6 06 0317 R 10                   MOV    BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
028E  E8 0334 R                         CALL   FIXDR                   ;RESTORE DRIVE
0291  EB A1                             JMP    WIWRT2                  ;WRITE NEXT SECTOR
0293                             WIWRT5:
0293  8E 06 031C R                      MOV    ES,WORD PTR WIPAR+6
0297  8B 3E 031E R                      MOV    DI,WORD PTR WIPAR+8     ;SAVE DATA BUFFER ADDR.
029B                             WIWRT6:
029B  A0 0320 R                         MOV    AL,BYTE PTR WIPAR+10    ;GET ACTUAL SDH REG. CONTENTS
029E  E6 C6                             OUT    SDH,AL
02A0  B0 20                             MOV    AL,READ
02A2  E6 C7                             OUT    COMND,AL                ;OUTPUT READ FUNCTION
02A4  E8 03BD R                         CALL   WAIT
02A7  E4 C6                             IN     AL,SDH
02A9  0C 18                             OR     AL,18H
02AB  E6 C6                             OUT    SDH,AL                  ;CLEAR DRIVE LAMP
02AD  51                                PUSH   CX
02AE  B9 0200                           MOV    CX,512
02B1                             WIWRT6A:
02B1  E4 C0                             IN     AL,DATA                 ;GET READ DATA
02B3  AE                                SCASB                          ;COMPARE WITH DATA WRITTEN
02B4  E1 FB                             LOOPZ  WIWRT6A
02B6  75 25                             JNZ    WIWRT7
02B8                             WIWRT6B:
02B8  59                                POP    CX
02B9  A0 031A R                         MOV    AL,BYTE PTR WIPAR+4     ;GET STATUS
02BC  24 F9                             AND    AL,0F9H
02BE  3C 50                             CMP    AL,50H
02C0  74 2B                             JZ     WIWRT8
02C2  FE 0E 0215 R                      DEC    BYTE PTR RETRYC
02C6  75 D3                             JNZ    WIWRT6                  ;PERFORM RETRY
02C8  C6 06 0215 R 05                   MOV    BYTE PTR RETRYC,5
02CD  FE 0E 0216 R                      DEC    BYTE PTR RETRYC+1
02D1  74 27                             JZ     WIWRTB                  ;READ ERROR
02D3  C6 06 0317 R 10                   MOV    BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
02D8  E8 0334 R                         CALL   FIXDR                   ;RESTORE DRIVE
02DB  EB BE                             JMP    WIWRT6
02DD                             WIWRT7:
02DD  83 F9 00                          CMP    CX,0
02E0  74 04                             JZ     WIWRT7B
02E2                             WIWRT7A:
02E2  E4 C0                             IN     AL,DATA
02E4  E2 FC                             LOOP   WIWRT7A
02E6                             WIWRT7B:
02E6  80 0E 031A R 20                   OR     BYTE PTR WIPAR+4,DWRFA  ;SET WRITE FAULT ERROR
02EB  EB CB                             JMP    SHORT   WIWRT6B
02ED                             WIWRT8:
02ED  C7 06 0215 R 0505                 MOV    WORD PTR RETRYC,0505H   ;SET RETRY DEFAULT VALUE
02F3  E9 0252 R                         JMP    WIWRT3
02F6                             WIWRTA:
02F6  59                                POP    CX
02F7  E9 01DA R                         JMP    WIERR4                  ;DATA VERIFY ERROR
02FA                             WIWRTB:
02FA  E9 01B9 R                         JMP    WIERR                   ;SET ERROR STATUS
```

```
                            ;
                            ;
                            ;       WRITE DATA AND VERIFY
                            ;
02FD                        WIWRTV:
02FD  C6 06 0217 R FF               MOV     BYTE PTR WRTFLG,0FFH    ;SET WRITE/VERIFY FLAG
0302  E9 021D R                     JMP     WIWRT1
                            ;
                            ;
                            ;      ROUTINE TO CHECK IF UNIT AVAILABLE
                            ;         EXIT: CARRY ON=UNIT ERROR
                            ;
0305                        CHKDRIVE:
0305  50                            PUSH    AX
0306  A0 0316 R                     MOV     AL,BYTE PTR WIPAR       ;GET UNIT NO. TO WORK WITH
0309  8A 26 000A R                  MOV     AH,BYTE PTR WINDEV+10   ;GET NO.OF UNITS IN SYSTEM
030D  3A E0                         CMP     AH,AL
030F  58                            POP     AX
0310  76 02                         JBE     CHKDRIVE1
0312  F8                            CLC
0313  C3                            RET
0314                        CHKDRIVE1:
0314  F9                            STC
0315  C3                            RET
```

```
                           ;
                           ;*********************************************
                           ;*                                           *
                           ;*      PERIPHERAL INTERFACE MODULE (PIM)    *
                           ;*                                           *
                           ;*             WINCHESTER DISK               *
                           ;*                                           *
                           ;*********************************************
                           ;
                           ;UNIT 0= HEAD 0 AND 1
                           ;UNIT 1= HEAD 2 AND 3
                           ;
                           ;
                           ;          WINCHESTER DISK PARAMETER BLOCK
                           ;          ******************************
                           ;
                           ;
0316  00                   WIPAR   DB      0            ; WIPAR + 0     DISK UNIT
0317  10                           DB      REST         ; WIPAR + 1     FUNCTION
0318  0000                         DW      0            ; WIPAR + 2     SECTOR LO
                                                        ; WIPAR + 3     SECTOR HI
031A  00                           DB      0            ; WIPAR + 4     STATUS 1
031B  00                           DB      0            ; WIPAR + 5     STATUS 2
031C  0000                         DW      0            ; WIPAR + 6     BUFFER ADDR.(SEGMENT)
031E  0000                         DW      0            ; WIPAR + 8     BUFFER ADDR.(OFFSET)
0320  00                           DB      0            ; WIPAR + 10    ACTUAL SDH REG. CONTENTS
                           ;
                           ;
                           ;
                           ;*****************************************
                           ;*                                       *
                           ;*  CHECK IF WINCHESTER DRIVE IS         *
                           ;*   CONNECTED AND POWERED ON.           *
                           ;*                                       *
                           ;*   EXIT: ZERO FLAG ON = DRIVE READY    *
                           ;*                                       *
                           ;*****************************************
                           ;
0321                       FIXREADY:
0321  B0 55                        MOV     AL,55H
0323  E6 C4                        OUT     CYLLO,AL             ;OUTPUT PATTERN TO R/W PORT
0325  B0 AA                        MOV     AL,0AAH
0327  E6 C3                        OUT     SECNO,AL
0329  E4 C4                        IN      AL,CYLLO             ;READ PATTERN BACK AND COMPARE
032B  3C 55                        CMP     AL,55H
032D  75 04                        JNZ     FIXREADY1
032F  E4 C3                        IN      AL,SECNO
0331  3C AA                        CMP     AL,0AAH
0333                       FIXREADY1:
0333  C3                           RET
```

```
                              ;*******************************************
                              ;*                                         *
                              ;*     WINCHESTER DISK DRIVER              *
                              ;*                                         *
                              ;*      ENTRY: PARAMETER BLOCK FILLED UP   *
                              ;*      EXIT:  STATUS BYTES IN PARAM.      *
                              ;*             BLOCK UPDATED AND ALL       *
                              ;*             REGISTERS SAVED.            *
                              ;*******************************************
0334  50                      FIXDR:  PUSH    AX
0335  53                              PUSH    BX
0336  51                              PUSH    CX
0337  52                              PUSH    DX
0338  A1 0318 R                       MOV     AX,WORD PTR WIPAR+2     ;GET LOGIC SECTOR NUMBER
033B  B9 0011                         MOV     CX,17
033E  BA 0000                         MOV     DX,0
0341  F7 F1                           DIV     CX                      ;CALCULATE CYL/HEAD
0343  50                              PUSH    AX
0344  8A C2                           MOV     AL,DL
0346  E6 C3                           OUT     SECNO,AL                ;SET SECTOR NUMBER
0348  8A 1E 0316 R                    MOV     BL,BYTE PTR WIPAR       ;GET DISK UNIT #
034C  8A FB                           MOV     BH,BL
034E  81 E3 0601                      AND     BX,0601H
0352  D0 C7                           ROL     BH,1
0354  0A DF                           OR      BL,BH
0356  D0 C3                           ROL     BL,1
0358  58                              POP     AX
0359  50                              PUSH    AX
035A  24 01                           AND     AL,01H                  ;GET HEAD BIT
035C  0A C3                           OR      AL,BL
035E  0C A0                           OR      AL,SDHREG               ;ECC/CRC AND BYTES PER SECTOR
0360  E6 C6                           OUT     SDH,AL                  ;SET ECC/CRC-BYTES/SECT-DRIVE-HEAD
0362  A2 0320 R                       MOV     BYTE PTR WIPAR+10,AL    ;SAVE ACTUAL SDH REG. CONTENTS
0365  58                              POP     AX
0366  D1 C8                           ROR     AX,1
0368  E6 C4                           OUT     CYLLO,AL                ;SET CYLINDER LOW
036A  80 E4 03                        AND     AH,03H
036D  8A C4                           MOV     AL,AH
036F  E6 C5                           OUT     CYLHI,AL                ;SET CYLINDER HIGH
0371  E4 C7                           IN      AL,STAT                 ;GET DISK STATUS
0373  A2 031A R                       MOV     BYTE PTR WIPAR+4,AL
0376  24 80                           AND     AL,CBUSY                ;CHECK IF CONTROLLER BUSY
0378  75 16                           JNZ     FIXD3
037A  A0 0317 R                       MOV     AL,BYTE PTR WIPAR+1
037D  E6 C7                           OUT     COMND,AL                ;SET FUNCTION
037F  24 F0                           AND     AL,0F0H
0381  3C 20                           CMP     AL,READ
0383  74 16                           JZ      RD                      ;GO READ DATA
0385  3C 30                           CMP     AL,WRITE
0387  74 4E                           JZ      WR                      ;GO WRITE DATA
0389  3C 50                           CMP     AL,FORMAT
038B  74 46                           JZ      WRB                     ;GO FORMAT ONE TRACK
038D  EB 5C 90                        JMP     WR2                     ;SEEK OR RESTORE
0390  E4 C6                   FIXD3:  IN      AL,SDH
0392  0C 18                           OR      AL,18H
0394  E6 C6                           OUT     SDH,AL                  ;CLEAR DISK LAMP
0396  5A                              POP     DX
0397  59                              POP     CX
0398  5B                              POP     BX
0399  58                              POP     AX
039A  C3                              RET
```

```
                        ;       ****************************
                        ;       *    READ ROUTINE          *
                        ;       ****************************
                        ;
039B  E8 03BD R         RD:     CALL    WAIT                    ;WAIT UNTIL READ COMPLETE
039E  1E                        PUSH    DS
039F  8B 1E 031E R              MOV     BX,WORD PTR WIPAR+8     ;GET OFFSET
03A3  8E 1E 031C R              MOV     DS,WORD PTR WIPAR+6     ;GET SEGMENT ADDR.
03A7  B9 0200                   MOV     CX,512                  ;INPUT COUNT
03AA  E4 C0             RD2:    IN      AL,DATA                 ;INPUT DATA
03AC  88 07                     MOV     BYTE PTR[BX],AL         ;SAVE INPUT
03AE  43                        INC     BX
03AF  E0 F9                     LOOPNZ  RD2          ;CONTINUE UNTIL ALL BYTES IN BUFFER
                                                     ;BUT STOP BEFORE BUFFER ADDR. WRAP AROUND
03B1  83 F9 00                  CMP     CX,0
03B4  74 04                     JZ      RD4
03B6  E4 C0             RD3:    IN      AL,DATA                 ;CLEAR CONTROLLER BUFFER
03B8  E2 FC                     LOOP    RD3
03BA  1F                RD4:    POP     DS
03BB  EB D3                     JMP     SHORT   FIXD3
                        ;
                        ;
                        ;       ****************************
                        ;       *    WAIT ROUTINE          *
                        ;       ****************************
                        ;
03BD  E4 C7             WAIT:   IN      AL,STAT                 ;GET STATUS
03BF  24 80                     AND     AL,CBUSY
03C1  75 FA                     JNZ     WAIT                    ;LOOP UNTIL DISK READY
03C3  E4 C7                     IN      AL,STAT
03C5  A2 031A R                 MOV     BYTE PTR WIPAR+4,AL     ;SAVE STATUS
03C8  D0 D8                     RCR     AL,1
03CA  72 01                     JC      ER1                     ;JUMP IF ERROR CONDITION
03CC  C3                        RET
                        ;
03CD  E4 C1             ER1:    IN      AL,ERROR                ;GET ERROR STATUS
03CF  A2 031B R                 MOV     BYTE PTR WIPAR+5,AL     ;SAVE STATUS
03D2  C3                        RET
                        ;
                        ;       ****************************
                        ;       *    WRITE ROUTINE         *
                        ;       ****************************
                        ;
03D3  B0 11             WR0:    MOV     AL,17
03D5  E6 C2                     OUT     SECNT,AL                ;SET SECT COUNT FOR FORMAT
                        ;
03D7  1E                WR:     PUSH    DS
03D8  8B 1E 031E R              MOV     BX,WORD PTR WIPAR+8     ;BUFFER ADDR.(OFFSET)
03DC  8E 1E 031C R              MOV     DS,WORD PTR WIPAR+6     ;BUFFER ADDR.(SEGMENT)
03E0  B9 0200                   MOV     CX,512                  ;INPUT COUNT
03E3  8A 07             WR1:    MOV     AL,BYTE PTR[BX]         ;GET BYTE FROM BUFFER
03E5  E6 C0                     OUT     DATA,AL                 ;OUTPUT DATA
03E7  43                        INC     BX
03E8  E2 F9                     LOOP    WR1
03EA  1F                        POP     DS
03EB  E8 03BD R         WR2:    CALL    WAIT                    ;WAIT UNTIL FUNCT. COMPLETE
03EE  EB A0                     JMP     SHORT   FIXD3
```

```
                           ;
03F0                       WIINIT:
03F0  E8 00DC R                    CALL    GETLEN                  ;GET DRIVE REQU. STRUCT.LENGTH
03F3  3C 16                        CMP     AL,22
03F5  72 5A                        JB      WIINIT1                 ;BAD STRUCT. LENGTH
03F7  06                           PUSH    ES
03F8  53                           PUSH    BX
03F9  B8 0000                      MOV     AX,0
03FC  8E C0                        MOV     ES,AX
03FE  BB 0416                      MOV     BX,WINCHDRIVES          ;ADDR. OF SYSTEM PARAM.
0401  26: 8A 07                    MOV     AL,ES:BYTE PTR [BX]     ;GET NO. OF DRIVES ON SYSTEM
0404  D0 C0                        ROL     AL,1                    ;MAKE NO. OF UNITS
0406  43                           INC     BX
0407  26: 8B 0F                    MOV     CX,ES:WORD PTR [BX]     ;GET RETRY COUNTS
040A  5B                           POP     BX
040B  07                           POP     ES
040C  88 2E 0213 R                 MOV     BYTE PTR RETRYDEF,CH
0410  88 0E 0214 R                 MOV     BYTE PTR RETRYDEF+1,CL  ;SAVE RETRY COUNTS
0414  A2 000A R                    MOV     BYTE PTR WINDEV+10,AL   ;SET NUMBER OF UNITS
0417  26: 88 47 0D                 MOV     ES:BYTE PTR MEDIA[BX],AL
041B  26: C7 47 0E 03F0 R          MOV     ES:WORD PTR TRANS[BX],OFFSET WIINIT
0421  26: 8C 4F 10                 MOV     ES:WORD PTR TRANS+2[BX],CS
0425  26: C7 47 12 000E R          MOV     ES:WORD PTR COUNT[BX],OFFSET WIBPB      ;ADDR. OF BPB ARRAY
042B  26: 8C 4F 14                 MOV     ES:WORD PTR COUNT+2[BX],CS
042F  C7 06 0095 R 00C4 R          MOV     WORD PTR WITBL,OFFSET EXIT
0435  E8 0321 R                    CALL    FIXREADY
0438  75 1F                        JNZ     WIINIT4
043A  C6 06 0317 R 10              MOV     BYTE PTR WIPAR+1,REST
043F  80 26 0316 R 01              AND     BYTE PTR WIPAR,01
0444  E8 0334 R                    CALL    FIXDR                   ;RESTORE DRIVE
0447  80 3E 031A R 50              CMP     BYTE PTR WIPAR+4,50H
044C  75 06                        JNZ     WIINIT2                 ;ERROR CONDITION
044E  E9 00C4 R                    JMP     EXIT
0451                       WIINIT1:
0451  E9 00B5 R                    JMP     LENERR
0454                       WIINIT2:
0454  A0 031A R                    MOV     AL,BYTE PTR WIPAR+4
0457  24 80                        AND     AL,CBUSY
0459                       WIINIT4:
0459  B0 02                        MOV     AL,2                    ;DRIVE NOT READY
045B  75 02                        JNZ     WIINIT3
045D  B0 0C                        MOV     AL,12                   ;GENERAL FAILURE
045F                       WIINIT3:
045F  E9 00BF R                    JMP     ERREXIT                 ;SAVE STATUS AND EXIT
                           ;
                           ;
                           ;
0462                       CSEG   ENDS
                           ;
                           ;
                           END     BEGIN
```

```
Segments and groups:

                N a m e             Size    align   combine class

CSEG . . . . . . . . . . . . . .    0462    PARA    NONE

Symbols:

                N a m e             Type    Value   Attr

ABC. . . . . . . . . . . . . . .    Number  0004
BBD. . . . . . . . . . . . . . .    Number  0080
BEGIN. . . . . . . . . . . . . .    L NEAR  0000    CSEG
BUSEXIT. . . . . . . . . . . . .    L NEAR  00AD    CSEG
CBUSY. . . . . . . . . . . . . .    Number  0080
CDRQ . . . . . . . . . . . . . .    Number  0008
CERR . . . . . . . . . . . . . .    Number  0001
CHKDRIVE . . . . . . . . . . . .    L NEAR  0305    CSEG
CHKDRIVE1. . . . . . . . . . . .    L NEAR  0314    CSEG
CMD. . . . . . . . . . . . . . .    Number  0002
CMDERR . . . . . . . . . . . . .    L NEAR  00BD    CSEG
CMDLEN . . . . . . . . . . . . .    Number  0000
COMND. . . . . . . . . . . . . .    Number  00C7
CORRD. . . . . . . . . . . . . .    Number  0004
COUNT. . . . . . . . . . . . . .    Number  0012
CRCID. . . . . . . . . . . . . .    Number  0020
CYLHI. . . . . . . . . . . . . .    Number  00C5
CYLLO. . . . . . . . . . . . . .    Number  00C4
DAMNFD . . . . . . . . . . . . .    Number  0001
DATA . . . . . . . . . . . . . .    Alias   HBASE
DREADY . . . . . . . . . . . . .    Number  0040
DSEEC. . . . . . . . . . . . . .    Number  0010
DWRFA. . . . . . . . . . . . . .    Number  0020
ENTRY. . . . . . . . . . . . . .    L NEAR  003E    CSEG
ENTRY1 . . . . . . . . . . . . .    L NEAR  0067    CSEG
ER1. . . . . . . . . . . . . . .    L NEAR  03CD    CSEG
ERREXIT. . . . . . . . . . . . .    L NEAR  00BF    CSEG
ERROR. . . . . . . . . . . . . .    Number  00C1
EXIT . . . . . . . . . . . . . .    L NEAR  00C4    CSEG
EXIT1. . . . . . . . . . . . . .    L NEAR  00C6    CSEG
EXITP. . . . . . . . . . . . . .    F PROC  00C4    CSEG    Length =0018
FIXD3. . . . . . . . . . . . . .    L NEAR  0398    CSEG
FIXDR. . . . . . . . . . . . . .    L NEAR  0334    CSEG
FIXREADY . . . . . . . . . . . .    L NEAR  0321    CSEG
FIXREADY1. . . . . . . . . . . .    L NEAR  0333    CSEG
FORMAT . . . . . . . . . . . . .    Number  0050
GETBPB . . . . . . . . . . . . .    L NEAR  010C    CSEG
GETBPB1. . . . . . . . . . . . .    L NEAR  0133    CSEG
GETBPB2. . . . . . . . . . . . .    L NEAR  0131    CSEG
GETLEN . . . . . . . . . . . . .    L NEAR  00DC    CSEG
HBASE. . . . . . . . . . . . . .    Number  00C0
IDNFD. . . . . . . . . . . . . .    Number  0010
INTERRUPT. . . . . . . . . . . .    L NEAR  003A    CSEG
LENERR . . . . . . . . . . . . .    L NEAR  00B5    CSEG
MEDERR . . . . . . . . . . . . .    L NEAR  00B9    CSEG
MEDIA. . . . . . . . . . . . . .    Number  000D
MEDIAC . . . . . . . . . . . . .    L NEAR  00E8    CSEG
MEDIAC1. . . . . . . . . . . . .    L NEAR  0105    CSEG
PTRSAV . . . . . . . . . . . . .    L WORD  0036    CSEG
RD . . . . . . . . . . . . . . .    L NEAR  039B    CSEG
```

```
RD2. . . . . . . . . . . . . . . .     L NEAR  03AA    CSEG
RD3. . . . . . . . . . . . . . . .     L NEAR  03B6    CSEG
RD4. . . . . . . . . . . . . . . .     L NEAR  03BA    CSEG
READ . . . . . . . . . . . . . . .     Number  0020
REST . . . . . . . . . . . . . . .     Number  0010
RETRYC . . . . . . . . . . . . . .     L WORD  0215    CSEG
RETRYDEF . . . . . . . . . . . . .     L WORD  0213    CSEG
SDH. . . . . . . . . . . . . . . .     Number  00C6
SDHREG . . . . . . . . . . . . . .     Number  00A0
SECNO. . . . . . . . . . . . . . .     Number  00C3
SECNT. . . . . . . . . . . . . . .     Number  00C2
SEEK . . . . . . . . . . . . . . .     Number  0070
START. . . . . . . . . . . . . . .     Number  0014
STAT . . . . . . . . . . . . . . .     Number  00C7
STATUS . . . . . . . . . . . . . .     Number  0003
STRATE . . . . . . . . . . . . . .     Number  000B
STRATEGY . . . . . . . . . . . . .     L NEAR  002B    CSEG
STRATP . . . . . . . . . . . . . .     F PROC  002B    CSEG    Length =000B
TR0. . . . . . . . . . . . . . . .     Number  0002
TRANS. . . . . . . . . . . . . . .     Number  000E
UNCOR. . . . . . . . . . . . . . .     Number  0040
UNIT . . . . . . . . . . . . . . .     Number  0001
UNITERR. . . . . . . . . . . . . .     L NEAR  00B1    CSEG
UNITTAB. . . . . . . . . . . . . .     L BYTE  0143    CSEG    Length =0008
WAIT . . . . . . . . . . . . . . .     L NEAR  03BD    CSEG
WIBPB. . . . . . . . . . . . . . .     L WORD  000E    CSEG
WIBPB1 . . . . . . . . . . . . . .     L WORD  001E    CSEG
WIERR. . . . . . . . . . . . . . .     L NEAR  01B9    CSEG
WIERR1 . . . . . . . . . . . . . .     L NEAR  01D1    CSEG
WIERR2 . . . . . . . . . . . . . .     L NEAR  01D3    CSEG
WIERR3 . . . . . . . . . . . . . .     L NEAR  01D6    CSEG
WIERR4 . . . . . . . . . . . . . .     L NEAR  01DA    CSEG
WIERR5 . . . . . . . . . . . . . .     L NEAR  01DE    CSEG
WIERR6 . . . . . . . . . . . . . .     L NEAR  01E2    CSEG
WIERR7 . . . . . . . . . . . . . .     L NEAR  01E6    CSEG
WIERR8 . . . . . . . . . . . . . .     L NEAR  01EA    CSEG
WIINIT . . . . . . . . . . . . . .     L NEAR  03F0    CSEG
WIINIT1. . . . . . . . . . . . . .     L NEAR  0451    CSEG
WIINIT2. . . . . . . . . . . . . .     L NEAR  0454    CSEG
WIINIT3. . . . . . . . . . . . . .     L NEAR  045F    CSEG
WIINIT4. . . . . . . . . . . . . .     L NEAR  0459    CSEG
WIINRETRIES. . . . . . . . . . . .     Number  0418
WINCHDRIVES. . . . . . . . . . . .     Number  0416
WINDEV . . . . . . . . . . . . . .     L WORD  0000    CSEG
WIOUTRETRIES . . . . . . . . . . .     Number  0417
WIPAR. . . . . . . . . . . . . . .     L BYTE  0316    CSEG
WIREAD . . . . . . . . . . . . . .     L NEAR  014B    CSEG
WIREAD1. . . . . . . . . . . . . .     L NEAR  0162    CSEG
WIREAD2. . . . . . . . . . . . . .     L NEAR  0189    CSEG
WIREAD3. . . . . . . . . . . . . .     L NEAR  01A9    CSEG
WIREAD4. . . . . . . . . . . . . .     L NEAR  01B6    CSEG
WITBL. . . . . . . . . . . . . . .     L WORD  0095    CSEG
WIWRT. . . . . . . . . . . . . . .     L NEAR  0218    CSEG
WIWRT1 . . . . . . . . . . . . . .     L NEAR  021D    CSEG
WIWRT2 . . . . . . . . . . . . . .     L NEAR  0234    CSEG
WIWRT3 . . . . . . . . . . . . . .     L NEAR  0252    CSEG
WIWRT3A. . . . . . . . . . . . . .     L NEAR  0262    CSEG
WIWRT3B. . . . . . . . . . . . . .     L NEAR  0270    CSEG
WIWRT4 . . . . . . . . . . . . . .     L NEAR  0273    CSEG
WIWRT5 . . . . . . . . . . . . . .     L NEAR  0293    CSEG
WIWRT6 . . . . . . . . . . . . . .     L NEAR  029B    CSEG
```

```
WIWRT6A. . . . . . . . . . . . .        L NEAR  02B1    CSEG
WIWRT6B. . . . . . . . . . . . .        L NEAR  02B8    CSEG
WIWRT7 . . . . . . . . . . . . .        L NEAR  02DD    CSEG
WIWRT7A. . . . . . . . . . . . .        L NEAR  02E2    CSEG
WIWRT7B. . . . . . . . . . . . .        L NEAR  02E6    CSEG
WIWRT8 . . . . . . . . . . . . .        L NEAR  02ED    CSEG
WIWRTA . . . . . . . . . . . . .        L NEAR  02F6    CSEG
WIWRTB . . . . . . . . . . . . .        L NEAR  02FA    CSEG
WIWRTC . . . . . . . . . . . . .        L NEAR  020D    CSEG
WIWRTD . . . . . . . . . . . . .        L NEAR  0210    CSEG
WIWRTV . . . . . . . . . . . . .        L NEAR  02FD    CSEG
WPC. . . . . . . . . . . . . . .        Number  00C1
WR . . . . . . . . . . . . . . .        L NEAR  03D7    CSEG
WR0. . . . . . . . . . . . . . .        L NEAR  03D3    CSEG
WR1. . . . . . . . . . . . . . .        L NEAR  03E3    CSEG
WR2. . . . . . . . . . . . . . .        L NEAR  03EB    CSEG
WRITE. . . . . . . . . . . . . .        Number  0030
WRTFLG . . . . . . . . . . . . .        L BYTE  0217    CSEG
```

LINK MAP OF IO.SYS (LINKED MODULES: DUMMY, IOSYS01, KBD—DRV, DSKDRV, BASINIT, SYSINIT, SYSIMES, FWVERRD)

```
Start    Stop     Length  Name         Class

00000H   003FFH   0400H   DUMMY        DUM
00400H   02793H   2394H   CSEG         CODE
027A0H   02F53H   07B4H   SYSINITSEG   SYSTEM—INIT
02F60H   02F7DH   001EH   FWSEG        FRCODE
```

```
     Address        Publics by Name

     027A:07A0      BADCOM
     027A:078E      BADLD
     027A:0746      BADOPM
     027A:076D      BADSIZ
     027A:0012      BUFFERS
     0040:058C      BUS—EXIT
     0040:08BB      CHRTRN
     0040:0613      CHTRANS
     0040:0223      CLEAR—1
     0040:0233      CLEAR—2
     027A:076A      CRLFM
     027A:0005      CURRENT—DOS—LOCATION
     0040:0227      DEC—SIGN—1
     0040:0237      DEC—SIGN—2
     027A:0011      DEFAULT—DRIVE
     0040:1582      DEF—FUN
     027A:000B      DEVICE—LIST
     0040:0476      DEVSTART
     0040:2197      DREND
     0040:04C8      DRVMAX
     0040:1EAE      DSK—INT
     0040:0B79      ED
     0040:0590      ERROR—0
     0040:0594      ERROR—1
     0040:05B8      ERROR—10
     0040:05BC      ERROR—11
     0040:05C0      ERROR—12
     0040:0598      ERROR—2
     0040:059C      ERROR—3
```

Address	Publics by Name
0040:05A0	ERROR—4
0040:05A4	ERROR—5
0040:05A8	ERROR—6
0040:05AC	ERROR—7
0040:05B0	ERROR—8
0040:05B4	ERROR—9
0040:05C2	ERR—EXIT
0040:05C7	EXIT
0040:05C9	EXIT1
027A:0013	FILES
027A:0009	FINAL—DOS—LOCATION
0040:2274	FIRM—MESS
0040:1362	FLAG—BUF
0040:0013	FLOPPY—DRIVES
0040:0011	FLTAB
0040:0015	FL—IN—RETRIES
0040:0014	FL—OUT—RETRIES
0040:1364	FUNCOFF
0040:001D	FUNCTBL
0040:22BC	HWINIT
0040:05DB	I29—HANDLER
0040:05FB	INT—TRAP
0040:16B7	KBD—OUT
0040:021D	KBD—TT
0040:140C	KEY—IN
0040:1553	KEY—INIT
0040:1368	KEY—IN—FL
0040:14A4	KEY—ND—IN
0040:152A	KEY—ST
0040:1360	LANGUAGE
027A:000F	MEMORY—SIZE
0040:060D	MONO—COLOR
0040:0897	OUTCTR
0040:0538	PTRSAV
02F6:0000	READ—FW—VER
0040:1366	REMNUM
0040:0546	RE—INIT
027A:0000	SYSINIT
027A:07B4	SYSSIZE

Address	Publics by Value
0040:0011	FLTAB
0040:0013	FLOPPY—DRIVES
0040:0014	FL—OUT—RETRIES
0040:0015	FL—IN—RETRIES
0040:001D	FUNCTBL
0040:021D	KBD—TT
0040:0223	CLEAR—1
0040:0227	DEC—SIGN—1
0040:0233	CLEAR—2
0040:0237	DEC—SIGN—2
0040:0476	DEVSTART
0040:04C8	DRVMAX
0040:0538	PTRSAV
0040:0546	RE—INIT
0040:058C	BUS—EXIT
0040:0590	ERROR—0
0040:0594	ERROR—1
0040:0598	ERROR—2
0040:059C	ERROR—3
0040:05A0	ERROR—4
0040:05A4	ERROR—5
0040:05A8	ERROR—6
0040:05AC	ERROR—7
0040:05B0	ERROR—8
0040:05B4	ERROR—9
0040:05B8	ERROR—10
0040:05BC	ERROR—11
0040:05C0	ERROR—12
0040:05C2	ERR—EXIT
0040:05C7	EXIT
0040:05C9	EXIT1
0040:05DB	I29—HANDLER
0040:05FB	INT—TRAP
0040:060D	MONO—COLOR
0040:0613	CHTRANS
0040:0897	OUTCTR
0040:08BB	CHRTRN
0040:0B79	ED
0040:1360	LANGUAGE
0040:1362	FLAG—BUF
0040:1364	FUNCOFF
0040:1366	REMNUM

Address	Publics by Value
0040:1368	KEY—IN—FL
0040:140C	KEY—IN
0040:14A4	KEY—ND—IN
0040:152A	KEY—ST
0040:1553	KEY—INIT
0040:1582	DEF—FUN
0040:16B7	KBD—OUT
0040:1EAE	DSK—INT
0040:2197	DREND
0040:2274	FIRM—MESS
0040:22BC	HWINIT
027A:0000	SYSINIT
027A:0005	CURRENT—DOS—LOCATION
027A:0009	FINAL—DOS—LOCATION
027A:000B	DEVICE—LIST
027A:000F	MEMORY—SIZE
027A:0011	DEFAULT—DRIVE
027A:0012	BUFFERS
027A:0013	FILES
027A:0746	BADOPM
027A:076A	CRLFM
027A:076D	BADSIZ
027A:078E	BADLD
027A:07A0	BADCOM
027A:07B4	SYSSIZE
02F6:0000	READ—FW—VER

```
                        C          INCLUDE FLXPIND.ASM
                        C
                        C   ;##################
                        C   ;### MODE FLAGS ###
                        C   ;##################
                        C   ;
                        C   ;
 = 0002                 C   WT48ON96  EQU   02H          ; Bit 1 set in CONFIG_FLAGS enables writing
                        C   ;                            ; to 48TPI diskettes on 96TPI drives
                        C   ;
                        C   ;###################
                        C   ;###  I/O PORTS  ###
                        C   ;###################
                        C   ;
                        C   ;
                        C   ; FDC
                        C   ; ---
                        C   ;
 = 0051                 C   DCOMD    EQU     51H         ; DISK COMMAND PORT
 = 0050                 C   DSTAT    EQU     50H         ; DISK STATUS PORT
 = 0051                 C   FDCRA    EQU     51H         ; READ DMA FROM FDC PORT
                        C   ;
                        C   ;
                        C   ; DMA
                        C   ; ---
                        C   ;
 = 002A                 C   DMAMB    EQU     2AH         ; WRITE SINGLE MASK REGISTER BIT
 = 002B                 C   DMAMD    EQU     2BH         ; DMA MODE PORT
 = 0026                 C   COAD     EQU     26H         ; DMA ADDR PORT
 = 0027                 C   COTC     EQU     27H         ; DMA LENGTH PORT
                        C   ;
                        C   ;
                        C   ; SYSTEM STATUS
                        C   ; -------------
                        C   ;
 = 0013                 C   SYSSTA  EQU     13H          ; SYSTEM STATUS PORT
 = 0014                 C   MOTORON EQU     14H          ; MOTOR ON PORT
                        C   ;
                        C   ;
                        C   ; BANK SELECT
                        C   ; -----------
                        C   ;
 = 00E0                 C   BANK     EQU     0E0H        ; BANK SELECT E0 :    0K -  64K
                        C   ;
                        C   ;
                        C   ;#####################
                        C   ;###  FDC COMMANDS  ###
                        C   ;#####################
                        C   ;
                        C   ;
 = 0002                 C   READTRK EQU     02H          ; READ TRACK COMMAND
 = 0003                 C   SPECIFY EQU     03H          ; SPECIFY COMMAND
 = 0005                 C   WRITDAT EQU     05H          ; WRITE DATA COMMAND
 = 0006                 C   READDAT EQU     06H          ; READ DATA COMMAND
 = 0007                 C   RESTORE EQU     07H          ; RESTORE COMMAND
 = 0008                 C   FDCSIS  EQU     08H          ; SENSE INTERRUPT STATUS
 = 000A                 C   IDREAD  EQU     0AH          ; READ ID COMMAND
 = 000D                 C   WRITFMT EQU     0DH          ; FORMAT A TRACK
 = 000F                 C   SEEKTRK EQU     0FH          ; SEEK A TRACK
```

```
                     C   ;
                     C   ;**********************
                     C   ;***  FDC VARIABLES  ***
                     C   ;**********************
                     C   ;
                     C   ;
0007  00             C   CYLMODE DB     00          ; 0 = CYLINDER MODE, 1 = not CYLINDER MODE
0008  00             C   DRV     DB     00          ; DRIVE NUMBER
0009  00             C   HEAD    DB     00          ; HEAD NUMBER
000A  00             C   TRACK   DB     00          ; TRACK NUMBER
000B  00             C   SECTOR  DB     00          ; SECTOR NUMBER
                     C                              ;
000C  0000           C   SECCNT  DW     0000        ; Number of sectors for I/O
                     C   ;
                     C   ;
000E  00             C   COMSTR  DB     00          ; COMMAND STRING LENGTH
000F  00             C           DB     00          ; COMMAND STRING (max. 9 bytes)
0010  00             C           DB     00          ;
0011  00             C           DB     00          ;
0012  00             C           DB     00          ;
0013  00             C           DB     00          ;
0014  00             C           DB     00          ;
0015  00             C           DB     00          ;
0016  00             C           DB     00          ;
0017  00             C           DB     00          ;
                     C                              ;
0018  00             C   ERRBUF  DB     00          ; STATUS BYTE 0
0019  00             C           DB     00          ; STATUS BYTE 1
001A  00             C           DB     00          ; STATUS BYTE 2
001B  00             C           DB     00          ; CYLINDER/TRACK
001C  00             C           DB     00          ; HEAD 0 or HEAD 1
001D  00             C           DB     00          ; SECTOR
001E  00             C           DB     00          ; SECTOR SIZE
                     C                              ;
                     C   ;
                     C   ;*********************
                     C   ;***  DMA COMMANDS  ***
                     C   ;*********************
                     C   ;
                     C   ;
= 0047               C   DMAWRT  EQU    47H         ; WRITE DMA COMMAND
= 004B               C   DMAREAD EQU    4BH         ; READ DMA COMMAND
                     C   ;
                     C   ;
                     C   ;**********************
                     C   ;***  DMA VARIABLES  ***
                     C   ;**********************
                     C   ;
                     C   ;
001F  0000           C   DMAADDR DW     0000        ; DMA ADDR OFFSET
0021  0000           C           DW     0000        ;          SEGMENT
                     C                              ;
0023  0000           C   DMALENG DW     0000        ; DMA LENGTH
0025  00             C   DMAFUNC DB     00          ; DMA FUNCTION
```

```
                            C   ;
                            C   ;########################
                            C   ;###  DISK VARIABLES  ###
                            C   ;########################
                            C   ;
                            C   ;
0026  00                    C   TPI_DR  DB      00              ; 0 = 48 tpi, 1 = 96 tpi drive
                            C                                   ;
0027  00                    C   TPI_DI  DB      00              ; 0 = 48 tpi, 1 = 96 tpi disk
                            C                                   ;
0028  08                    C   SECTRK  DB      08              ; SECTORS PER TRACK
0029  40                    C   DENSITY DB      40H             ; DOUBLE DENSITY BIT (MFM)
002A  02                    C   BYTSEC  DB      02              ; BYTES PER SECTOR (N): 00 - 128 bytes
                            C                                   ;                       01 - 256 bytes
                            C                                   ;                       02 - 512 bytes
                            C                                   ;                          .
                            C                                   ;                          .
                            C                                   ;                          .
002B  1B                    C   GPL     DB      1BH             ; GAP LENGTH
                            C                                   ;
002C  F6                    C   PATTERN DB      0F6H            ; FORMAT PATTERN
                            C                                   ;
002D  05                    C   RETRIES DB      05              ; Number of retries
                            C   ;
                            C   ;
002E  0000                  C   SSB     DW      0000            ; Special Sector Buffer for BANK conflict
                            C                                   ; (not expanded for some CP/M O.S.)
0030  03FF [                            DW      1023 DUP (0)    ; Max. possible sector size
               0000
                   ]
```

```
                              C          INCLUDE FLXPIMC.ASM
                              C
                              C   ;**********************************************************************
                              C   ;**********************************************************************
                              C   ;**********************************************************************
                              C   ;
                              C   ;
                              C   ;    ROUTINE NAME:        DREAD
                              C   ;                         DWRITE
                              C   ;
                              C   ;
                              C   ;    FUNCTION:            DREAD - low level READ DATA
                              C   ;                         DWRITE - low level WRITE DATA
                              C   ;
                              C   ;    ENTRY VIA:           CALL
                              C   ;
                              C   ;
                              C   ;    ENTRY CONDITIONS:    Following variables are set:
                              C   ;                         TPI_DR, TPI_DI, BYTSEC, CYLMODE, DRV, HEAD, TRACK,
                              C   ;                         SECTOR, SECCNT (Number of sectors), SECTRK,
                              C   ;                         and DMAADDR (SEGMENT and OFFSET)
                              C   ;
                              C   ;    EXIT VIA:            RETURN
                              C   ;
                              C   ;
                              C   ;    EXIT CONDITIONS:     STATUS (returned in ERRBUF)
                              C   ;
                              C   ;**********************************************************************
                              C   ;**********************************************************************
                              C   ;**********************************************************************
                              C   ;
002E                          C   DREAD:                           ;
002E  B1 06                   C          MOV    CL,READDAT        ; CL <-- READ DATA COMMAND
0030  C6 06 0025 R 47         C          MOV    DMAFUNC,DMAWRT    ; DMAFUNC <-- WRITE DMA COMMAND
0035  EB 23                   C          JMP    SHORT I00         ;
0037                          C   DWRITE:                          ;
0037  B1 05                   C          MOV    CL,WRITDAT        ; CL <-- WRITE DATA COMMAND
0039  C6 06 0025 R 4B         C          MOV    DMAFUNC,DMAREAD ; DMAFUNC <-- READ DMA COMMAND
                              C                                   ;
003E  F7 06 0000 E 0002       C          TEST   CONFIG_FLAGS,WT48ON96 ; If writing of 48TPI diskettes enabled
                              C                                       ; for 96TPI drives skip protection check
0044  75 14                   C          JNZ    I00
0046  A0 0027 R               C          MOV    AL,TPI_DI         ; If 96 tpi drive
0049  2A 06 0026 R            C          SUB    AL,TPI_DR         ; and
004D  79 0B                   C          JNS    I00               ; 48 tpi disk
                              C                                   ; then
004F  C6 06 0018 R 40         C          MOV    BYTE PTR ERRBUF,40H ; 'WRITE PROTECT'
0054  C6 06 0019 R 02         C          MOV    BYTE PTR ERRBUF+1,02 ;
0059  C3                      C          RET                      ;
005A                          C   I00:                             ;
005A  A0 0026 R               C          MOV    AL,TPI_DR         ; If 48 tpi drive
005D  2A 06 0027 R            C          SUB    AL,TPI_DI         ; and
0061  79 0B                   C          JNS    I01               ; 96 tpi disk
                              C                                   ; then
0063  C6 06 0018 R 40         C          MOV    BYTE PTR ERRBUF,40H ; 'UNKNOWN MEDIA'
0068  C6 06 001A R 01         C          MOV    BYTE PTR ERRBUF+2,01 ;
006D  C3                      C          RET                      ;
```

```
086E                      C  I01:                          ;
086E  83 3E 000C R 00     C        CMP   SECCNT,0          ; Check if an I/O is necessary
0873  75 01               C        JNZ   I02               ; Jump if necessary
0875  C3                  C        RET                     ; Return if not necessary
0876                      C  I02:                          ;
                          C                                ; Check TRACK conflict
0876  B7 00               C        MOV   BH,00             ; --------------------
0878  8A 1E 0028 R        C        MOV   BL,SECTRK         ; BX <-- SECTORS PER TRACK
087C  FE C3               C        INC   BL                ;
087E  2A 1E 000B R        C        SUB   BL,SECTOR         ; BX - remainding sectors in track
                          C                                ;
0882  A0 0007 R           C        MOV   AL,CYLMODE        ; If CYLINDER MODE
0885  0A 06 0009 R        C        OR    AL,HEAD           ; and HEAD 0
0889  75 04               C        JNZ   I03               ;
088B  02 1E 0028 R        C        ADD   BL,SECTRK         ; then add sectors of corresponding track
088F                      C  I03:                          ;
088F  3B 1E 000C R        C        CMP   BX,SECCNT         ; Compare remaining sectors with SECCNT
0893  72 04               C        JB    I04               ; Jump if more than one I/O
0895  8B 1E 000C R        C        MOV   BX,SECCNT         ;
0899                      C  I04:                          ; BX - number of sectors fitting in TRACK
                          C                                ;
                          C                                ; Check BANK conflict
                          C                                ; -------------------
0899  A1 0021 R           C        MOV   AX,DMAADDR+2      ; AX <-- DMA SEGMENT
089C  D1 E0               C        SHL   AX,1              ;
089E  D1 E0               C        SHL   AX,1              ;
08A0  D1 E0               C        SHL   AX,1              ;
08A2  D1 E0               C        SHL   AX,1              ;
08A4  03 06 001F R        C        ADD   AX,DMAADDR        ; AX <-- absolute addr within BANK
08A8  F7 D8               C        NEG   AX                ; AX <-- remaining bytes within BANK
                          C                                ;
08AA  51                  C        PUSH  CX                ;
08AB  8A 0E 002A R        C        MOV   CL,BYTSEC         ;
08AF  BA 0080             C        MOV   DX,80H            ;
08B2  D3 E2               C        SHL   DX,CL             ; DX <-- sector size
08B4  59                  C        POP   CX                ;
                          C                                ;
08B5  8B F2               C        MOV   SI,DX             ; SI <-- sector size
08B7  BA 0000             C        MOV   DX,0000           ; DX <-- 0000
08BA  F7 F6               C        DIV   SI                ; AX <-- number of sectors fitting in BANK
                          C                                ;
08BC  3B C3               C        CMP   AX,BX             ; Check if we must do Special Sector Handling
08BE  72 03               C        JB    I06               ; Jump if we must
                          C                                ;
08C0  E9 0947 R           C        JMP   I015              ; Jump around if not
08C3                      C  I06:                          ;
08C3  93                  C        XCHG  BX,AX             ; BX <-- number of sectors fitting in BANK
08C4  83 FB 00            C        CMP   BX,00             ; Check if we must do now Special Sector Handling
08C7  74 03               C        JZ    I07               ; Jump if we must     ---
                          C                                ;
08C9  EB 7C 90            C        JMP   I015              ; Jump around if not
```

```
                            C
08CC                        C  IO7:                          ;** Special Sector Handling
                            C                                ;** -----------------------
08CC  83 2E 000C R 01       C          SUB    SECCNT,01      ;** SECCNT <-- remainding sectors for next I/O
08D1  51                    C          PUSH   CX             ;**
08D2  8A 0E 002A R          C          MOV    CL,BYTSEC      ;**
08D6  B8 0080               C          MOV    AX,80H         ;**
08D9  D3 E0                 C          SHL    AX,CL          ;** AX <-- sector size
08DB  59                    C          POP    CX             ;**
                            C                                ;**
08DC  A3 0023 R             C          MOV    DMALENG,AX     ;** DMALENG <-- sector size
                            C                                ;**
08DF  80 E1 0F              C          AND    CL,0FH         ;** Clear upper bits
08E2  80 F9 05              C          CMP    CL,WRITDAT     ;** Check if WRITE DATA COMMAND
08E5  75 1B                 C          JNZ    IO9            ;** Jump around if not
                            C                                ;*
08E7  51                    C          PUSH   CX             ;* Save CX
08E8  8B 36 001F R          C          MOV    SI,DMAADDR     ;* SI <-- source offset
08EC  BF 002E R             C          MOV    DI,OFFSET SSB  ;* DI <-- destination offset
08EF  8B 0E 0023 R          C          MOV    CX,DMALENG     ;* CX <-- sector size
08F3  D1 E9                 C          SHR    CX,1           ;* We move WORDS
08F5  FC                    C          CLD                   ;* incrementing
08F6  1E                    C          PUSH   DS             ;* Save DS
08F7  A1 0021 R             C          MOV    AX,DMAADDR+2   ;*
08FA  8E D8                 C          MOV    DS,AX          ;* DS <-- SEGMENT of TRANSFER ADDR
08FC  07                    C          POP    ES             ;*
08FD  06                    C          PUSH   ES             ;* ES <-- our SEGMENT of Special Sector Buffer
                            C                                ;*
                            C                                ;* W R I T E  D A T A  C O M M A N D:
08FE  F3/ A5                C  REP     MOVSW                 ;* Move BANK into Special Sector Buffer
                            C                                ;* ------------------------------------
0900  1F                    C          POP    DS             ;* Restore DS
0901  59                    C          POP    CX             ;* Restore CX
0902                        C  IO9:                          ;**
0902  A1 001F R             C          MOV    AX,DMAADDR     ;**
0905  50                    C          PUSH   AX             ;** Save DMA OFFSET
0906  A1 0021 R             C          MOV    AX,DMAADDR+2   ;**
0909  50                    C          PUSH   AX             ;** Save DMA SEGMENT
                            C                                ;**
090A  B8 002E R             C          MOV    AX,OFFSET SSB  ;**
090D  A3 001F R             C          MOV    DMAADDR,AX     ;** new OFFSET <-- Special Sector Buffer
0910  8C D8                 C          MOV    AX,DS          ;**
0912  A3 0021 R             C          MOV    DMAADDR+2,AX   ;** new SEGMENT <-- our SEGMENT
                            C                                ;**
0915  E8 0967 R             C          CALL   IO             ;** Do I/O
                            C                                ;** ------
0918  72 03                 C          JC     IO10           ;** Jump if normal termination
091A  58                    C          POP    AX             ;** else
091B  58                    C          POP    AX             ;** flush STACK
091C  C3                    C          RET                   ;** and return with bad status in ERRBUF
```

```
091D                  C  I010:                           ;**
091D  58              C          POP    AX               ;**
091E  A3 0021 R       C          MOV    DMAADDR+2,AX     ;** Restore DMA SEGMENT
0921  8E C0           C          MOV    ES,AX            ;**
0923  58              C          POP    AX               ;**
0924  A3 001F R       C          MOV    DMAADDR,AX       ;** Restore DMA OFFSET
                      C                                  ;**
0927  80 E1 0F        C          AND    CL,0FH           ;** Clear upper bits
092A  80 F9 06        C          CMP    CL,READDAT       ;** Check if READ DATA COMMAND
092D  75 12           C          JNZ    I011             ;** Jump around if not
                      C                                  ;*
092F  51              C          PUSH   CX               ;* Save CX
0930  BE 002E R       C          MOV    SI,OFFSET SSB    ;* SI <-- source offset
0933  8B 3E 001F R    C          MOV    DI,DMAADDR       ;* DI <-- destination offset
0937  8B 0E 0023 R    C          MOV    CX,DMALENG       ;* CX <-- sector size
093B  D1 E9           C          SHR    CX,1             ;* We move WORDS
093D  FC              C          CLD                     ;* incrementing
                      C                                  ;*R E A D  D A T A  C O M M A N D:
093E  F3/ A5          C  REP     MOVSW                   ;* Move Special Sector Buffer into BANK
                      C                                  ;* ---------------------------------
0940  59              C          POP    CX               ;* Restore CX
                      C                                  ;*
0941                  C  I011:                           ;**
0941  BB 0001         C          MOV    BX,0001          ;** BX - number of sectors of previous I/O
0944  EB 60 90        C          JMP    I030             ;** Jump to update variables for next I/O
                      C
0947                  C  I015:                           ; BX - number of sectors for I/O
0947  53              C          PUSH   BX               ; -----------------------------
0948  29 1E 000C R    C          SUB    SECCNT,BX        ; SECCNT <-- remainding sectors for next I/O
                      C                                  ;
094C  51              C          PUSH   CX               ;
094D  8A 0E 002A R    C          MOV    CL,BYTSEC        ;
0951  B8 0080         C          MOV    AX,80H           ;
0954  D3 E0           C          SHL    AX,CL            ; AX <-- sector size
0956  59              C          POP    CX               ;
                      C                                  ;
0957  F7 E3           C          MUL    BX               ; * sectors for I/O gives DMA LENGTH
0959  A3 0023 R       C          MOV    DMALENG,AX       ; DMALENG <-- DMA LENGTH
                      C                                  ;
095C  E8 0967 R       C          CALL   IO               ; Do I/O
                      C                                  ; ------
095F  72 02           C          JC     I017             ; Jump if normal termination
0961  58              C          POP    AX               ; else flush STACK
0962  C3              C          RET                     ; and return with bad status in ERRBUF
0963                  C  I017:                           ;
0963  5B              C          POP    BX               ; BX - number of sectors of previous I/O
0964  EB 40 90        C          JMP    I030             ; Jump to update variables for next I/O
```

```
                              C
0967                          C   IO:                             ; Disk I/O
                              C                                   ; --------
0967  A0 002D R               C          MOV    AL,RETRIES        ; AL <--  retry counter
096A                          C   IO20:                           ;
096A  50                      C          PUSH   AX                ; Save retry counter
096B  E8 0AE2 R               C          CALL   SETUP9            ; Set up COMMAND STRING and DMA
096E  E8 0B70 R               C          CALL   XWAIT             ; Send COMMAND STRING to FDC
0971  E8 0B8E R               C          CALL   GETBYT            ; Get STATUS BYTES
0974  58                      C          POP    AX                ; Restore retry counter
                              C                                   ;
0975  F6 06 0018 R C0         C          TEST   ERRBUF,0C0H       ; Test for normal termination
097A  75 02                   C          JNZ    IO21              ; Jump on error
097C  F9                      C          STC                      ; Set status flag
097D  C3                      C          RET                      ; Return with good status
                              C                                   ;
097E                          C   IO21:                           ;
097E  F6 06 0018 R 08         C          TEST   ERRBUF,08H        ; Test for 'NOT READY'
0983  74 02                   C          JZ     IO22              ;
0985  F8                      C          CLC                      ; Set status flag
0986  C3                      C          RET                      ; Return immediately if disk 'NOT READY'
0987                          C   IO22:                           ;
0987  F6 06 0019 R 02         C          TEST   ERRBUF+1,02H      ; Test for 'WRITE PROTECTED'
098C  74 02                   C          JZ     IO23              ;
098E  F8                      C          CLC                      ; Set status flag
098F  C3                      C          RET                      ; Return immediately if 'WRITE PROTECTED'
0990                          C   IO23:                           ;
0990  F6 06 0018 R 80         C          TEST   ERRBUF,80H        ; Test for 'INVALID COMMAND'
0995  74 02                   C          JZ     IO24              ;
0997  F8                      C          CLC                      ; Set status flag
0998  C3                      C          RET                      ; Return immediately if 'INVALID COMMAND'
0999                          C   IO24:                           ;
0999  FE C8                   C          DEC    AL                ; Decrement retry counter
099B  74 07                   C          JZ     IO25              ; Jump to exit with bad status
                              C                                   ;
099D  50                      C          PUSH   AX                ; Save retry counter
099E  E8 0A2C R               C          CALL   DREST             ; Do a low level RESTORE
09A1  58                      C          POP    AX                ; Restore retry counter
09A2  EB C6                   C          JMP    IO20              ; Do retries
                              C                                   ;
09A4                          C   IO25:                           ;
09A4  F8                      C          CLC                      ; Set status flag
09A5  C3                      C          RET                      ; Return with bad status
```

```
                              C
09A6                          C  IO30:                          ; Update variables for next I/O
                              C                                 ; -----------------------------
                              C                                 ; BX - number of sectors of previous I/O
                              C                                 ;
09A6  83 3E 000C R 00         C          CMP     SECCNT,0       ; Check if another I/O is necessary
09AB  75 01                   C          JNZ     IO31           ; Jump if necessary
09AD  C3                      C          RET                    ; Return if not necessary
09AE                          C  IO31:                          ;
09AE  8B 16 0023 R            C          MOV     DX,DMALENG     ; DX <-- previous DMA LENGTH
09B2  D1 EA                   C          SHR     DX,1           ;
09B4  D1 EA                   C          SHR     DX,1           ;
09B6  D1 EA                   C          SHR     DX,1           ;
09B8  D1 EA                   C          SHR     DX,1           ; DX - previous DMA LENGTH in paragraphs
09BA  01 16 0021 R            C          ADD     WORD PTR DMAADDR+2,DX ; Update DMAADDR (SEGMENT)
09BE  00 1E 000B R            C          ADD     SECTOR,BL      ; Update SECTOR variable
09C2  A0 0028 R               C          MOV     AL,SECTRK      ; AL <-- sectors per track
09C5  80 3E 0007 R 00         C          CMP     CYLMODE,00     ; Check if CYLINDER MODE
09CA  74 29                   C          JZ      IO34           ; Jump if CYLINDER MODE
                              C                                 ;
                              C                                 ; Not CYLINDER MODE
                              C                                 ; -----------------
09CC  3A 06 000B R            C          CMP     AL,SECTOR      ; Check for legal SECTOR variable
09D0  72 03                   C          JB      IO32           ; Jump if not legal
09D2  E9 006E R               C          JMP     IO1            ; Do next I/O
09D5                          C  IO32:                          ;
09D5  C6 06 000B R 01         C          MOV     SECTOR,1       ; Set SECTOR to begin of track
09DA  80 3E 000A R 27         C          CMP     TRACK,39       ; Check if side 1 is full
09DF  74 07                   C          JZ      IO33           ; Jump if full
                              C                                 ;
09E1  FE 06 000A R            C          INC     TRACK          ; Increment TRACK
09E5  E9 006E R               C          JMP     IO1            ; Do next I/O
09E8                          C  IO33:                          ;
09E8  C6 06 0009 R 01         C          MOV     HEAD,1         ; If side 1 is full
09ED  C6 06 000A R 00         C          MOV     TRACK,0        ; then initialize for side 2
09F2  E9 006E R               C          JMP     IO1            ; Do next I/O
09F5                          C  IO34:                          ;
                              C                                 ; CYLINDER MODE
                              C                                 ; -------------
09F5  3A 06 000B R            C          CMP     AL,SECTOR      ; Check for legal SECTOR variable
09F9  72 03                   C          JB      IO35           ; Jump if not legal
                              C                                 ;
09FB  E9 006E R               C          JMP     IO1            ; Do next I/O
09FE                          C  IO35:                          ;
09FE  80 3E 0009 R 01         C          CMP     HEAD,1         ; Check if cylinder is full
0A03  74 16                   C          JZ      IO36           ; Jump if full
                              C                                 ;
0A05  D0 E0                   C          SHL     AL,1           ; AL <-- sectors per cylinder
0A07  3A 06 000B R            C          CMP     AL,SECTOR      ; Check if cylinder is full
0A0B  72 0E                   C          JB      IO36           ; Jump if full
                              C                                 ;
0A0D  D0 E8                   C          SHR     AL,1           ; AL <-- sectors per track
0A0F  28 06 000B R            C          SUB     SECTOR,AL      ; Set SECTOR variable within
0A13  C6 06 0009 R 01         C          MOV     HEAD,1         ; corresponding track with HEAD 1
0A18  E9 006E R               C          JMP     IO1            ; Do next I/O
0A1B                          C  IO36:                          ;
0A1B  FE 06 000A R            C          INC     TRACK          ; Increment TRACK
0A1F  C6 06 0009 R 00         C          MOV     HEAD,0         ; Set HEAD 0
0A24  C6 06 000B R 01         C          MOV     SECTOR,1       ; Set SECTOR to begin of cylinder
0A29  E9 006E R               C          JMP     IO1            ; Do next I/O
```

```
                         C  ;**************************************************************************
                         C  ;**************************************************************************
                         C  ;**************************************************************************
                         C  ;
                         C  ;
                         C  ;   ROUTINE NAME:       DREST
                         C  ;
                         C  ;
                         C  ;   FUNCTION:           Low level RESTORE
                         C  ;
                         C  ;
                         C  ;   ENTRY VIA:          CALL
                         C  ;
                         C  ;
                         C  ;   ENTRY CONDITIONS:   DRV variable is set
                         C  ;
                         C  ;
                         C  ;   EXIT VIA:           RETURN
                         C  ;
                         C  ;
                         C  ;   EXIT CONDITIONS:    CL - preserved
                         C  ;
                         C  ;**************************************************************************
                         C  ;**************************************************************************
                         C  ;**************************************************************************
                         C  ;
                         C  ;
0A2C                     C  DREST:                           ;
0A2C  B4 02              C          MOV    AH,02             ; Special retry for CP/M
                         C                                   ;
0A2E                     C  DREST1:                          ; Set up COMMAND STRING
                         C                                   ; ---------------------
0A2E  C6 06 000E R 02    C          MOV    COMSTR,2          ; COMMAND STRING <-- LENGTH 2
0A33  C6 06 000F R 07    C          MOV    COMSTR+1,RESTORE;                 <-- RESTORE COMMAND
0A38  A0 0008 R          C          MOV    AL,DRV            ;
0A3B  A2 0010 R          C          MOV    COMSTR+2,AL       ;                 <-- DRIVE NUMBER
                         C                                   ;
0A3E  50                 C          PUSH   AX                ; Save retry counter
0A3F  E8 0B70 R          C          CALL   XWAIT             ; Send COMMAND STRING to FDC
0A42                     C  DREST2:                          ;
0A42  E4 13              C          IN     AL,SYSSTA         ; Wait on interrupt
0A44  24 08              C          AND    AL,08             ; Test DISK INTERRUPT BIT
0A46  74 FA              C          JZ     DREST2            ; Jump if no interrupt
                         C                                   ;
0A48  E8 0AB2 R          C          CALL   DSIS              ; Reset interrupt via low level SENSE
                         C                                   ; INTERRUPT STATUS
                         C                                   ;
0A4B  58                 C          POP    AX                ; Restore retry counter
0A4C  F6 06 0018 R C0    C          TEST   ERRBUF,0C0H       ; Test for normal termination
0A51  74 04              C          JZ     DREST3            ; Jump if normal termination
                         C                                   ;
0A53  FE CC              C          DEC    AH                ; Decrement retry counter
0A55  75 D7              C          JNZ    DREST1            ; Do special retry !
0A57                     C  DREST3:                          ; Reason: MOTOR OFF & RESTORE in CP/M
0A57  C3                 C          RET                      ;
```

```
                              C  ;########################################################################
                              C  ;########################################################################
                              C  ;########################################################################
                              C  ;
                              C  ;
                              C  ;    ROUTINE NAME:        DSEEK
                              C  ;
                              C  ;
                              C  ;    FUNCTION:            Low level SEEK A TRACK
                              C  ;
                              C  ;
                              C  ;    ENTRY VIA:           CALL
                              C  ;
                              C  ;
                              C  ;    ENTRY CONDITIONS:    Following variables are set:
                              C  ;                         TPI_DR, TPI_DI, DRV, HEAD, and TRACK
                              C  ;
                              C  ;    EXIT VIA:            RETURN
                              C  ;
                              C  ;
                              C  ;    EXIT CONDITIONS:     CL - preserved
                              C  ;                         STATUS (returned in ERRBUF)
                              C  ;
                              C  ;########################################################################
                              C  ;########################################################################
                              C  ;########################################################################
                              C  ;
0A58                          C  DSEEK:                         ; Set up COMMAND STRING
                              C                                 ; ---------------------
0A58  C6 06 000E R 03         C          MOV    COMSTR,3        ; COMMAND STRING <-- LENGTH 3
0A5D  C6 06 000F R 0F         C          MOV    COMSTR+1,SEEKTRK;                <-- SEEK COMMAND
0A62  A0 0009 R               C          MOV    AL,HEAD         ;
0A65  D0 E0                   C          SHL    AL,1            ;
0A67  D0 E0                   C          SHL    AL,1            ;
0A69  0A 06 0008 R            C          OR     AL,DRV          ;
0A6D  A2 0010 R               C          MOV    COMSTR+2,AL     ;                <-- DRIVE & HEAD
0A70  A0 000A R               C          MOV    AL,TRACK        ;
                              C                                 ;
0A73  8A 26 0027 R            C          MOV    AH,TPI_DI       ;
0A77  3A 26 0026 R            C          CMP    AH,TPI_DR       ; Check if tpi from disk and drive are equal
0A7B  74 02                   C          JZ     DSEEK1          ; Jump if equal
                              C                                 ; else
0A7D  D0 E0                   C          SHL    AL,1            ; 48 tpi disk in 96 tpi drive
0A7F                          C  DSEEK1:                        ;
0A7F  A2 0011 R               C          MOV    COMSTR+3,AL     ;                <-- TRACK
                              C                                 ;
0A82  E8 0B70 R               C          CALL   XWAIT           ; Send COMMAND STRING to FDC
0A85                          C  DSEEK2:                        ;
0A85  E4 13                   C          IN     AL,SYSSTA       ; Wait on interrupt
0A87  24 80                   C          AND    AL,80           ; Test DISK INTERRUPT BIT
0A89  74 FA                   C          JZ     DSEEK2          ; jump if no interrupt
                              C                                 ;
0A8B  E8 0AB2 R               C          CALL   DSIS            ; Reset interrupt via low level SENSE
                              C                                 ; INTERRUPT STATUS
0A8E  C3                      C          RET                    ;
```

```
                            C
                            C  ;******************************************************************************
                            C  ;******************************************************************************
                            C  ;******************************************************************************
                            C  ;
                            C  ;
                            C  ;    ROUTINE NAME:       DREADID
                            C  ;
                            C  ;    FUNCTION:           Low level READ ID
                            C  ;                        (Used to get SECTOR SIZE)
                            C  ;
                            C  ;    ENTRY VIA:          CALL
                            C  ;
                            C  ;
                            C  ;    ENTRY CONDITIONS:   Following variables are set:
                            C  ;                        DRV and HEAD
                            C  ;
                            C  ;    EXIT VIA:           RETURN
                            C  ;
                            C  ;
                            C  ;    EXIT CONDITIONS:    STATUS and BYTES PER SECTOR (returned in ERRBUF)
                            C  ;
                            C  ;
                            C  ;******************************************************************************
                            C  ;******************************************************************************
                            C  ;******************************************************************************
                            C  ;
                            C  ;
0A8F                        C  DREADID:                        ; Set up COMMAND STRING
                            C                                  ; ---------------------
0A8F  C6 06 000E R 02       C          MOV     COMSTR,2        ; COMMAND STRING <-- LENGTH 2
0A94  B0 0A                 C          MOV     AL,IDREAD       ;
0A96  0A 06 0029 R          C          OR      AL,DENSITY      ;
0A9A  A2 000F R             C          MOV     COMSTR+1,AL     ;                <-- READ ID COMMAND & DENSITY
0A9D  A0 0009 R             C          MOV     AL,HEAD         ;
0AA0  D0 E0                 C          SHL     AL,1            ;
0AA2  D0 E0                 C          SHL     AL,1            ;
0AA4  0A 06 0008 R          C          OR      AL,DRV          ;
0AA8  A2 0010 R             C          MOV     COMSTR+2,AL     ;                <-- DRIVE & HEAD
                            C                                  ;
0AAB  E8 0078 R             C          CALL    XWAIT           ; Send COMMAND STRING to FCB
0AAE  E8 00BE R             C          CALL    GETBYT          ; Get STATUS BYTES (sector size)
0AB1  C3                    C          RET                     ;
```

```
                            C ;*****************************************************************************
                            C ;*****************************************************************************
                            C ;*****************************************************************************
                            C ;
                            C ;
                            C ;
                            C ;   ROUTINE NAME:      DSIS
                            C ;
                            C ;
                            C ;   FUNCTION:          Low level SENSE INTERRUPT STATUS
                            C ;                      (used to reset interrupt)
                            C ;
                            C ;   ENTRY VIA:         CALL
                            C ;
                            C ;
                            C ;   ENTRY CONDITIONS:  NONE
                            C ;
                            C ;
                            C ;   EXIT VIA:          RETURN
                            C ;
                            C ;   EXIT CONDITIONS:   STATUS (returned in ERRBUF)
                            C ;
                            C ;*****************************************************************************
                            C ;*****************************************************************************
                            C ;*****************************************************************************
                            C ;
                            C ;
0AB2                        C DSIS:                           ; Set up COMMAND STRING
                            C                                 ; ---------------------
0AB2  C6 06 000E R 01       C        MOV    COMSTR,1          ; COMMAND STRING <-- LENGTH 1
0AB7  C6 06 000F R 08       C        MOV    COMSTR+1,FDCSIS ;               <-- FDCSIS COMMAND
                            C                                 ;
0ABC  E8 0B70 R             C        CALL   XWAIT             ; Send COMMAND STRING to FDC
0ABF  E8 0B8E R             C        CALL   GETBYT            ; Get STATUS BYTES
0AC2  C3                    C        RET                      ;
```

```
                             C  ;********************************************************************************
                             C  ;********************************************************************************
                             C  ;********************************************************************************
                             C  ;
                             C  ;
                             C  ;    ROUTINE NAME:       DFORMAT
                             C  ;
                             C  ;
                             C  ;    FUNCTION:           Low level FORMAT A TRACK
                             C  ;
                             C  ;
                             C  ;    ENTRY VIA:          CALL
                             C  ;
                             C  ;
                             C  ;    ENTRY CONDITIONS:   Following variables are set:
                             C  ;                        TPI_DR, TPI_DI, BYTSEC, DRV, HEAD, TRACK,
                             C  ;                        SECTRK, PATTERN,
                             C  ;                        and DMAADDR (SEGMENT and OFFSET)
                             C  ;
                             C  ;    EXIT VIA:           RETURN
                             C  ;
                             C  ;
                             C  ;    EXIT CONDITIONS:    STATUS (returned in ERRBUF)
                             C  ;
                             C  ;
                             C  ;********************************************************************************
                             C  ;********************************************************************************
                             C  ;********************************************************************************
                             C  ;
                             C  ;
0AC3                         C  DFORMAT:                         ;
0AC3  B1 0D                  C          MOV     CL,WRITFMT       ; CL <-- FORMAT COMMAND
0AC5  C6 06 0025 R 4B        C          MOV     DMAFUNC,DMAREAD  ; DMAFUNC <-- READ DMA COMMAND
0ACA  B7 00                  C          MOV     BH,00            ;
0ACC  8A 1E 0028 R           C          MOV     BL,SECTRK        ;
0AD0  D1 E3                  C          SHL     BX,1             ;
0AD2  D1 E3                  C          SHL     BX,1             ;
0AD4  89 1E 0023 R           C          MOV     DMALENG,BX       ; DMALENG <-- DMA LENGTH (SECTRK*4)
                             C                                   ;
0AD8  E8 0B37 R              C          CALL    SETUP6           ; Set up COMMAND STRING and DMA
0ADB  E8 0B70 R              C          CALL    XWAIT            ; Send COMMAND STRING to FDC
0ADE  E8 0B8E R              C          CALL    GETBYT           ; Get STATUS BYTES
0AE1  C3                     C          RET                      ;
```

```
                            C ;######################################################################
                            C ;######################################################################
                            C ;######################################################################
                            C ;
                            C ;
                            C ;     ROUTINE NAME:       SETUP9
                            C ;
                            C ;
                            C ;     FUNCTION:           Set up (9 byte) COMMAND STRING and DMA
                            C ;
                            C ;     ENTRY VIA:          CALL
                            C ;
                            C ;
                            C ;     ENTRY CONDITIONS:   CL - COMMAND
                            C ;                         Following variables are set:
                            C ;                         TPI_DR, TPI_DI, BYTSEC, CYLMODE, DRV, HEAD, TRACK,
                            C ;                         SECTOR, SECTRK
                            C ;                         DMAADDR (SEGMENT and OFFSET)
                            C ;                         DMALENG and DMAFUNC
                            C ;
                            C ;     EXIT VIA:           RETURN
                            C ;
                            C ;
                            C ;     EXIT CONDITIONS:    NONE
                            C ;
                            C ;######################################################################
                            C ;######################################################################
                            C ;######################################################################
                            C ;
 0AE2                       C SETUP9:                           ;
0 AE2  E8 0A58 R            C         CALL    DSEEK             ; First do low level SEEK A TRACK
                            C                                   ;
0AE 5  C6 06 000E R 09      C         MOV     COMSTR,9          ; COMMAND STRING <-- LENGTH 9
0AEA   0A 0E 0029 R         C         OR      CL,DENSITY        ;
0AEE   80 3E 0007 R 00      C         CMP     CYLMODE,00        ;
0AF3   75 03                C         JNZ     SET1              ;
                            C                                   ;
0AF5  80  C9 80             C         OR      CL,80H            ;
0AF8                        C SET1:                             ;
0AF8  88 0 E 000F R         C         MOV     COMSTR+1,CL       ;          <-- FUNCTION & DENSITY & MT
0AFC  A0 00 09 R            C         MOV     AL,HEAD           ;
0AFF  D0 E0                 C         SHL     AL,1              ;
0B01  D0 E0                 C         SHL     AL,1              ;
0B03  0A 06 00 08 R         C         OR      AL,DRV            ;
0B07  A2 0010 R             C         MOV     COMSTR+2,AL       ;          <-- DRIVE & HEAD
0B0A  A0 000A R             C         MOV     AL,TRACK          ;
0B0D  A2 0011 R             C         MOV     COMSTR+3,AL       ;          <-- TRACK
0B10  A0 0009 R             C         MOV     AL,HEAD           ;
0B13  A2 0012 R             C         MOV     COMSTR+4,AL       ;          <-- HEAD
0B16  A0 000B R             C         MOV     AL,SECTOR         ;
0B19  A2 0013 R             C         MOV     COMSTR+5,AL       ;          <-- SECTOR
0B1C  A0 002A R             C         MOV     AL,BYTSEC         ;
0B1F  A2 0014 R             C         MOV     COMSTR+6,AL       ;          <-- BYTES PER SECTOR
0B22  A0 0028 R             C         MOV     AL,SECTRK         ;
0B25  A2 0015 R             C         MOV     COMSTR+7,AL       ;          <-- SECTORS PER TRACK
0B28  A0 002B R             C         MOV     AL,GPL            ;
0B2B  A2 0016 R             C         MOV     COMSTR+8,AL       ;          <-- GAP LENGTH
0B2E  C6 06 0017 R FF       C         MOV     COMSTR+9,0FFH     ;          <-- DTL
                            C                                   ;
0B33  E8 0BB7 R             C         CALL    DMA               ; Initialize DMA
0B36  C3                    C         RET                       ;
```

```
                               C  ;*************************************************************************
                               C  ;*************************************************************************
                               C  ;*************************************************************************
                               C  ;
                               C  ;
                               C  ;    ROUTINE NAME:       SETUP6
                               C  ;
                               C  ;    FUNCTION:           Set up (6 byte) COMMAND STRING and DMA
                               C  ;
                               C  ;
                               C  ;    ENTRY VIA:          CALL
                               C  ;
                               C  ;
                               C  ;    ENTRY CONDITIONS:   CL - (FORMAT) COMMAND
                               C  ;                        Following variables are set:
                               C  ;                        TPI_DR, TPI_DI, BYTSEC, DRV, HEAD, TRACK, PATTERN
                               C  ;                        DMAADDR (SEGMENT and OFFSET)
                               C  ;                        DMALENG and DMAFUNC
                               C  ;
                               C  ;    EXIT VIA:           RETURN
                               C  ;
                               C  ;
                               C  ;    EXIT CONDITIONS:    NONE
                               C  ;
                               C  ;
                               C  ;*************************************************************************
                               C  ;*************************************************************************
                               C  ;*************************************************************************
                               C  ;
                               C  ;
0B37                           C  SETUP6:                          ;
0B37  E8 0A58 R                C          CALL    DSEEK            ; First do low level SEEK A TRACK
                               C                                   ;
0B3A  C6 06 008E R 06          C          MOV     COMSTR,6         ; COMMAND STRING <-- LENGTH 6
0B3F  0A 0E 0029 R             C          OR      CL,DENSITY       ;
0B43  88 0E 008F R             C          MOV     COMSTR+1,CL      ;                <-- FUNCTION & DENSITY
0B47  A0 0009 R                C          MOV     AL,HEAD          ;
0B4A  D0 E0                    C          SHL     AL,1             ;
0B4C  D0 E0                    C          SHL     AL,1             ;
0B4E  0A 06 0008 R             C          OR      AL,DRV           ;
0B52  A2 0010 R                C          MOV     COMSTR+2,AL      ;                <-- DRIVE & HEAD
0B55  A0 002A R                C          MOV     AL,BYTSEC        ;
0B58  A2 0011 R                C          MOV     COMSTR+3,AL      ;                <-- BYTES PER SECTOR
0B5B  A0 0028 R                C          MOV     AL,SECTRK        ;
0B5E  A2 0012 R                C          MOV     COMSTR+4,AL      ;                <-- SECTORS PER TRACK
0B61  C6 06 0013 R 50          C          MOV     COMSTR+5,50H     ;                <-- GAP LENGTH
0B66  A0 002C R                C          MOV     AL,PATTERN       ;
0B69  A2 0014 R                C          MOV     COMSTR+6,AL      ;                <-- PATTERN
                               C                                   ;
0B6C  E8 0BB7 R                C          CALL    DMA              ; Initialize DMA
0B6F  C3                       C          RET                      ;
```

```
                      C  ;***************************************************************************
                      C  ;***************************************************************************
                      C  ;***************************************************************************
                      C  ;
                      C  ;
                      C  ;    ROUTINE NAME:       XWAIT
                      C  ;
                      C  ;
                      C  ;    FUNCTION:           Send COMMAND STRING to FDC
                      C  ;
                      C  ;
                      C  ;    ENTRY VIA:          CALL
                      C  ;
                      C  ;
                      C  ;    ENTRY CONDITIONS:   NONE
                      C  ;
                      C  ;
                      C  ;    EXIT VIA:           RETURN
                      C  ;
                      C  ;
                      C  ;    EXIT CONDITIONS:    CL - preserved
                      C  ;
                      C  ;***************************************************************************
                      C  ;***************************************************************************
                      C  ;***************************************************************************
                      C  ;
                      C  ;
0B70                  C  XWAIT:                               ;
0B70  E8 0BA5 R       C          CALL    MOTORCK             ; SWITCH MOTOR ON
                      C                                      ;
0B73  8A 2E 000E R    C          MOV     CH,COMSTR           ; CH <-- COMMAND STRING LENGTH
0B77  BB 000E R       C          MOV     BX,OFFSET COMSTR; BX <-- Addr of COMMAND STRING
0B7A                  C  XWAIT1:                              ;
0B7A  43              C          INC     BX                  ;
0B7B  E8 0B9E R       C          CALL    FDCRDY              ; Wait until FDC is ready
0B7E  8A 07           C          MOV     AL,BYTE PTR [BX]; AL <-- next COMMAND STRING byte
0B80  E6 51           C          OUT     DCOMD,AL            ; Send byte to FDC
0B82  FE CD           C          DEC     CH                  ; Decrement counter
0B84  75 F4           C          JNZ     XWAIT1              ; Loop until last byte
                      C                                      ;
0B86  E8 0B9E R       C          CALL    FDCRDY              ; Wait until FDC is ready
                      C                                      ;
0B89  B0 07           C          MOV     AL,07               ;
0B8B  E6 2A           C          OUT     DMAMB,AL            ; Disable DMA CHANNEL
0B8D  C3              C          RET                         ;
```

```
                            C
                            C  ;************************************************************************
                            C  ;************************************************************************
                            C  ;
                            C  ;
                            C  ;    ROUTINE NAME:       GETBYT
                            C  ;
                            C  ;
                            C  ;    FUNCTION:           Get STATUS BYTES into ERRBUF
                            C  ;
                            C  ;
                            C  ;    ENTRY VIA:          CALL
                            C  ;
                            C  ;
                            C  ;    ENTRY CONDITIONS:   NONE
                            C  ;
                            C  ;    EXIT VIA:           RETURN
                            C  ;
                            C  ;
                            C  ;    EXIT CONDITIONS:    NONE
                            C  ;
                            C  ;************************************************************************
                            C  ;************************************************************************
                            C  ;************************************************************************
                            C  ;
                            C  ;
0B8E                        C  GETBYT:                     ;
0B8E  BB 0018 R             C          MOV     BX,OFFSET ERRBUF; BX <-- Addr of ERROR BUFFER
0B91                        C  GETBYT1:                    ;
0B91  E4 51                 C          IN      AL,FDCRA        ; Read STATUS BYTE from FDC
0B93  88 07                 C          MOV     BYTE PTR [BX],AL; into ERROR BUFFER
0B95  43                    C          INC     BX              ;
0B96  E8 0B9E R             C          CALL    FDCRDY          ; Wait until FDC is ready
0B99  A8 40                 C          TEST    AL,40H          ; Check if FDC has another byte
0B9B  75 F4                 C          JNZ     GETBYT1         ; Jump to fetch next byte
0B9D  C3                    C          RET                     ;
```

```
                        C ;
                        C ;******************************************************************************
                        C ;******************************************************************************
                        C ;******************************************************************************
                        C ;
                        C ;
                        C ;    ROUTINE NAME:       FDCRDY
                        C ;
                        C ;
                        C ;    FUNCTION:           Wait until FDC is ready
                        C ;
                        C ;
                        C ;    ENTRY VIA:          CALL
                        C ;
                        C ;
                        C ;    ENTRY CONDITIONS:   NONE
                        C ;
                        C ;
                        C ;    EXIT VIA:           RETURN
                        C ;
                        C ;
                        C ;    EXIT CONDITIONS:    NONE
                        C ;
                        C ;******************************************************************************
                        C ;******************************************************************************
                        C ;******************************************************************************
                        C ;
                        C ;
0B9E                    C FDCRDY:                          ;
0B9E  E4 50             C         IN      AL,DSTAT         ; AL <-- DISK STATUS
0BA0  A8 80             C         TEST    AL,80H           ; Test MASTER REQUEST BIT
0BA2  74 FA             C         JZ      FDCRDY           ; Jump if no MASTER REQUEST (means: in execution)
                        C                                  ;
0BA4  C3                C         RET                      ; Return if FDC is ready
```

```
                          C  ;
                          C  ;************************************************************************
                          C  ;************************************************************************
                          C  ;************************************************************************
                          C  ;
                          C  ;
                          C  ;     ROUTINE NAME:       MOTORCK
                          C  ;
                          C  ;
                          C  ;     FUNCTION:           Check if motor is on
                          C  ;
                          C  ;
                          C  ;     ENTRY VIA:          CALL
                          C  ;
                          C  ;
                          C  ;     ENTRY CONDITIONS:   NONE
                          C  ;
                          C  ;
                          C  ;     EXIT VIA:           RETURN
                          C  ;
                          C  ;
                          C  ;     EXIT CONDITIONS:    Motor is on
                          C  ;
                          C  ;************************************************************************
                          C  ;************************************************************************
                          C  ;************************************************************************
                          C  ;
                          C  ;
0BA5                      C  MOTORCK:                           ;
0BA5  E4 13               C          IN      AL,SYSSTA          ; AL <-- SYSTEM STATUS
0BA7  24 01               C          AND     AL,01              ; Test DISK MOTOR ON BIT
0BA9  E6 14               C          OUT     MOTORON,AL         ; Switch motor on
0BAB  75 01               C          JNZ     MOTORCK1           ;
0BAD  C3                  C          RET                        ; Return if motor was on
0BAE                      C  MOTORCK1:                          ;
0BAE  BB FFFF             C          MOV     BX,0FFFFH          ; Wait some time if motor was off
0BB1                      C  MOTORCK2:                          ;
0BB1  D4 0A               C          AAM                        ; (83)
0BB3  4B                  C          DEC     BX                 ; ( 2)
0BB4  75 FB               C          JNZ     MOTORCK2           ; ( 8)  = 93 CLOCKS * FFFF = 1 sec
                          C                                     ;
0BB6  C3                  C          RET                        ;
```

```
                          C ;####################################################################
                          C ;####################################################################
                          C ;####################################################################
                          C ;
                          C ;
                          C ;   ROUTINE NAME:      DMA
                          C ;
                          C ;
                          C ;   FUNCTION:          DMA routines
                          C ;
                          C ;
                          C ;   ENTRY VIA:         CALL
                          C ;
                          C ;
                          C ;   ENTRY CONDITIONS:  Following variables are set:
                          C ;                      DMAADDR (SEGMENT and OFFSET)
                          C ;                      DMALENG and DMAFUNC
                          C ;
                          C ;   EXIT VIA:          RETURN
                          C ;
                          C ;
                          C ;   EXIT CONDITIONS:   NONE
                          C ;
                          C ;####################################################################
                          C ;####################################################################
                          C ;####################################################################
                          C ;
0BB7                      C DMA:                                  ;
0BB7  A0 0025 R           C         MOV     AL,DMAFUNC            ; DMAFUNC <-- DMA FUNCTION
0BBA  E6 2B               C         OUT     DMAMO,AL              ; OUT MODE
                          C                                       ;
0BBC  A1 0021 R           C         MOV     AX,DMAADDR+2          ; AX <-- DMA SEGMENT
0BBF  D1 E0               C         SHL     AX,1                  ;
0BC1  D1 E0               C         SHL     AX,1                  ;
0BC3  D1 E0               C         SHL     AX,1                  ;
0BC5  D1 E0               C         SHL     AX,1                  ;
0BC7  03 06 001F R        C         ADD     AX,DMAADDR            ; AX <-- absolute addr within BANK
0BCB  E6 26               C         OUT     COAD,AL               ; OUT DMA ADDR low
0BCD  BA C4               C         MOV     AL,AH                 ;
0BCF  E6 26               C         OUT     COAD,AL               ; OUT DMA ADDR high
                          C                                       ;
0BD1  A1 0023 R           C         MOV     AX,DMALENG            ; AX <-- DMA LENGTH
0BD4  48                  C         DEC     AX                    ;
0BD5  E6 27               C         OUT     COTC,AL               ; OUT DMA LENGTH low
0BD7  8A C4               C         MOV     AL,AH                 ;
0BD9  E6 27               C         OUT     COTC,AL               ; OUT DMA LENGTH high
                          C                                       ;
0BDB  B6 00               C         MOV     DH,00                 ;
0BDD  B2 E0               C         MOV     DL,BANK               ; DX - BANK 0 initialisation
0BDF  80 D2 00            C         ADC     DL,00                 ; DX - next BANK if SEGMENT + OFFSET > 64K
0BE2  A1 0021 R           C         MOV     AX,DMAADDR+2          ; AX <-- DMA SEGMENT
0BE5  D0 EC               C         SHR     AH,1                  ;
0BE7  D0 EC               C         SHR     AH,1                  ;
0BE9  D0 EC               C         SHR     AH,1                  ;
0BEB  D0 EC               C         SHR     AH,1                  ;
0BED  02 D4               C         ADD     DL,AH                 ; DX <-- BANK SELECT PORT
0BEF  EE                  C         OUT     DX,AL                 ; SELECT BANK
                          C                                       ; -----------
0BF0  B0 03               C         MOV     AL,03                 ;
0BF2  E6 2A               C         OUT     DMAMB,AL              ; Enable FDC CHANNEL
0BF4  C3                  C         RET                           ;
```

```
0BF5                      DSKTBL:
0BF5  0E16 R                      DW      DSK_INIT        ; 0  - INIT
0BF7  1A                          DB      26              ; Length of drive request structure
                                                          ;
0BF8  0E64 R                      DW      MEDIAC          ; 1  - MEDIA CHECK
0BFA  0F                          DB      15              ; Length of drive request structure
                                                          ;
0BFB  0E9B R                      DW      GET_BPB         ; 2  - Build BPB
0BFD  16                          DB      22              ; Length of drive request structure
                                                          ;
0BFE  0D38 R                      DW      CMDERR          ; 3  - IOCTL INPUT (currently returns error)
0C00  16                          DB      22              ; Length of drive request structure
                                                          ;
0C01  002E R                      DW      DREAD           ; 4  - READ
0C03  16                          DB      22              ; Length of drive request structure
                                                          ;
0C04  0000                        DW      0000            ; 5  - NON DESTRUCTIVE INPUT (char. devices)
0C06  00                          DB      00              ; Length of drive request structure
                                                          ;
0C07  0000                        DW      0000            ; 6  - INPUT STATUS (char. devices)
0C09  00                          DB      00              ; Length of drive request structure
                                                          ;
0C0A  0000                        DW      0000            ; 7  - INPUT FLUSH (char. devices)
0C0C  00                          DB      00              ; Length of drive request structure
                                                          ;
0C0D  0037 R                      DW      DWRITE          ; 8  - WRITE
0C0F  16                          DB      22              ; Length of drive request structure
                                                          ;
0C10  1112 R                      DW      DVERIFY         ; 9  - WRITE WITH VERIFY
0C12  16                          DB      22              ; Length of drive request structure
                                                          ;
0C13  0000                        DW      0000            ; 10 - OUTPUT STATUS (char. devices)
0C15  00                          DB      00              ; Length of drive request structure
                                                          ;
0C16  0000                        DW      0000            ; 11 - OUTPUT FLUSH (char. devices)
0C18  00                          DB      00              ; Length of drive request structure
                                                          ;
0C19  0D38 R                      DW      CMDERR          ; 12 - IOCTL OUTPUT (currently returns error)
0C1B  16                          DB      22              ; Length of drive request structure
                                                          ;
```

```
                           ;***************************************************************************
                           ;***************************************************************************
                           ;***************************************************************************
                           ;
                           ;
                           ;   ROUTINE NAME:       DSK_INT
                           ;
                           ;
                           ;   FUNCTION:           DISK INTERRUPT ROUTINE FOR PROCESSING I/O PACKETS
                           ;
                           ;
                           ;   ENTRY VIA:          CALL
                           ;
                           ;
                           ;   ENTRY CONDITIONS:   NONE
                           ;
                           ;
                           ;   EXIT VIA:           RETURN
                           ;
                           ;
                           ;   EXIT CONDITIONS:    SOME DISK VARIABLES ARE SET
                           ;
                           ;***************************************************************************
                           ;***************************************************************************
                           ;***************************************************************************
                           ;
0C1C                       DSK_INT:                          ;
0C1C  56                           PUSH    SI                ;
0C1D  50                           PUSH    AX                ;
0C1E  51                           PUSH    CX                ;
0C1F  52                           PUSH    DX                ;
0C20  57                           PUSH    DI                ;
0C21  55                           PUSH    BP                ;
0C22  1E                           PUSH    DS                ;
0C23  06                           PUSH    ES                ;
0C24  53                           PUSH    BX                ;
                                                             ;
0C25  0E                           PUSH    CS                ;
0C26  1F                           POP     DS                ; Set DATA SEGMENT to CODE SEGMENT
                                                             ;
0C27  C4 1E 0000 E                 LES     BX,[PTRSAV]       ; Retrieve pointer to I/O data packet
0C2B  8C 06 0003 R                 MOV     ES_SAVE,ES        ;
0C2F  89 1E 0005 R                 MOV     BX_SAVE,BX        ; Save pointer to I/O data packet
                                                             ;
0C33  26: 8B 47 12                 MOV     AX,ES:[BX.COUNT];
0C37  A3 000C R                    MOV     SECCNT,AX         ; PIM.SECCNT <-- sector count
                                                             ;
0C3A  26: 8B 47 0E                 MOV     AX,ES:[BX.TRANS];
0C3E  A3 001F R                    MOV     DMAADDR,AX        ; PIM.DMAADDR <-- transfer addr (OFFSET)
0C41  26: 8B 47 10                 MOV     AX,ES:[BX.TRANS+2] ;
0C45  A3 0021 R                    MOV     DMAADDR+2,AX      ; PIM.DMAADDR <-- transfer addr (SEGMENT)
                                                             ;
0C48  26: 8A 47 01                 MOV     AL,ES:[BX.UNIT] ;
0C4C  A2 000B R                    MOV     DRV,AL            ; PIM.DRV <-- unit code
                                                             ;
0C4F  80 3E 0000 E 01              CMP     FLOPPY_DRIVES,1 ;
0C54  75 1C                        JNZ     INT1              ; Jump if a two drive system
```

```
                                                      ;
                                                      ; Single drive system
                                                      ; ...................
0C56  3A 06 0002 E               CMP    AL,FL_FLAGS+2  ; Request to same drive ?
0C5A  A2 0002 E                  MOV    FL_FLAGS+2,AL  ; Update regardless
0C5D  C6 06 0000 R 00            MOV    DRV,00         ; and use drive 0
0C62  74 0E                      JZ     INT1           ; Jump if the same drive
0C64                      INT0:                        ; else
0C64  E8 11EB R                  CALL   PRDICH         ; Prompt disk change message
0C67  53                         PUSH   BX             ;
0C68  E8 0BA5 R                  CALL   MOTORCK        ;
0C6B  5B                         POP    BX             ;
0C6C  E4 13                      IN     AL,SYSSTA      ;
0C6E  A8 80                      TEST   AL,80          ; Test DISK INTERRUPT BIT
0C70  74 F2                      JZ     INT0           ; Jump if no interrupt
0C72                      INT1:                        ;
0C72  26: 8A 47 01               MOV    AL,ES:[BX.UNIT] ;
0C76  B4 13 90 90                MOV    AH,BPB_SIZE    ;
0C7A  F6 E4                      MUL    AH             ;
0C7C  BE 107A R                  MOV    SI,OFFSET BPB0 ;
0C7F  03 F0                      ADD    SI,AX          ; SI - addr of BPB of selected drive
                                                       ;
0C81  8B 44 0D                   MOV    AX,[SI.SECPTR] ;
0C84  A2 0028 R                  MOV    SECTRK,AL      ; PIM.SECTRK <-- sectors per track
                                                       ;
0C87  8B 44 0F                   MOV    AX,[SI.HEADS]  ;
0C8A  C6 06 0007 R 00            MOV    CYLMODE,00     ; PIM.CYLMODE <-- 00  if 2 heads
0C8F  3C 02                      CMP    AL,02          ;
0C91  74 05                      JZ     INT2           ;
                                                       ;
0C93  C6 06 0007 R 01            MOV    CYLMODE,01     ; PIM.CYLMODE <-- 01  if 1 head
0C98                      INT2:                        ;
0C98  8B 04                      MOV    AX,[SI.SECSIZE] ;
0C9A  3D 0080                    CMP    AX,80H         ;
0C9D  B0 00                      MOV    AL,00          ; N <-- 00  if sector size = 128
0C9F  74 0D                      JZ     INT4           ;
0CA1                      INT3:                        ;
0CA1  FE C0                      INC    AL             ; N <-- 01  if sector size = 256
0CA3  D0 DC                      RCR    AH,1           ;       02                   512
0CA5  72 07                      JC     INT4           ;       03                  1024
                                                       ;
0CA7  3C 03                      CMP    AL,03          ; If sector size > 1024
0CA9  72 F6                      JB     INT3           ;
                                                       ; then
0CAB  E9 0DE4 R                  JMP    ERROR_7        ; 'UNKNOWN MEDIA'
0CAE                      INT4:                        ;
0CAE  A2 002A R                  MOV    BYTSEC,AL      ; PIM.BYTSEC <-- N
                                                       ;
0CB1  8B 44 08                   MOV    AX,[SI.SECTORS] ; AX <-- total number of sectors
0CB4  F7 24                      MUL    WORD PTR [SI.SECSIZE] ; * sector size in bytes
                                                       ; gives total disk capacity in bytes
0CB6  8A C4                      MOV    AL,AH          ;
0CB8  8A E2                      MOV    AH,DL          ;
0CBA  D1 E8                      SHR    AX,1           ;
0CBC  D1 E8                      SHR    AX,1           ; AX - total disk capacity in KB
                                                       ;
0CBE  8B 4C 0F                   MOV    CX,[SI.HEADS]  ;
0CC1  49                         DEC    CX             ;
0CC2  D3 E8                      SHR    AX,CL          ; / sideness gives disk capacity per surface
                                                       ;
0CC4  3D 00C8                    CMP    AX,200         ; Check if disk capacity per surface > 200 KB
```

```
0CC7  C6 06 0027 R 00                MOV    TPI_DI,00       ; PIM.TPI_DI <-- 48 tpi if below or equal
0CCC  76 05                          JBE    INT5            ;
                                                            ;
0CCE  C6 06 0027 R 01                MOV    TPI_DI,01       ; PIM.TPI_DI <-- 96 tpi if above
0CD3                         INT5:                          ;
0CD3  B4 00                          MOV    AH,00           ;
0CD5  A0 0028 R                      MOV    AL,SECTRK       ;
0CD8  F7 64 0F                       MUL    [SI.HEADS]      ;
0CDB  8B C8                          MOV    CX,AX           ; CX <-- sectors per cylinder
                                                            ;
0CDD  26: 8B 47 14                   MOV    AX,ES:[BX.START]; AX <-- start sector
0CE1  33 D2                          XOR    DX,DX           ;
0CE3  F7 F1                          DIV    CX              ; AX - track
0CE5  42                             INC    DX              ; DX - sector (MS-DOS starts with log sector 0)
0CE6  A2 000A R                      MOV    TRACK,AL        ; PIM.TRACK <-- track
                                                            ;
0CE9  3A 16 0028 R                   CMP    DL,SECTRK       ; Test for side 0 or side 1
0CED  C6 06 0009 R 00                MOV    HEAD,00         ; PIM.HEAD <-- 0
0CF2  88 16 000B R                   MOV    SECTOR,DL       ; PIM.SECTOR <-- sector
0CF6  76 0D                          JBE    INT6            ; Jump if side 0
                                                            ;
0CF8  2A 16 0028 R                   SUB    DL,SECTRK       ; DL - sector within track
0CFC  C6 06 0009 R 01                MOV    HEAD,01         ; PIM.HEAD <-- 1
0D01  88 16 000B R                   MOV    SECTOR,DL       ; PIM.SECTOR <-- sector
0D05                         INT6:                          ;
0D05  BE 0BF5 R                      MOV    SI,OFFSET DSKTBL; SI <-- addr of disk-table
0D08  B4 00                          MOV    AH,00           ;
0D0A  26: 8A 47 02                   MOV    AL,ES:[BX.CMD]  ; AX <-- command code
0D0E  03 F0                          ADD    SI,AX           ;
0D10  03 F0                          ADD    SI,AX           ; Compute entry in disk-table
0D12  03 F0                          ADD    SI,AX           ;
                                                            ;
0D14  3C 0C                          CMP    AL,12           ; If more than 12 commands
0D16  76 03                          JBE    INT7            ; then
0D18  E9 0DD4 R                      JMP    ERROR_3         ; 'UNKNOWN COMMAND'
0D1B                         INT7:                          ;
0D1B  26: 8A 07                      MOV    AL,ES:[BX.CMDLEN] ;
0D1E  80 7C 02 00                    CMP    BYTE PTR [SI+2],00 ;
0D22  74 11                          JZ     INT9            ; Skip character device commands
                                                            ;
0D24  3A 44 02                       CMP    AL,[SI+2]       ; If wrong length
0D27  73 03                          JAE    INT8            ; then
0D29  E9 0DDC R                      JMP    ERROR_5         ; 'BAD DRIVE REQUEST STRUCTURE LENGTH'
0D2C                         INT8:                          ;
0D2C  FF 14                          CALL   [SI]            ; Perform I/O packet command
                                                            ;
0D2E  F6 06 0010 R C0                TEST   ERRBUF,0C0H     ; Test for normal termination
0D33  75 07                          JNZ    DSKERR          ; Jump to disk error routine
0D35                         INT9:                          ;
0D35  E9 0DFF R                      JMP    EXIT            ; Jump to EXIT
                                                            ;
0D38                         CMDERR:                        ;
0D38  58                             POP    AX              ; Flush return address from stack
0D39  E9 0DD4 R                      JMP    ERROR_3         ; Generate 'UNKNOWN COMMAND' error
```

```
0D3C                          DSKERR:                            ;
0D3C  8E 06 0003 R                    MOV     ES,ES_SAVE         ;
0D40  8B 1E 0005 R                    MOV     BX,BX_SAVE         ; Retrieve pointer to I/O data packet
0D44  26: 8A 47 02                    MOV     AL,ES:[BX.CMD]     ; AL <-- Command code
0D48  3C 04                           CMP     AL,04              ; Test if READ
0D4A  74 0A                           JZ      DSKERR0            ; Jump if READ
0D4C  3C 08                           CMP     AL,08              ; Test if WRITE
0D4E  74 06                           JZ      DSKERR0            ; Jump if WRITE
0D50  3C 09                           CMP     AL,09              ; Test if VERIFY
0D52  74 02                           JZ      DSKERR0            ; Jump if VERIFY
                                                                 ;
0D54  EB 06                           JMP     SHORT DSKERR1      ; Jump if not READ, WRITE or VERIFY
0D56                          DSKERR0:                           ;
0D56  26: C7 47 12 0000               MOV     ES:[BX.COUNT],0    ; Transfer counter <-- 0000
0D5C                          DSKERR1:                           ;
0D5C  F6 06 0019 R 02                 TEST    ERRBUF+1,02        ; Test for 'WRITE PROTECTED'
0D61  74 03                           JZ      DSKERR2            ;
0D63  EB 63 90                        JMP     ERROR_0            ; Jump if 'WRITE PROTECTED'
0D66                          DSKERR2:                           ;
0D66  F6 06 0018 R 08                 TEST    ERRBUF,08          ; Test for 'DRIVE NOT READY'
0D6B  74 03                           JZ      DSKERR3            ;
0D6D  EB 61 90                        JMP     ERROR_2            ; Jump if 'DRIVE NOT READY'
0D70                          DSKERR3:                           ;
0D70  F6 06 0018 R 80                 TEST    ERRBUF,80H         ; Test for 'UNKNOWN COMMAND'
0D75  74 03                           JZ      DSKERR4            ;
0D77  EB 5B 90                        JMP     ERROR_3            ; Jump if 'UNKNOWN COMMAND'
0D7A                          DSKERR4:                           ;
0D7A  F6 06 0019 R 20                 TEST    ERRBUF+1,20H       ; Test for 'CRC ERROR'
0D7F  74 03                           JZ      DSKERR5            ;
0D81  EB 55 90                        JMP     ERROR_4            ; Jump if 'CRC ERROR'
0D84                          DSKERR5:                           ;
0D84  F6 06 0018 R 20                 TEST    ERRBUF,20H         ; Test for 'SEEK ERROR'
0D89  75 0A                           JNZ     DSKERR6            ;
0D8B  F6 06 001A R 10                 TEST    ERRBUF+2,10H       ; Test for 'SEEK ERROR'
0D90  74 03                           JZ      DSKERR6            ;
0D92  EB 4C 90                        JMP     ERROR_6            ; Jump if 'SEEK ERROR'
0D95                          DSKERR6:                           ;
0D95  F6 06 001A R 01                 TEST    ERRBUF+2,01        ; Test for 'UNKNOWN MEDIA'
0D9A  74 03                           JZ      DSKERR7            ;
0D9C  EB 46 90                        JMP     ERROR_7            ; Jump if 'UNKNOWN MEDIA'
0D9F                          DSKERR7:                           ;
0D9F  F6 06 0019 R 04                 TEST    ERRBUF+1,04        ; Test for 'SECTOR NOT FOUND'
0DA4  74 03                           JZ      DSKERR8            ;
0DA6  EB 40 90                        JMP     ERROR_8            ; Jump if 'SECTOR NOT FOUND'
0DA9                          DSKERR8:                           ;
0DA9  B4 05                           MOV     AH,WRITDAT         ;
0DAB  22 26 000F R                    AND     AH,COMSTR+1        ;
0DAF  80 FC 05                        CMP     AH,WRITDAT         ;
0DB2  75 03                           JNZ     DSKERR9            ;
0DB4  EB 3A 90                        JMP     ERROR_10           ; Jump if error after WRITE COMMAND
0DB7                          DSKERR9:                           ;
0DB7  B4 06                           MOV     AH,READDAT         ;
0DB9  22 26 000F R                    AND     AH,COMSTR+1        ;
0DBD  80 FC 06                        CMP     AH,READDAT         ;
0DC0  75 03                           JNZ     DSKERR10           ;
0DC2  EB 30 90                        JMP     ERROR_11           ; Jump if error after READ COMMAND
0DC5                          DSKERR10:                          ;
0DC5  EB 31 90                        JMP     ERROR_12           ; Rest becomes 'GENERAL FAILURE'
```

```
                          ;
                          ; Common error processing routine.
                          ;  AL contains actual error code.
                          ;
                          ;  Error # 0 = Write Protect violation.
                          ;          1 = Unkown unit.
                          ;          2 = Drive not ready.
                          ;          3 = Unknown command in I/O packet.
                          ;          4 = CRC error.
                          ;          5 = Bad drive request structure length.
                          ;          6 = Seek error.
                          ;          7 = Unknown media discovered.
                          ;          8 = Sector not found.
                          ;          9 = Printer out of paper.
                          ;         10 = Write fault.
                          ;         11 = Read fault.
                          ;         12 = General failure.
                          ;

0DC8                      ERROR_0:
0DC8  32 C0                       XOR    AL,AL          ;Write protect violation.
0DCA  EB 2E                       JMP    SHORT ERR_EXIT
0DCC                      ERROR_1:
0DCC  B0 01                       MOV    AL,1           ;Unknown unit.
0DCE  EB 2A                       JMP    SHORT ERR_EXIT
0DD0                      ERROR_2:
0DD0  B0 02                       MOV    AL,2           ;Drive not ready.
0DD2  EB 26                       JMP    SHORT ERR_EXIT
0DD4                      ERROR_3:
0DD4  B0 03                       MOV    AL,3           ;Unknown command in I/O packet.
0DD6  EB 22                       JMP    SHORT ERR_EXIT
0DD8                      ERROR_4:
0DD8  B0 04                       MOV    AL,4           ;CRC error.
0DDA  EB 1E                       JMP    SHORT ERR_EXIT
0DDC                      ERROR_5:
0DDC  B0 05                       MOV    AL,5           ;Bad drive request structure length.
0DDE  EB 1A                       JMP    SHORT ERR_EXIT
0DE0                      ERROR_6:
0DE0  B0 06                       MOV    AL,6           ;Seek error.
0DE2  EB 16                       JMP    SHORT ERR_EXIT
0DE4                      ERROR_7:
0DE4  B0 07                       MOV    AL,7           ;Unknown media discovered.
0DE6  EB 12                       JMP    SHORT ERR_EXIT
0DE8                      ERROR_8:
0DE8  B0 08                       MOV    AL,8           ;Sector not found.
0DEA  EB 0E                       JMP    SHORT ERR_EXIT
0DEC                      ERROR_9:
0DEC  B0 09                       MOV    AL,9           ;Printer out of paper.
0DEE  EB 0A                       JMP    SHORT ERR_EXIT
0DF0                      ERROR_10:
0DF0  B0 0A                       MOV    AL,10          ;Write fault.
0DF2  EB 06                       JMP    SHORT ERR_EXIT
0DF4                      ERROR_11:
0DF4  B0 0B                       MOV    AL,11          ;Read fault.
0DF6  EB 02                       JMP    SHORT ERR_EXIT
0DF8                      ERROR_12:
0DF8  B0 0C                       MOV    AL,12          ;General failure.
                          ;
                          ;fall through to ERR_EXIT
                          ;
```

```
0DFA                    ERR_EXIT:
0DFA  B4 81                     MOV     AH,10000001B    ;Set error and done bits.
0DFC  F9                        STC                     ;Set carry bit also.
0DFD  EB 02                     JMP     SHORT EXIT1     ;Quick way out.

0DFF                    EXITP   PROC    FAR             ;

0DFF  B4 01             EXIT:   MOV     AH,00000001B    ;Set done bit for MSDOS.
0E01  8B 1E 0005 R      EXIT1:  MOV     BX,BX_SAVE
0E05  8E 1E 0003 R              MOV     DS,ES_SAVE
0E09  89 47 03                  MOV     [BX.STATUS],AX  ;Save operation complete and status.

0E0C  5B                        POP     BX              ;Restore registers.
0E0D  07                        POP     ES
0E0E  1F                        POP     DS
0E0F  5D                        POP     BP
0E10  5F                        POP     DI
0E11  5A                        POP     DX
0E12  59                        POP     CX
0E13  58                        POP     AX
0E14  5E                        POP     SI
0E15  CB                        RET
0E16                    EXITP   ENDP
```

```
                                   ;**************************************************************************
                                   ;**************************************************************************
                                   ;**************************************************************************
                                   ;
                                   ;
                                   ;   ROUTINE NAME:       DSK_INIT
                                   ;
                                   ;
                                   ;   FUNCTION:           DISK INITIALIZE
                                   ;
                                   ;
                                   ;   ENTRY VIA:          CALL
                                   ;
                                   ;
                                   ;   ENTRY CONDITIONS:   NONE
                                   ;
                                   ;
                                   ;   EXIT VIA:           RETURN
                                   ;
                                   ;
                                   ;   EXIT CONDITIONS:    NONE
                                   ;
                                   ;**************************************************************************
                                   ;**************************************************************************
                                   ;**************************************************************************
                                   ;
                                   ;
0E16                               DSK_INIT:                         ;
0E16  26: C6 47 0D 02                      MOV    ES:[BX.MEDIA],2 ; I/O data packet <-- 2 units
                                                                  ;
0E1B  BE 1211 R                            MOV    SI,OFFSET DREND ;
0E1E  26: 89 77 0E                         MOV    ES:[BX.TRANS],SI; I/O data packet <-- BREAK ADDR (OFFSET)
0E22  26: 8C 4F 10                         MOV    ES:[BX.TRANS+2],CS ;          <--            (SEGMENT)
                                                                  ;
0E26  BE 1076 R                            MOV    SI,OFFSET INITTAB ;
0E29  26: 89 77 12                         MOV    ES:[BX.COUNT],SI; I/O data packet <-- Pointer to
0E2D  26: 8C 4F 14                         MOV    ES:[BX.START],CS;                     BPB array
                                                                  ;
0E31  A0 0000 E                            MOV    AL,FL_OUT_RETRIES ;
0E34  A2 002D R                            MOV    RETRIES,AL      ; Set retry counter
                                                                  ;
0E37  C7 06 0000 E 0101                    MOV    FLTAB,0101H     ; Set flags in common area
0E3D  C6 06 0000 E 00                      MOV    BYTE PTR FL_FLAGS,00 ;
0E42  C6 06 0001 E 00                      MOV    BYTE PTR FL_FLAGS+1,00 ;
                                                                  ;
                                                                  ; Set shortest HLT to improve the performance
                                                                  ; ........................................
0E47  C6 06 000E R 03                      MOV    COMSTR,03       ; COMMAND STRING <-- LENGTH 3
0E4C  C6 06 000F R 03                      MOV    COMSTR+1,SPECIFY;                <-- SPECIFY COMMAND
0E51  C6 06 0010 R D6                      MOV    COMSTR+2,0D6H   ;                <-- STEPPING RATE & HUT
0E56  C6 06 0011 R 02                      MOV    COMSTR+3,02     ;                <-- HLT & ND
                                                                  ;
0E5B  E8 0B70 R                            CALL   XWAIT           ; Send COMMAND STRING to FDC
                                                                  ;
0E5E  C6 06 0010 R 00                      MOV    ERRBUF,00       ; Set normal termination
0E63  C3                                   RET                    ;
```

```
                                                ;
                         ;************************************************************************
                         ;************************************************************************
                         ;************************************************************************
                         ;
                         ;
                         ;    ROUTINE NAME:      MEDIAC
                         ;
                         ;
                         ;    FUNCTION:          MEDIA CHECK
                         ;
                         ;
                         ;    ENTRY VIA:         CALL
                         ;
                         ;
                         ;    ENTRY CONDITIONS:  NONE
                         ;
                         ;
                         ;    EXIT VIA:          RETURN
                         ;
                         ;
                         ;    EXIT CONDITIONS:   NONE
                         ;
                         ;************************************************************************
                         ;************************************************************************
                         ;************************************************************************
                         ;
                         ;
0E64                     MEDIAC:                                  ;
0E64  80 3E 0000 E 01            CMP    FLOPPY_DRIVES,1 ;
0E69  74 06                      JZ     MEDIAC1         ; Jump if a single drive system
                                                        ;
0E6B  E4 13                      IN     AL,SYSSTA       ;
0E6D  A8 08                      TEST   AL,08           ; Test DISK INTERRUPT BIT
0E6F  74 06                      JZ     MEDIAC2         ; Jump if no interrupt
0E71                     MEDIAC1:                       ;
0E71  C7 06 0000 E 0101          MOV    FLTAB,0101H     ; Set interrupt flags for both units
0E77                     MEDIAC2:                       ;
0E77  C6 06 0018 R 00            MOV    ERRBUF,00       ; Set normal termination
                                                        ;
0E7C  BE 0000 E                  MOV    SI,OFFSET FLTAB ; SI <-- Addr of FLAG TABLE
0E7F  B4 00                      MOV    AH,00           ;
0E81  A0 0000 R                  MOV    AL,DRV          ; AX <-- Unit code
0E84  03 F0                      ADD    SI,AX           ; Compute entry in FLAG TABLE
                                                        ;
0E86  80 3C 00                   CMP    BYTE PTR [SI],0 ; Check if interrupt flag is set for that unit
0E89  74 09                      JZ     MEDIAC3         ; Jump if not set
                                                        ;
0E8B  C6 04 00                   MOV    BYTE PTR [SI],0 ; Reset interrupt flag
0E8E  26: C6 47 0E 00            MOV    BYTE PTR ES:[BX.TRANS],00 ; I/O data packet <-- don't know
0E93  C3                         RET                    ;
0E94                     MEDIAC3:                       ;
0E94  26: C6 47 0E 01            MOV    BYTE PTR ES:[BX.TRANS],01 ; I/O data packet <-- not changed
0E99  C3                         RET                    ;
```

```
                                ;################################################################################
                                ;################################################################################
                                ;################################################################################
                                ;
                                ;
                                ;  ROUTINE NAME:       GET_BPB
                                ;
                                ;
                                ;  FUNCTION:           Get BPB (BIOS PARAMETER BLOCK)
                                ;
                                ;
                                ;  ENTRY VIA:          CALL
                                ;
                                ;
                                ;  ENTRY CONDITIONS:   DRV variable is set
                                ;
                                ;
                                ;  EXIT VIA:           RETURN
                                ;
                                ;
                                ;  EXIT CONDITIONS:    SOME DISK VARIABLES ARE SET
                                ;                      STATUS (returned in ERRBUF)
                                ;
                                ;################################################################################
                                ;################################################################################
                                ;################################################################################

0E9A                            EC:                             ; Entry counter
0E9A  90                                DB      90H             ;
0E9B                            GET_BPB:                        ; Fill PIM variables to read BOOT SECTOR
                                                                ; ---------------------------------------
                                                                ;
0E9B  C6 06 0027 R 00                   MOV     TPI_DI,00       ; PIM.TPI_DI <-- 48 tpi disk
0EA0  C6 06 0007 R 00                   MOV     CYLMODE,00      ; PIM.CYLINDER MODE <-- 00
                                                                ; PIM.DRV already set
0EA5  C6 06 0009 R 00                   MOV     HEAD,00         ; PIM.HEAD <-- 00
0EAA  C6 06 000A R 00                   MOV     TRACK,00        ; PIM.TRACK <-- 00
0EAF  C6 06 000B R 01                   MOV     SECTOR,01       ; PIM.SECTOR <-- 1st phy  = 0th log
0EB4  C7 06 000C R 0001                 MOV     SECCNT,0001     ; PIM.SECCNT <-- one sector
                                                                ; DMA ADDRESS of scratch buffer already set
                                                                ;
0EBA  E8 002E R                         CALL    DREAD           ; Read BOOT SECTOR
                                                                ;
0EBD  F6 06 0018 R C0                   TEST    ERRBUF,0C0H     ; Test for normal termination
0EC2  74 1A                             JZ      GETBPB1         ; Jump if normal termination
                                                                ;
0EC4  FE 06 0E9A R                      INC     BYTE PTR EC     ;
0EC8  FE 06 002A R                      INC     BYTSEC          ;
0ECC  80 26 002A R 03                   AND     BYTSEC,03       ; BYTSEC <-- next sector size
                                                                ;
0ED1  80 3E 0E9A R 94                   CMP     BYTE PTR EC,94H ; Maximum of 4 attempts (BYTSEC = 0, 1, 2, 3)
0ED6  72 C3                             JB      GET_BPB         ; Jump to try again
0ED8  C6 06 0E9A R 90                   MOV     BYTE PTR EC,90H ; else reset EC
0EDD  C3                                RET                     ; and return
```

```
0EDE                    GETBPB1:                            ;
0EDE  C6 06 0E9A R 90           MOV    BYTE PTR EC,90H ; Reset EC
0EE3  8E 06 0003 R              MOV    ES,ES_SAVE      ;
0EE7  8B 1E 0005 R              MOV    BX,BX_SAVE      ;
0EEB  8B 36 001F R              MOV    SI,DMAADDR      ;
0EEF  83 C6 0B                  ADD    SI,11           ; SI <-- scratch BUFFER.BPB
                                                       ;
0EF2  B4 13 90 90               MOV    AH,BPB_SIZE     ;
0EF6  26: 8A 47 01              MOV    AL,ES:[BX.UNIT] ;
0EFA  F6 E4                     MUL    AH              ;
0EFC  BF 107A R                 MOV    DI,OFFSET BPB0  ;
0EFF  03 F8                     ADD    DI,AX           ; DI <-- BUFFER AREA for BPB of selected DRIVE
0F01  57                        PUSH   DI              ; Save addr of BUFFER AREA
                                                       ;
                                                       ; Update I/O data packet
0F02  26: 89 7F 12              MOV    ES:[BX.COUNT],DI;      <-- BPB POINTER
0F06  26: 8C 4F 14              MOV    ES:[BX.COUNT+2],CS;    <-- CODE SEGMENT
                                                       ;
0F0A  1E                        PUSH   DS              ;
0F0B  07                        POP    ES              ; ES <-- DS
0F0C  A1 0021 R                 MOV    AX,DMAADDR+2    ;
0F0F  8E D8                     MOV    DS,AX           ; DS <-- scratch BUFFER SEGMENT
                                                       ;
0F11  FC                        CLD                    ; incrementing
0F12  B9 0013 90                MOV    CX,BPB_SIZE     ; Length of (extended) BPB
0F16  F3/ A4            REP     MOVSB                  ; Move BPB from scratch BUFFER into BUFFER AREA
0F18  06                        PUSH   ES              ;
0F19  1F                        POP    DS              ; DS <-- ES
                                                       ;
001     0F1A  5E                       POP    SI             ; Restore addr of BUFFER AREA
0F1B  8B 44 0B                  MOV    AX,WORD PTR [SI.SECTORS] ; AX <-- total number of sectors
0F1E  0B C0                     OR     AX,AX           ; Check if no BPB in BOOT SECTOR (IBM - Formats)
0F20  75 03                     JNZ    GETBPB2         ;
                                                       ;
0F22  E9 0FF7 R                 JMP    GETBPB10        ; If so, jump to determine BPB by FAT-ID
0F25                    GETBPB2:                       ;
0F25  8A 44 0A                  MOV    AL,BYTE PTR [SI.MEDIAID] ; AL <-- media descriptor byte
0F28  3C F8                     CMP    AL,0F8H         ;
0F2A  73 03                     JNB    GETBPB3         ;
                                                       ;
0F2C  E9 0FF7 R                 JMP    GETBPB10        ; Jump if unknown media descriptor (CPM - Formats)
0F2F                    GETBPB3:                       ;
0F2F  3C FC                     CMP    AL,0FCH         ;
0F31  74 06                     JZ     GETBPB4         ; Jump if single sided NCR - Format
0F33  3C FE                     CMP    AL,0FEH         ;
0F35  74 02                     JZ     GETBPB4         ; Jump if single sided NCR - Format
0F37  EB 05                     JMP    SHORT GETBPB5   ;
0F39                    GETBPB4:                       ;
0F39  C7 44 0F 0001             MOV    WORD PTR [SI.HEADS],0001 ; HEADS <-- 1  if single sided NCR - Format
0F3E                    GETBPB5:                       ;
0F3E  8E 06 0003 R              MOV    ES,ES_SAVE      ;
0F42  8B 1E 0005 R              MOV    BX,BX_SAVE      ; Update I/O data packet
0F46  26: 88 47 0D              MOV    ES:[BX.MEDIA],AL;      <-- media descriptor
```

```
                                                  ;
0F4A                      GETBPB6:                ;* tpi determination of drive
                                                  ;* -------------------------
0F4A  80 3E 0000 E 00           CMP   BYTE PTR FL_FLAGS,00 ;*
0F4F  74 07                     JZ    GETBPB7     ;* Jump to set TPI_DR
0F51  A0 0001 E                 MOV   AL,FL_FLAGS+1  ;* else
0F54  A2 0026 R                 MOV   TPI_DR,AL   ;* PIM.TPI_DR <-- value from common area
0F57  C3                        RET               ;* and return
0F58                      GETBPB7:                ;*
                                                  ;* 1) Check if tpi from drive & disk are equal
                                                  ;* ........................................
0F58  C6 06 0026 R 01           MOV   TPI_DR,01   ;* PIM.TPI_DR <-- 96 tpi
0F5D  C6 06 0027 R 01           MOV   TPI_DI,01   ;* PIM.TPI_DI <-- 96 tpi
                                                  ;* PIM.CYLMODE already set
                                                  ;* PIM.DRV already set
0F62  C6 06 000A R 02           MOV   TRACK,02    ;* PIM.TRACK <-- 2
0F67  C6 06 000B R 01           MOV   SECTOR,01   ;* PIM.SECTOR <-- 1
0F6C  C7 06 000C R 0001         MOV   SECCNT,0001 ;* PIM.SECCNT <-- one sector
                                                  ;* PIM.DMAADDR already set
                                                  ;*
0F72  E8 002E R                 CALL  DREAD       ;* Try to read
                                                  ;*
0F75  F6 06 0018 R C0           TEST  ERRBUF,0C0H ;* Test for normal termination
0F7A  74 20                     JZ    GETBPB8     ;* Jump if tpi from drive & disk are equal
                                                  ;* else
                                                  ;* try with other tpi variables
                                                  ;*
                                                  ;* 2) Check if 48 tpi disk in 96 tpi drive
                                                  ;* ......................................
                                                  ;* PIM.TPI_DR <-- 96 tpi already set
0F7C  C6 06 0027 R 00           MOV   TPI_DI,00   ;* PIM.TPI_DI <-- 48 tpi
                                                  ;* PIM.CYLMODE already set
                                                  ;* PIM.DRV already set
                                                  ;* PIM.TRACK already set
                                                  ;* PIM.SECTOR already set
0F81  C7 06 000C R 0001         MOV   SECCNT,0001 ;* PIM.SECCNT <-- one sector
                                                  ;* PIM.DMAADDR already set
                                                  ;*
0F87  E8 002E R                 CALL  DREAD       ;* Try to read
                                                  ;*
0F8A  F6 06 0018 R C0           TEST  ERRBUF,0C0H ;* Test for normal termination
0F8F  74 44                     JZ    GETBPB9     ;* Jump if 48 tpi disk in 96 tpi drive
                                                  ;* else
0F91  C6 06 0018 R 40           MOV   BYTE PTR ERRBUF,40H ;* 'UNKNOWN MEDIA'
0F96  C6 06 001A R 01           MOV   BYTE PTR ERRBUF+2,01 ;* and
0F9B  C3                        RET               ;* return
```

```
                                                      ;*
0F9C                        GETBPB8:                  ;* determine tpi of drive from disk capacity
                                                      ;* ......................................
0F9C  8E 06 0003 R               MOV    ES,ES_SAVE     ;*
0FA0  8B 1E 0005 R               MOV    BX,BX_SAVE     ;*
0FA4  26: 8B 77 12               MOV    SI,ES:[BX.COUNT];* SI <-- addr of BPB
                                                      ;*
0FA8  8B 44 08                   MOV    AX,[SI.SECTORS] ;* AX <-- total number of sectors
0FAB  F7 24                      MUL    WORD PTR [SI.SECSIZE] ;* multiplied with sector size in bytes
                                                      ;* gives total disk capacity in bytes
0FAD  8A C4                      MOV    AL,AH          ;*
0FAF  8A E2                      MOV    AH,DL          ;*
0FB1  D1 E8                      SHR    AX,1           ;*
0FB3  D1 E8                      SHR    AX,1           ;* AX - total disk capacity in KB
                                                      ;*
0FB5  8B 4C 0F                   MOV    CX,[SI.HEADS]  ;*
0FB8  49                         DEC    CX             ;*
0FB9  D3 E8                      SHR    AX,CL          ;* / sideness gives disk capacity per surface
                                                      ;*
0FBB  3D 00C8                    CMP    AX,200         ;* Check if disk capacity per surface > 200 KB
0FBE  77 15                      JA     GETBPB9        ;* Let variables unchanged (96 tpi) if above
                                                      ;*
0FC0  C6 06 0026 R 00            MOV    TPI_DR,00      ;* PIM.TPI_DR <-- 48 tpi if below or equal
0FC5  C6 06 0027 R 00            MOV    TPI_DI,00      ;* PIM.TPI_DI <-- 48 tpi if below or equal
                                                      ;*
0FCA  C6 06 0001 E 00            MOV    BYTE PTR FL_FLAGS+1,00 ;* Set 48 tpi drive in common area
0FCF  C6 06 0000 E FF            MOV    BYTE PTR FL_FLAGS,0FFH ;* Set flag
0FD4  C3                         RET                   ;*
0FD5                        GETBPB9:                  ;*
                                                      ;* Specify other stepping rate for 96 tpi drive
                                                      ;* ........................................
0FD5  C6 06 000E R 03            MOV    COMSTR,03      ;* COMMAND STRING <-- LENGTH 3
0FDA  C6 06 000F R 03            MOV    COMSTR+1,SPECIFY ;*            <-- SPECIFY COMMAND
0FDF  C6 06 0010 R E6            MOV    COMSTR+2,0E6H  ;*              <-- STEPPING RATE & HUT
0FE4  C6 06 0011 R 02            MOV    COMSTR+3,02    ;*              <-- HLT & ND
                                                      ;*
0FE9  E8 0B70 R                  CALL   XWAIT          ;* Send COMMAND STRING to FDC
                                                      ;*
0FEC  C6 06 0001 E 01            MOV    BYTE PTR FL_FLAGS+1,01 ;* Set 96 tpi drive in common area
0FF1  C6 06 0000 E FF            MOV    BYTE PTR FL_FLAGS,0FFH ;* Set flag
0FF6  C3                         RET                   ;*

0FF7                        GETBPB10:                 ; Fill PIM variables to read 1st FAT
                                                      ; ----------------------------------
                                                      ; PIM.CYLINDER MODE <-- already set
                                                      ; PIM.DRV already set
                                                      ; PIM.HEAD <-- already set
                                                      ; PIM.TRACK <-- already set
0FF7  C6 06 000B R 02            MOV    SECTOR,02      ; PIM.SECTOR <-- 2nd phy  = 1st log
0FFC  C7 06 000C R 0001          MOV    SECCNT,0001    ; PIM.SECCNT <-- one sector
                                                      ; DMA ADDRESS of scratch buffer already set
                                                      ;
1002  E8 082E R                  CALL   DREAD          ; Read 1st FAT sector
                                                      ;
1005  F6 06 0018 R C0            TEST   ERRBUF,0C0H    ; Test for normal termination
100A  74 01                      JZ     GETBPB11       ; Jump if normal termination
100C  C3                         RET                   ; else return
```

```
100D                        GETBPB11:                          ;
100D  8B 3E 001F R                  MOV     DI,DMAADDR         ;
1011  8E 06 0021 R                  MOV     ES,DMAADDR+2       ; ES:DI <-- scratch buffer address
1015  26: 8A 05                     MOV     AL,ES:[DI]         ; AL <-- media descriptor byte from FAT
                                                               ;
1018  BE 10FF R                     MOV     SI,OFFSET F9800 ; SI <-- Pointer to BPB
101B  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
101E  74 32                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1020  BE 10EC R                     MOV     SI,OFFSET FB720 ; SI <-- Pointer to BPB
1023  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
1026  74 2A                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1028  BE 10D9 R                     MOV     SI,OFFSET FC180 ; SI <-- Pointer to BPB
102B  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
102E  74 22                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1030  BE 10C6 R                     MOV     SI,OFFSET FD360 ; SI <-- Pointer to BPB
1033  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
1036  74 1A                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1038  BE 10B3 R                     MOV     SI,OFFSET FE160 ; SI <-- Pointer to BPB
103B  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
103E  74 12                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1040  BE 10A0 R                     MOV     SI,OFFSET FF320 ; SI <-- Pointer to BPB
1043  3A 44 0A                      CMP     AL,[SI.MEDIAID] ; Compare media descriptors in FAT & BPB
1046  74 0A                         JZ      GETBPB12           ; Jump if equal
                                                               ;
1048  C6 06 001B R 40               MOV     BYTE PTR ERRBUF,40H ;
104D  C6 06 001A R 01               MOV     BYTE PTR ERRBUF+2,01 ; No match - generate 'UNKNOWN MEDIA'
1052                        GETBPB12:                          ;
1052  8E 06 0003 R                  MOV     ES,ES_SAVE         ;
1056  8B 1E 0005 R                  MOV     BX,BX_SAVE         ; Update I/O data packet
105A  26: 88 47 0D                  MOV     ES:[BX.MEDIA],AL;        <-- media descriptor
                                                               ;
105E  26: 8B 7F 12                  MOV     DI,ES:[BX.COUNT]; DI <-- BUFFER AREA for BPB of selected drive
1062  1E                            PUSH    DS                 ;
1063  07                            POP     ES                 ; ES <-- DS
                                                               ;
1064  FC                            CLD                        ; incrementing
1065  B9 0013 90                    MOV     CX,BPB_SIZE        ; Length of (extended) BPB
1069  F3/ A4                REP     MOVSB                      ; Move BPB into BUFFER AREA
                                                               ;
106B  F6 06 001B R C0               TEST    ERRBUF,0C0H        ; Test for normal termination
1070  74 01                         JZ      GETBPB13           ; Jump if normal termination
1072  C3                            RET                        ; else return
1073                        GETBPB13:                          ;
1073  E9 0F4A R                     JMP     GETBPB6            ; Jump to determine tpi of drive
```

```
                                   ;
                                   ; ************************************
                                   ; ***  BPB's (BIOS PARAMETER BLOCKS)  ***
                                   ; ************************************
                                   ;
                                   ;
1076                               INITTAB:                           ;
1076  10FF R                              DW      F9000          ;
1078  10FF R                              DW      F9000          ;
                                                                      ;
= 107A                             BPB_START       EQU     $
                                                                      ;
107A                               BPB0:                              ; BUFFER AREA for BPB of DRIVE 0
                                                                      ; (with default values for initialisation)
                                                                      ;
107A  0200                                DW      512            ; Sector size
107C  02                                  DB      2              ; Sectors per allocation unit
107D  0001                                DW      1              ; Number of reserved sectors
107F  02                                  DB      2              ; Number of FATs
1080  0070                                DW      7*16           ; Number of directory entries
1082  02D0                                DW      2*40*9         ; Total number of sectors
1084  FD                                  DB      0FDH           ; Media byte
1085  0002                                DW      2              ; Sectors for one FAT
1087  0009                                DW      9              ; Extension: Sectors per track
1089  0002                                DW      2              ; Extension: Number of heads
108B  0000                                DW      0              ; Extension: Number of hidden sectors
                                                                      ;
= 0013                             BPB_SIZE        EQU     $-BPB_START
                                                                      ;
108D                               BPB1:                              ; BUFFER AREA for BPB of DRIVE 1
                                                                      ; (with default values for initialisation)
                                                                      ;
108D  0200                                DW      512            ; Sector size
108F  02                                  DB      2              ; Sectors per allocation unit
1090  0001                                DW      1              ; Number of reserved sectors
1092  02                                  DB      2              ; Number of FATs
1093  0070                                DW      7*16           ; Number of directory entries
1095  02D0                                DW      2*40*9         ; Total number of sectors
1097  FD                                  DB      0FDH           ; Media byte
1098  0002                                DW      2              ; Sectors for one FAT
109A  0009                                DW      9              ; Extension: Sectors per track
109C  0002                                DW      2              ; Extension: Number of heads
109E  0000                                DW      0              ; Extension: Number of hidden sectors

10A0                               FF320:                             ;
10A0  0200                                DW      512            ; Sector size
10A2  02                                  DB      2              ; Sectors per allocation unit
10A3  0001                                DW      1              ; Number of reserved sectors
10A5  02                                  DB      2              ; Number of FATs
10A6  0070                                DW      7*16           ; Number of directory entries
10A8  0280                                DW      2*40*8         ; Total number of sectors
10AA  FF                                  DB      0FFH           ; Media byte
10AB  0001                                DW      1              ; Sectors for one FAT
10AD  0008                                DW      8              ; Extension: Sectors per track
10AF  0002                                DW      2              ; Extension: Number of heads
10B1  0000                                DW      0              ; Extension: Number of hidden sectors
```

```
10B3                FE160:                  ;
10B3  0200                 DW    512        ; Sector size
10B5  01                   DB    1          ; Sectors per allocation unit
10B6  0001                 DW    1          ; Number of reserved sectors
10B8  02                   DB    2          ; Number of FATs
10B9  0040                 DW    4*16       ; Number of directory entries
10BB  0140                 DW    40*8       ; Total number of sectors
10BD  FE                   DB    0FEH       ; Media byte
10BE  0001                 DW    1          ; Sectors for one FAT
10C0  0008                 DW    8          ; Extension: Sectors per track
10C2  0001                 DW    1          ; Extension: Number of heads
10C4  0000                 DW    0          ; Extension: Number of hidden sectors
10C6                FD360:                  ;
10C6  0200                 DW    512        ; Sector size
10C8  02                   DB    2          ; Sectors per allocation unit
10C9  0001                 DW    1          ; Number of reserved sectors
10CB  02                   DB    2          ; Number of FATs
10CC  0070                 DW    7*16       ; Number of directory entries
10CE  02D0                 DW    2*40*9     ; Total number of sectors
10D0  FD                   DB    0FDH       ; Media byte
10D1  0002                 DW    2          ; Sectors for one FAT
10D3  0009                 DW    9          ; Extension: Sectors per track
10D5  0002                 DW    2          ; Extension: Number of heads
10D7  0000                 DW    0          ; Extension: Number of hidden sectors
10D9                FC180:                  ;
10D9  0200                 DW    512        ; Sector size
10DB  01                   DB    1          ; Sectors per allocation unit
10DC  0001                 DW    1          ; Number of reserved sectors
10DE  02                   DB    2          ; Number of FATs
10DF  0040                 DW    4*16       ; Number of directory entries
10E1  0168                 DW    40*9       ; Total number of sectors
10E3  FC                   DB    0FCH       ; Media byte
10E4  0002                 DW    2          ; Sectors for one FAT
10E6  0009                 DW    9          ; Extension: Sectors per track
10E8  0001                 DW    1          ; Extension: Number of heads
10EA  0000                 DW    0          ; Extension: Number of hidden sectors
10EC                FB720:                  ;
10EC  0200                 DW    512        ; Sector size
10EE  04                   DB    4          ; Sectors per allocation unit
10EF  0001                 DW    1          ; Number of reserved sectors
10F1  02                   DB    2          ; Number of FATs
10F2  0070                 DW    7*16       ; Number of directory entries
10F4  05A0                 DW    2*80*9     ; Total number of sectors
10F6  FB                   DB    0FBH       ; Media byte
10F7  0002                 DW    2          ; Sectors for one FAT
10F9  0009                 DW    9          ; Extension: Sectors per track
10FB  0002                 DW    2          ; Extension: Number of heads
10FD  0000                 DW    0          ; Extension: Number of hidden sectors
10FF                F9800:                  ;
10FF  0400                 DW    1024       ; Sector size
1101  02                   DB    2          ; Sectors per allocation unit
1102  0001                 DW    1          ; Number of reserved sectors
1104  02                   DB    2          ; Number of FATs
1105  00A0                 DW    5*32       ; Number of directory entries
1107  0320                 DW    2*80*5     ; Total number of sectors
1109  F9                   DB    0F9H       ; Media byte
110A  0001                 DW    1          ; Sectors for one FAT
110C  0005                 DW    5          ; Extension: Sectors per track
110E  0002                 DW    2          ; Extension: Number of heads
1110  0000                 DW    0          ; Extension: Number of hidden sectors
```

```
                                 ;************************************************************************
                                 ;************************************************************************
                                 ;************************************************************************
                                 ;
                                 ;
                                 ;   ROUTINE NAME:      DVERIFY
                                 ;
                                 ;
                                 ;   FUNCTION:          WRITE with verify
                                 ;
                                 ;
                                 ;   ENTRY VIA:         CALL
                                 ;
                                 ;
                                 ;   ENTRY CONDITIONS:  Following  variables are set:
                                 ;                      TPI_DR, TPI_DI, BYTSEC, CYLMODE, DRV, HEAD, TRACK,
                                 ;                      SECTOR, SECCNT (Number of sectors), SECTRK,
                                 ;                      and DMAADDR (SEGMENT and OFFSET)
                                 ;
                                 ;   EXIT VIA:          RETURN
                                 ;
                                 ;
                                 ;   EXIT CONDITIONS:   STATUS (returned in ERRBUF)
                                 ;
                                 ;
                                 ;************************************************************************
                                 ;************************************************************************
                                 ;************************************************************************
                                 ;
                                 ;
1112                             DVERIFY:                    ;
1112  8A 26 0009 R                     MOV    AH,HEAD        ; Save PIM.HEAD
1116  A0 000A R                        MOV    AL,TRACK       ;     PIM.TRACK
1119  50                               PUSH   AX             ;
111A  8A 26 000B R                     MOV    AH,SECTOR      ;     PIM.SECTOR
111E  50                               PUSH   AX             ;
111F  A1 000C R                        MOV    AX,SECCNT      ;     PIM.SECCNT variables to READ
1122  50                               PUSH   AX             ;                        after WRITE
                                                             ;
1123  E8 0037 R                        CALL   DWRITE         ; Do low level WRITE DATA
                                                             ;
1126  58                               POP    AX             ;
1127  A3 000C R                        MOV    SECCNT,AX      ; Restore PIM.SECCNT
112A  58                               POP    AX             ;
112B  88 26 000B R                     MOV    SECTOR,AH      ;        PIM.SECTOR
112F  58                               POP    AX             ;
1130  A2 000A R                        MOV    TRACK,AL       ;        PIM.TRACK
1133  88 26 0009 R                     MOV    HEAD,AH        ;        PIM.HEAD variables to READ
                                                             ;                         after WRITE
                                                             ;
1137  F6 06 0018 R C0                  TEST   ERRBUF,0C0H    ; Test for normal termination
113C  74 01                            JZ     DVER0          ; Jump if normal termination
                                                             ; else
113E  C3                               RET                   ; return
```

```
113F                      DVER0:                            ;
                                                            ; Fill PIM variables to READ after WRITE
                                                            ; ---------------------------------------
113F  C7 06 001F R 11AA R          MOV    DMAADDR,OFFSET BYTEBUF ; DMAADDR <-- OFFSET
1145  8C 0E 0021 R                 MOV    DMAADDR+2,CS      ;             <-- SEGMENT
1149  C7 06 0023 R 0001            MOV    DMALENG,0001      ; DMALENG     <-- 1 byte transfer
114F  C6 06 0025 R 47              MOV    DMAFUNC,DMAWRT    ; DMAFUNC     <-- WRITE DMA COMMAND
1154                      DVER1:                            ;
1154  B1 06                        MOV    CL,READDAT        ; CL <-- READ DATA COMMAND
1156  E8 0AE2 R                    CALL   SETUP9            ; Set up COMMAND STRING and DMA
1159  E8 0B70 R                    CALL   XWAIT             ; Send COMMAND STRING to FDC
115C  E8 0B8E R                    CALL   GETBYT            ; Get STATUS BYTES
                                                            ;
115F  F6 06 0018 R C0              TEST   ERRBUF,0C0H       ; Test for normal termination
1164  74 04                        JZ     DVER2             ; Jump if normal termination
                                                            ;
1166  58                           POP    AX                ; Flush return addr from stack
1167  E9 0DF8 R                    JMP    ERROR_10          ; Generate 'WRITE FAULT' error
116A                      DVER2:                            ;
116A  FF 0E 000C R                 DEC    SECCNT            ; Decrement sector counter
116E  75 01                        JNZ    DVER3             ; Jump if another I/O is necessary
1170  C3                           RET                      ; else return if I/O is complete
1171                      DVER3:                            ;
1171  A0 000B R                    MOV    AL,SECTOR         ;
1174  3A 06 0028 R                 CMP    AL,SECTRK         ; Check if next sector fits in track
1178  74 06                        JZ     DVER4             ; Jump if not
                                                            ;
117A  FE 06 000B R                 INC    SECTOR            ; Increment SECTOR (next sector fits in track)
117E  EB D4                        JMP    DVER1             ;
1180                      DVER4:                            ;
1180  C6 06 000B R 01              MOV    SECTOR,1          ; Set SECTOR to begin of track
                                                            ;
1185  80 3E 0007 R 00              CMP    CYLMODE,0         ; Check if double sided disk
118A  74 06                        JZ     DVER5             ; Jump if double sided
                                                            ;
                                                            ; Single sided
                                                            ; ------------
118C  FE 06 000A R                 INC    TRACK             ; Increment TRACK
1190  EB C2                        JMP    DVER1             ;
1192                      DVER5:                            ; Double sided
                                                            ; ------------
1192  80 3E 0009 R 00              CMP    HEAD,0            ;
1197  75 06                        JNZ    DVER6             ; If HEAD 0
1199  FE 06 0009 R                 INC    HEAD              ; then set HEAD 1
119D  EB B5                        JMP    DVER1             ;
119F                      DVER6:                            ; else
119F  C6 06 0009 R 00              MOV    HEAD,0            ; set HEAD 0
11A4  FE 06 000A R                 INC    TRACK             ; and increment TRACK
11A8  EB AA                        JMP    DVER1             ;
                                                            ;
                                                            ;
11AA                      BYTEBUF:                          ;
11AA  ??                           DB     ?                 ; BYTE BUFFER to detect CRC errors
```

```
                               ;*****************************************************************************
                               ;*****************************************************************************
                               ;*****************************************************************************
                               ;
                               ;
                               ;   ROUTINE NAME:       PRDICH
                               ;
                               ;
                               ;   FUNCTION:           PROMPT DISK CHANGE MESSAGE FOR SINGLE DRIVE SYSTEMS
                               ;
                               ;
                               ;   ENTRY VIA:          CALL
                               ;
                               ;
                               ;   ENTRY CONDITIONS:   DRV variable is set
                               ;
                               ;
                               ;   EXIT VIA:           RETURN
                               ;
                               ;
                               ;   EXIT CONDITIONS:    NONE
                               ;
                               ;*****************************************************************************
                               ;*****************************************************************************
                               ;*****************************************************************************
                               ;
                               ;
= 000D                         CR      EQU     0DH             ; Carriage return
= 000A                         LF      EQU     0AH             ; Line feed
= 0024                         MSGEND  EQU     24H             ; Message end
                                                               ;
11AB  0D 0A                    PROMPT1 DB      CR,LF
11AD  49 6E 73 65 72 74                DB      'Insert diskette for drive ',MSGEND
      20 64 69 73 6B 65
      74 74 65 20 66 6F
      72 20 64 72 69 76
      65 20 24

11C8  3A 20 61 6E 64 20        PROMPT2 DB                                      ': and strike',CR,LF
      73 74 72 69 6B 65
      0D 0A
11D6  61 6E 79 20 6B 65                DB      'any key when ready',CR,LF,MSGEND
      79 20 77 68 65 6E
      20 72 65 61 64 79
      0D 0A 24
```

```
11EB                        PRDICH:                            ;
11EB  BE 11AB R                     MOV     SI,OFFSET PROMPT1 ;
11EE  E8 1206 R                     CALL    DISP               ; Display PROMPT1
                                                               ;
11F1  A0 0002 E                     MOV     AL,FL_FLAGS+2      ;
11F4  04 41                         ADD     AL,'A'             ;
11F6  CD 29                         INT     29H                ; Display drive letter
                                                               ;
11F8  BE 11C8 R                     MOV     SI,OFFSET PROMPT2 ;
11FB  E8 1206 R                     CALL    DISP               ; Display PROMPT2
                                                               ;
11FE  E8 0000 E                     CALL    KBD_FL             ; Flush keyboard
                                                               ;
1201  B4 00                         MOV     AH,00              ;
1203  CD 16                         INT     16H                ; Read keyboard
                                                               ;
1205  C3                            RET                        ;
                                                               ;
                                                               ;
1206                        DISP:                              ;
1206  FC                            CLD                        ; Incrementing
1207                        DISP1:                             ;
1207  AC                            LODS    BYTE PTR [SI]      ; Get next character
1208  3C 24                         CMP     AL,MSGEND          ;
120A  74 04                         JZ      DISP2              ; Exit if message end
                                                               ; else
120C  CD 29                         INT     29H                ; display character
120E  EB F7                         JMP     SHORT DISP1        ;
1210                        DISP2:                             ;
1210  C3                            RET                        ;


1211                        DREND:                             ; End of driver

1211                        CSEG    ENDS
                            END
```

```
Structures and records:

                N a m e                 Width   # fields
                                        Shift   Width   Mask    Initial

DPB. . . . . . . . . . . . . . . .      0013    000B
  SECSIZE. . . . . . . . . . . . . .    0000
  ALLOC. . . . . . . . . . . . . . .    0002
  RESSEC . . . . . . . . . . . . . .    0003
  FATS . . . . . . . . . . . . . . .    0005
  MAXDIR . . . . . . . . . . . . . .    0006
  SECTORS. . . . . . . . . . . . . .    0008
  MEDIAID. . . . . . . . . . . . . .    000A
  FATSEC . . . . . . . . . . . . . .    000B
  SECPTR . . . . . . . . . . . . . .    000D
  HEADS. . . . . . . . . . . . . . .    000F
  HIDSEC . . . . . . . . . . . . . .    0011
IODAT. . . . . . . . . . . . . . .      0016    000A
  CMDLEN . . . . . . . . . . . . . .    0000
  UNIT . . . . . . . . . . . . . . .    0001
  CMD. . . . . . . . . . . . . . . .    0002
  STATUS . . . . . . . . . . . . . .    0003
  MEDIA. . . . . . . . . . . . . . .    000D
  TRANS. . . . . . . . . . . . . . .    000E
  COUNT. . . . . . . . . . . . . . .    0012
  START. . . . . . . . . . . . . . .    0014

Segments and groups:

                N a m e                 Size    align   combine class

CSEG . . . . . . . . . . . . . . .      1211    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                 Type    Value   Attr

BANK . . . . . . . . . . . . . . .      Number  00E0
BPB0 . . . . . . . . . . . . . . .      L NEAR  107A    CSEG
BPB1 . . . . . . . . . . . . . . .      L NEAR  108D    CSEG
BPB_SIZE . . . . . . . . . . . . .      Number  0013
BPB_START. . . . . . . . . . . . .      E NEAR  107A    CSEG
BX_SAVE. . . . . . . . . . . . . .      L WORD  0005    CSEG
BYTEBUF. . . . . . . . . . . . . .      L NEAR  11AA    CSEG
BYTSEC . . . . . . . . . . . . . .      L BYTE  002A    CSEG
CMDERR . . . . . . . . . . . . . .      L NEAR  0D38    CSEG
COAD . . . . . . . . . . . . . . .      Number  0026
CONSTR . . . . . . . . . . . . . .      L BYTE  000E    CSEG
CONFIG_FLAGS . . . . . . . . . . .      V WORD  0000    CSEG    External
COTC . . . . . . . . . . . . . . .      Number  0027
CR . . . . . . . . . . . . . . . .      Number  000D
CYLMODE. . . . . . . . . . . . . .      L BYTE  0007    CSEG
DCOMD. . . . . . . . . . . . . . .      Number  0051
DENSITY. . . . . . . . . . . . . .      L BYTE  0029    CSEG
DFORMAT. . . . . . . . . . . . . .      L NEAR  0AC3    CSEG
DISP . . . . . . . . . . . . . . .      L NEAR  1206    CSEG
DISP1. . . . . . . . . . . . . . .      L NEAR  1207    CSEG
DISP2. . . . . . . . . . . . . . .      L NEAR  1210    CSEG
DMA. . . . . . . . . . . . . . . .      L NEAR  0BB7    CSEG
DMAADDR. . . . . . . . . . . . . .      L WORD  001F    CSEG
DMAFUNC. . . . . . . . . . . . . .      L BYTE  0025    CSEG
```

```
DMALENG. . . . . . . . . . . . . .     L WORD  0023   CSEG
DMAMB. . . . . . . . . . . . . . .     Number  002A
DMAMO. . . . . . . . . . . . . . .     Number  002B
DMAREAD. . . . . . . . . . . . . .     Number  004B
DMAWRT . . . . . . . . . . . . . .     Number  0047
DREAD. . . . . . . . . . . . . . .     L NEAR  082E   CSEG
DREADID. . . . . . . . . . . . . .     L NEAR  0A8F   CSEG
DREND. . . . . . . . . . . . . . .     L NEAR  1211   CSEG   Global
DREST. . . . . . . . . . . . . . .     L NEAR  0A2C   CSEG
DREST1 . . . . . . . . . . . . . .     L NEAR  0A2E   CSEG
DREST2 . . . . . . . . . . . . . .     L NEAR  0A42   CSEG
DREST3 . . . . . . . . . . . . . .     L NEAR  0A57   CSEG
DRV. . . . . . . . . . . . . . . .     L BYTE  0008   CSEG
DSEEK. . . . . . . . . . . . . . .     L NEAR  0A58   CSEG
DSEEK1 . . . . . . . . . . . . . .     L NEAR  0A7F   CSEG
DSEEK2 . . . . . . . . . . . . . .     L NEAR  0A85   CSEG
DSIS . . . . . . . . . . . . . . .     L NEAR  0AB2   CSEG
DSKERR . . . . . . . . . . . . . .     L NEAR  0D3C   CSEG
DSKERR0. . . . . . . . . . . . . .     L NEAR  0D56   CSEG
DSKERR1. . . . . . . . . . . . . .     L NEAR  0D5C   CSEG
DSKERR10 . . . . . . . . . . . . .     L NEAR  0DC5   CSEG
DSKERR2. . . . . . . . . . . . . .     L NEAR  0D66   CSEG
DSKERR3. . . . . . . . . . . . . .     L NEAR  0D70   CSEG
DSKERR4. . . . . . . . . . . . . .     L NEAR  0D7A   CSEG
DSKERR5. . . . . . . . . . . . . .     L NEAR  0D84   CSEG
DSKERR6. . . . . . . . . . . . . .     L NEAR  0D95   CSEG
DSKERR7. . . . . . . . . . . . . .     L NEAR  0D9F   CSEG
DSKERR8. . . . . . . . . . . . . .     L NEAR  0DA9   CSEG
DSKERR9. . . . . . . . . . . . . .     L NEAR  0DB7   CSEG
DSKTBL . . . . . . . . . . . . . .     L NEAR  0BF5   CSEG
DSK_INIT . . . . . . . . . . . . .     L NEAR  0E16   CSEG
DSK_INT. . . . . . . . . . . . . .     L NEAR  0C1C   CSEG   Global
DSTAT. . . . . . . . . . . . . . .     Number  0050
DVER0. . . . . . . . . . . . . . .     L NEAR  113F   CSEG
DVER1. . . . . . . . . . . . . . .     L NEAR  1154   CSEG
DVER2. . . . . . . . . . . . . . .     L NEAR  116A   CSEG
DVER3. . . . . . . . . . . . . . .     L NEAR  1171   CSEG
DVER4. . . . . . . . . . . . . . .     L NEAR  1180   CSEG
DVER5. . . . . . . . . . . . . . .     L NEAR  1192   CSEG
DVER6. . . . . . . . . . . . . . .     L NEAR  119F   CSEG
DVERIFY. . . . . . . . . . . . . .     L NEAR  1112   CSEG
DWRITE . . . . . . . . . . . . . .     L NEAR  0837   CSEG
EC . . . . . . . . . . . . . . . .     L NEAR  0E9A   CSEG
ERRBUF . . . . . . . . . . . . . .     L BYTE  0018   CSEG
ERROR_0. . . . . . . . . . . . . .     L NEAR  0DC8   CSEG
ERROR_1. . . . . . . . . . . . . .     L NEAR  0DCC   CSEG
ERROR_10 . . . . . . . . . . . . .     L NEAR  0DF0   CSEG
ERROR_11 . . . . . . . . . . . . .     L NEAR  0DF4   CSEG
ERROR_12 . . . . . . . . . . . . .     L NEAR  0DF8   CSEG
ERROR_2. . . . . . . . . . . . . .     L NEAR  0DD0   CSEG
ERROR_3. . . . . . . . . . . . . .     L NEAR  0DD4   CSEG
ERROR_4. . . . . . . . . . . . . .     L NEAR  0DD8   CSEG
ERROR_5. . . . . . . . . . . . . .     L NEAR  0DDC   CSEG
ERROR_6. . . . . . . . . . . . . .     L NEAR  0DE0   CSEG
ERROR_7. . . . . . . . . . . . . .     L NEAR  0DE4   CSEG
ERROR_8. . . . . . . . . . . . . .     L NEAR  0DE8   CSEG
ERROR_9. . . . . . . . . . . . . .     L NEAR  0DEC   CSEG
ERR_EXIT . . . . . . . . . . . . .     L NEAR  0DFA   CSEG
ES_SAVE. . . . . . . . . . . . . .     L WORD  0003   CSEG
EXIT . . . . . . . . . . . . . . .     L NEAR  0DFF   CSEG
EXIT1. . . . . . . . . . . . . . .     L NEAR  0E01   CSEG
```

```
EXITP. . . . . . . . . . . . . . . .    F PROC  0DFF    CSEG    Length =0017
F9000. . . . . . . . . . . . . . . .    L NEAR  10FF    CSEG
FB720. . . . . . . . . . . . . . . .    L NEAR  10EC    CSEG
FC180. . . . . . . . . . . . . . . .    L NEAR  10D9    CSEG
FD360. . . . . . . . . . . . . . . .    L NEAR  10C6    CSEG
FDCRA. . . . . . . . . . . . . . . .    Number  0051
FDCRDY . . . . . . . . . . . . . . .    L NEAR  0B9E    CSEG
FDCSIS . . . . . . . . . . . . . . .    Number  0008
FE160. . . . . . . . . . . . . . . .    L NEAR  10B3    CSEG
FF320. . . . . . . . . . . . . . . .    L NEAR  10A0    CSEG
FLOPPY_DRIVES. . . . . . . . . . . .    V BYTE  0000    CSEG    External
FLTAB. . . . . . . . . . . . . . . .    V WORD  0000    CSEG    External
FL_FLAGS . . . . . . . . . . . . . .    V BYTE  0000    CSEG    External
FL_OUT_RETRIES . . . . . . . . . . .    V BYTE  0000    CSEG    External
GETBPB1. . . . . . . . . . . . . . .    L NEAR  0EDE    CSEG
GETBPB10 . . . . . . . . . . . . . .    L NEAR  0FF7    CSEG
GETBPB11 . . . . . . . . . . . . . .    L NEAR  100D    CSEG
GETBPB12 . . . . . . . . . . . . . .    L NEAR  1052    CSEG
GETBPB13 . . . . . . . . . . . . . .    L NEAR  1073    CSEG
GETBPB2. . . . . . . . . . . . . . .    L NEAR  0F25    CSEG
GETBPB3. . . . . . . . . . . . . . .    L NEAR  0F2F    CSEG
GETBPB4. . . . . . . . . . . . . . .    L NEAR  0F39    CSEG
GETBPB5. . . . . . . . . . . . . . .    L NEAR  0F3E    CSEG
GETBPB6. . . . . . . . . . . . . . .    L NEAR  0F4A    CSEG
GETBPB7. . . . . . . . . . . . . . .    L NEAR  0F58    CSEG
GETBPB8. . . . . . . . . . . . . . .    L NEAR  0F9C    CSEG
GETBPB9. . . . . . . . . . . . . . .    L NEAR  0FD5    CSEG
GETBYT . . . . . . . . . . . . . . .    L NEAR  0B8E    CSEG
GETBYT1. . . . . . . . . . . . . . .    L NEAR  0B91    CSEG
GET_BPB. . . . . . . . . . . . . . .    L NEAR  0E9B    CSEG
GPL. . . . . . . . . . . . . . . . .    L BYTE  002B    CSEG
HEAD . . . . . . . . . . . . . . . .    L BYTE  0009    CSEG
IDREAD . . . . . . . . . . . . . . .    Number  000A
INITTAB. . . . . . . . . . . . . . .    L NEAR  1076    CSEG
INT0 . . . . . . . . . . . . . . . .    L NEAR  0C64    CSEG
INT1 . . . . . . . . . . . . . . . .    L NEAR  0C72    CSEG
INT2 . . . . . . . . . . . . . . . .    L NEAR  0C98    CSEG
INT3 . . . . . . . . . . . . . . . .    L NEAR  0CA1    CSEG
INT4 . . . . . . . . . . . . . . . .    L NEAR  0CAE    CSEG
INT5 . . . . . . . . . . . . . . . .    L NEAR  0CD3    CSEG
INT6 . . . . . . . . . . . . . . . .    L NEAR  0D05    CSEG
INT7 . . . . . . . . . . . . . . . .    L NEAR  0D1B    CSEG
INT8 . . . . . . . . . . . . . . . .    L NEAR  0D2C    CSEG
INT9 . . . . . . . . . . . . . . . .    L NEAR  0D35    CSEG
IO . . . . . . . . . . . . . . . . .    L NEAR  0967    CSEG
IO0. . . . . . . . . . . . . . . . .    L NEAR  085A    CSEG
IO1. . . . . . . . . . . . . . . . .    L NEAR  086E    CSEG
IO10 . . . . . . . . . . . . . . . .    L NEAR  091D    CSEG
IO11 . . . . . . . . . . . . . . . .    L NEAR  0941    CSEG
IO15 . . . . . . . . . . . . . . . .    L NEAR  0947    CSEG
IO17 . . . . . . . . . . . . . . . .    L NEAR  0963    CSEG
IO2. . . . . . . . . . . . . . . . .    L NEAR  0876    CSEG
IO20 . . . . . . . . . . . . . . . .    L NEAR  096A    CSEG
IO21 . . . . . . . . . . . . . . . .    L NEAR  097E    CSEG
IO22 . . . . . . . . . . . . . . . .    L NEAR  0987    CSEG
IO23 . . . . . . . . . . . . . . . .    L NEAR  0990    CSEG
IO24 . . . . . . . . . . . . . . . .    L NEAR  0999    CSEG
IO25 . . . . . . . . . . . . . . . .    L NEAR  09A4    CSEG
IO3. . . . . . . . . . . . . . . . .    L NEAR  088F    CSEG
IO30 . . . . . . . . . . . . . . . .    L NEAR  09A6    CSEG
IO31 . . . . . . . . . . . . . . . .    L NEAR  09AE    CSEG
```

```
I032 . . . . . . . . . . . . . . .      L NEAR  09D5    CSEG
I033 . . . . . . . . . . . . . . .      L NEAR  09E8    CSEG
I034 . . . . . . . . . . . . . . .      L NEAR  09F5    CSEG
I035 . . . . . . . . . . . . . . .      L NEAR  09FE    CSEG
I036 . . . . . . . . . . . . . . .      L NEAR  0A1B    CSEG
I04. . . . . . . . . . . . . . . .      L NEAR  0899    CSEG
I06. . . . . . . . . . . . . . . .      L NEAR  08C3    CSEG
I07. . . . . . . . . . . . . . . .      L NEAR  08CC    CSEG
I09. . . . . . . . . . . . . . . .      L NEAR  0902    CSEG
KBD_FL . . . . . . . . . . . . . .      L NEAR  0000    CSEG    External
LF . . . . . . . . . . . . . . . .      Number  000A
MEDIAC . . . . . . . . . . . . . .      L NEAR  0E64    CSEG
MEDIAC1. . . . . . . . . . . . . .      L NEAR  0E71    CSEG
MEDIAC2. . . . . . . . . . . . . .      L NEAR  0E77    CSEG
MEDIAC3. . . . . . . . . . . . . .      L NEAR  0E94    CSEG
MOTORCK. . . . . . . . . . . . . .      L NEAR  0BA5    CSEG
MOTORCK1 . . . . . . . . . . . . .      L NEAR  0BAE    CSEG
MOTORCK2 . . . . . . . . . . . . .      L NEAR  0BB1    CSEG
MOTORON. . . . . . . . . . . . . .      Number  0014
MSGEND . . . . . . . . . . . . . .      Number  0024
PATTERN. . . . . . . . . . . . . .      L BYTE  002C    CSEG
PRDICH . . . . . . . . . . . . . .      L NEAR  11EB    CSEG
PROMPT1. . . . . . . . . . . . . .      L BYTE  11AB    CSEG
PROMPT2. . . . . . . . . . . . . .      L BYTE  11CB    CSEG
PTRSAV . . . . . . . . . . . . . .      V DWORD 0000    CSEG    External
READDAT. . . . . . . . . . . . . .      Number  0006
READTRK. . . . . . . . . . . . . .      Number  0002
RESTORE. . . . . . . . . . . . . .      Number  0007
RETRIES. . . . . . . . . . . . . .      L BYTE  002D    CSEG
SECCNT . . . . . . . . . . . . . .      L WORD  000C    CSEG
SECTOR . . . . . . . . . . . . . .      L BYTE  000B    CSEG
SECTRK . . . . . . . . . . . . . .      L BYTE  002B    CSEG
SEEKTRK. . . . . . . . . . . . . .      Number  000F
SET1 . . . . . . . . . . . . . . .      L NEAR  0AF8    CSEG
SETUP6 . . . . . . . . . . . . . .      L NEAR  0B37    CSEG
SETUP9 . . . . . . . . . . . . . .      L NEAR  0AE2    CSEG
SPECIFY. . . . . . . . . . . . . .      Number  0003
SSB. . . . . . . . . . . . . . . .      L WORD  002E    CSEG
SYSSTA . . . . . . . . . . . . . .      Number  0013
TPI_DI . . . . . . . . . . . . . .      L BYTE  0027    CSEG
TPI_DR . . . . . . . . . . . . . .      L BYTE  0026    CSEG
TRACK. . . . . . . . . . . . . . .      L BYTE  000A    CSEG
WRITDAT. . . . . . . . . . . . . .      Number  0005
WRITFMT. . . . . . . . . . . . . .      Number  000D
WT480N96 . . . . . . . . . . . . .      Number  0002
XWAIT. . . . . . . . . . . . . . .      L NEAR  0B70    CSEG
XWAIT1 . . . . . . . . . . . . . .      L NEAR  0B7A    CSEG
```

```
                            ;*********************************************************
                            ;*                                                       *
                            ;*        W I N C H E S T E R   D I S K                 *
                            ;*                                                       *
                            ;*                D R I V E R                            *
                            ;*                                                       *
                            ;*********************************************************
                            ;
                            ;
                            ;
                            ;     DEFINE OFFSETS FOR IO DATA PACKET
                            ;
= 0000                      CMDLEN  EQU     0               ;LENGTH OF THIS BLOCK
= 0001                      UNIT    EQU     1               ;SUB UNIT SPECIFIER
= 0002                      CMD     EQU     2               ;COMMAND CODE
= 0003                      STATUS  EQU     3               ;STATUS
= 000D                      MEDIA   EQU     13              ;MEDIA DESCRIPTOR
= 000E                      TRANS   EQU     14              ;TRANSFER ADDRESS
= 0012                      COUNT   EQU     18              ;COUNT OF SECTORS
= 0014                      START   EQU     20              ;FIRST BLOCK TO TRANSFER
                            ;
                            ;
                            ;
                            ;          WINCHESTER DISK DEFINITIONS
                            ;          ***************************
                            ;
                            ;*****************************************************
                            ;*                                                   *
                            ;*      PORT DEFINITIONS                             *
                            ;*                                                   *
                            ;*****************************************************
                            ;
= 00C0                      HBASE   EQU     0C0H            ;     CONTROLLER BASE ADDR.
=                           DATA    EQU     HBASE           ; R/W DATA REGISTER
= 00C1                      ERROR   EQU     HBASE+1         ; R   ERROR REGISTER
= 00C1                      WPC     EQU     HBASE+1         ;   W WRITE PRECOMP. REGISTER
= 00C2                      SECNT   EQU     HBASE+2         ; R/W SECTOR COUNT REGISTER
= 00C3                      SECNO   EQU     HBASE+3         ; R/W SECTOR NUMBER REGISTER
= 00C4                      CYLLO   EQU     HBASE+4         ; R/W CYLINDER LOW REGISTER
= 00C5                      CYLHI   EQU     HBASE+5         ; R/W CYLINDER HIGH REGISTER
= 00C6                      SDH     EQU     HBASE+6         ; R/W ECC/CRC-BYTES PER SECTOR-DRIVE-HEAD
= 00C7                      STAT    EQU     HBASE+7         ; R   STATUS REGISTER
= 00C7                      COMND   EQU     HBASE+7         ;   W COMMAND REGISTER
                            ;
                            ;
                            ;*****************************************************
                            ;*                                                   *
                            ;*      DISK FUNCTIONS                               *
                            ;*                                                   *
                            ;*****************************************************
                            ;
= 0000                      STRATE  EQU     0               ;STEPPING RATE TRACK TO TRACK = BUFFERED STEP
= 0010                      REST    EQU     10H OR STRATE   ;RESTORE COMMAND WITH STRATE
= 0070                      SEEK    EQU     70H OR STRATE   ;SEEK COMMAND WITH STRATE
= 0020                      READ    EQU     20H             ;READ COMMAND
= 0030                      WRITE   EQU     30H             ;WRITE COMMAND
= 0050                      FORMAT  EQU     50H             ;FORMAT COMMAND
```

```
                        ;
                        ;**********************************************
                        ;*                                            *
                        ;*      ERRROR  REGISTER  EQUATES             *
                        ;*                                            *
                        ;**********************************************
                        ;
 = 0001                 DAMNFD  EQU     01H             ; ADDR. MARK NOT FOUND
 = 0002                 TR0     EQU     02H             ; TRACK 0 ERROR
 = 0004                 ABC     EQU     04H             ; ABORTED COMMAND
 = 0010                 IDNFD   EQU     10H             ; ID NOT FOUND
 = 0020                 CRCID   EQU     20H             ; CRC-ERROR  ID-FIELD
 = 0040                 UNCOR   EQU     40H             ; UNCORRECTED DATA IN DATA FIELD
 = 0080                 BBD     EQU     80H             ; BAD BLOCK DETECTED
                        ;
                        ;**********************************************
                        ;*                                            *
                        ;*      STATUS REGISTER EQUATES               *
                        ;*                                            *
                        ;**********************************************
                        ;
 = 0001                 CERR    EQU     01H             ; CONTROLLER ERROR
 = 0004                 CORRD   EQU     04H             ; DATA CORRECTED IN DATA FIELD (ECC)
 = 0008                 CDRQ    EQU     08H             ; CONTROLLER DATA REQUEST
 = 0010                 DSEEC   EQU     10H             ; DRIVE SEEK COMPLETE
 = 0020                 DWRFA   EQU     20H             ; DRIVE WRITE FAULT
 = 0040                 DREADY  EQU     40H             ; DRIVE READY
 = 0080                 CBUSY   EQU     80H             ; CONTROLLER BUSY
                        ;
                        ;
                        ;**********************************************
                        ;*                                            *
                        ;*      SPECIALS                              *
                        ;*                                            *
                        ;**********************************************
                        ;
 = 00A0                 SDHREG  EQU     0A0H            ;ECC/512 BYTES PER SECTOR
                        ;
                        ;
                        ;
                        PUBLIC WI_STRATEGY              ;STRATEGY ENTRY POINT
                        PUBLIC WI_INTERRUPT             ;INTERRUPT ENTRY POINT
                        PUBLIC WI_START                 ;BEGIN OF DRIVER
                        PUBLIC WI_END                   ;END OF DRIVER
                        ;
                        ;
                        EXTRN WINDEV:WORD
                        EXTRN WINCH_DRIVES:BYTE         ;NO. OF WINCHESTER DRIVES
                        EXTRN WI_OUT_RETRIES:BYTE       ;NO. OF RETRIES  (OUT)
                        EXTRN WI_IN_RETRIES:BYTE        ;NO. OF RETRIES  (IN)
                        EXTRN WI_FLAGS:BYTE
                        ;
                        ;
```

```
0000                    CSEG    SEGMENT PUBLIC  'CODE'
                        ASSUME  CS:CSEG,DS:CSEG,ES:CSEG,SS:CSEG

0000                            ORG     0
                        ;
0000                    BEGIN:
                        ;
                        ;*****************************************************
                        ;*                                                   *
                        ;*   S P E C I A L   D E V I C E   H E A D E R       *
                        ;*                                                   *
                        ;*****************************************************
                        ;
                        ;---------------------------------------------+
                        ;     DWORD pointer to next device            ; 1 word offset.
                        ;        (-1,-1 if last device)               ; 1 word segement.
                        ;---------------------------------------------+
                        ;     Device attribute WORD                   ; 1 word.
                        ;       Bit 15 = 1 for character devices.     ;
                        ;                0 for block devices.         ;
                        ;                                             ;
                        ;       Charcter devices. (Bit 15=1)          ;
                        ;         Bit 0 = 1  current sti device.      ;
                        ;         Bit 1 = 1  current sto device.      ;
                        ;         Bit 2 = 1  current NUL device.      ;
                        ;         Bit 3 = 1  current Clock device.    ;
                        ;                                             ;
                        ;         Bit 14 = 1 IOCTL control bit.       ;
                        ;---------------------------------------------+
                        ;     Device strategy pointer.                ; 1 word offset.
                        ;---------------------------------------------+
                        ;     Device interrupt pointer.               ; 1 word offset.
                        ;---------------------------------------------+
                        ;     Device name field.                      ; 8 bytes.
                        ;       Character devices are any valid name  ;
                        ;         left justified, in a space filled   ;
                        ;         field.                              ;
                        ;       Block devices contain # of units in   ;
                        ;         the first byte.                     ;
                        ;---------------------------------------------+
                        ;
                        ;
0000                    WI_START:
                        ;
                        ;       RELEASE ID
0000  02                        DB      02              ;ISSUE
0001  02                        DB      02              ;SUB ISSUE
0002  00                        DB      00              ;PATCH LEVEL
```

```
                             ;
                             ;  SIMPLISTIC STRATEGY ROUTINE FOR NON-MULTI-TASKING SYSTEM
                             ;
                             ;      CURRENTLY JUST SAVES I/O PACKET POINTER IN PTRSAV
                             ;      FOR LATER PROCESSING BY THE INTERRUPT ROUTINE.
                             ;
0003                         STRATP  PROC    FAR
                             ;
0003                         WI_STRATEGY:
0003  2E: 89 1E 000E R               MOV     CS:WORD PTR PTRSAV,BX
0008  2E: 8C 06 0010 R               MOV     CS:WORD PTR PTRSAV+2,ES
000D  CB                             RET
                             ;
000E                         STRATP  ENDP
                             ;
000E  0000 0000              PTRSAV  DW      0,0             ;STRATEGY POINTER SAVE
                             ;
                             ;
0012                         WI_INTERRUPT:
0012  56                             PUSH    SI
0013  BE 006F R                      MOV     SI,OFFSET WITBL
                             ;
0016  50                     ENTRY:  PUSH    AX                      ;SAVE ALL NECESSARY REGISTERS.
0017  51                             PUSH    CX
0018  52                             PUSH    DX
0019  57                             PUSH    DI
001A  1E                             PUSH    DS
001B  06                             PUSH    ES
001C  53                             PUSH    BX
001D  0E                             PUSH    CS
001E  1F                             POP     DS                      ;SET DATA SEG. TO CODE SEG.
001F  8B 1E 000E R                   MOV     BX,WORD PTR PTRSAV      ;Retrieve pointer to I/O Packet.
0023  8E 06 0010 R                   MOV     ES,WORD PTR PTRSAV+2
0027  26: 8A 47 01                   MOV     AL,ES:UNIT[BX]          ;AL = Unit code.
002B  A2 0379 R                      MOV     BYTE PTR WIPAR,AL
002E  26: 8A 67 0D                   MOV     AH,ES:MEDIA[BX]         ;AH = Media descriptor.
0032  26: 80 7F 02 00                CMP     ES:BYTE PTR CMD[BX],0
0037  74 05                          JZ      ENTRY1                  ;SKIP MEDIA CHECK IF INIT FUNCT.
0039  80 FC F8                       CMP     AH,0F8H
003C  72 2E                          JB      WI_INTERRUPT1           ;Unknown Media descriptor
003E                          ENTRY1:
003E  26: 8B 4F 12                   MOV     CX,ES:COUNT[BX]         ;CX = Contains byte/sector count.
0042  26: 8B 57 14                   MOV     DX,ES:START[BX]         ;DX = Starting Logical sector.
0046  89 16 037B R                   MOV     WORD PTR WIPAR+2,DX
004A  97                             XCHG    DI,AX                   ;Move Unit/Media into DI temporarily
004B  26: 8A 47 02                   MOV     AL,ES:CMD[BX]           ;Retrieve Command type.
004F  32 E4                          XOR     AH,AH                   ;Clear upper half of AX for calc.
0051  03 F0                          ADD     SI,AX                   ;Comp. entry point in funct. table.
0053  03 F0                          ADD     SI,AX
0055  3C 0B                          CMP     AL,11                   ;Not more than 11 commands.
0057  77 6E                          JA      CMDERR                  ;Ah, well, error out.
0059  97                             XCHG    AX,DI                   ;Move Unit & Media back.
005A  26: 8B 7F 10                   MOV     DI,ES:TRANS+2 [BX]      ;DI = addess of Transfer address.
005E  89 3E 037F R                   MOV     WORD PTR WIPAR+6,DI     ;Buffer Addr. to PIM table
0062  26: 8B 7F 0E                   MOV     DI,ES:TRANS [BX]
0066  89 3E 0381 R                   MOV     WORD PTR WIPAR+8,DI
006A  FF 24                          JMP     WORD PTR[SI]            ;Perform I/O packet command.
```

```
006C                    WI_INTERRUPT1:
006C  EB 55 90                 JMP     MEDERR              ;UNKNOWN MEDIA DESCRIPTOR
                        ;
                        ;
                        ;
                        ;    WINCHESTER DISK FUNCTION TABLE
                        ;
006F  0453 R            WITBL  DW      WIINIT        ;0  - Initialize Driver.
0071  00F1 R                   DW      MEDIAC        ;1  - Return current media code.
0073  0116 R                   DW      GETBPB        ;2  - Get Bios Parameter Block.
0075  00C7 R                   DW      CMDERR        ;3  - Reserved. (currently returns error)
0077  01A4 R                   DW      WIREAD        ;4  - Block read.
0079  00B7 R                   DW      BUSEXIT       ;5  - (Not used, return busy flag)
007B  00CE R                   DW      EXIT          ;6  - Return status. (Not used)
007D  00CE R                   DW      EXIT          ;7  - Flush input buffer. (Not used.)
007F  0276 R                   DW      WIWRT         ;8  - Block write.
0081  0360 R                   DW      WIWRTV        ;9  - Block write with verify.
0083  00CE R                   DW      EXIT          ;10 - Return output status.
0085  00CE R                   DW      EXIT          ;11 - Flush output buffer. (Not used.)
                        ;
                        ;
                        ;     BIOS PARAMETER BLOCK ARRAY
                        ;
0087  0097 R            WIBPB  DW      WIBPB1C
0089  0097 R                   DW      WIBPB1C
008B  0097 R                   DW      WIBPB1C
008D  0097 R                   DW      WIBPB1C
008F  0097 R                   DW      WIBPB1C
0091  0097 R                   DW      WIBPB1C
0093  0097 R                   DW      WIBPB1C
0095  0097 R                   DW      WIBPB1C
                        ;
                        ;     BIOS PARAMETER BLOCK (BPB)
                        ;
                        ;
                        ;     SEAGATE
                        ;
0097  0200              WIBPB1C DW     512           ;BYTES PER SECTOR
0099  10                       DB      16            ;SECTOR PER ALLOCATION UNIT
009A  0001                     DW      1             ;RESERVED SECTORS
009C  01                       DB      1             ;NUMBER OF FAT'S
009D  01F0                     DW      496           ;NUMBER OF ROOT DIRECT. ENTRIES
009F  2882                     DW      10370         ;NUMBER OF SECTORS PER DISK
00A1  F9                       DB      0F9H          ;MEDIA DESCRIPTOR
00A2  0002                     DW      2             ;NUMBER OF FAT SECTORS
00A4  0011                     DW      17            ;SECTORS PER TRACK
00A6  0002                     DW      2             ;NUMBER OF HEADS
00A8  0000                     DW      0             ;HIDDEN SECTORS
                        ;
                        ;
                        ;     SEAGATE WITHOUT BOOT RECORD
                        ;
00AA  0200              WIBPB1A DW     512           ;BYTES PER SECTOR
00AC  10                       DB      16            ;SECTOR PER ALLOCATION UNIT
00AD  0000                     DW      0             ;RESERVED SECTORS
00AF  01                       DB      1             ;NUMBER OF FAT'S
00B0  0200                     DW      512           ;NUMBER OF ROOT DIRECT. ENTRIES
00B2  2882                     DW      10370         ;NUMBER OF SECTORS PER DISK
00B4  FA                       DB      0FAH          ;MEDIA DESCRIPTOR
00B5  0002                     DW      2             ;NUMBER OF FAT SECTORS
```

```
                                ;
                                ; COMMON ERROR PROCESSING ROUTINE.
                                ;  AL = ERROR CODE.
                                ;
                                ;  Error # 0 = Write Protect violation.
                                ;          1 = Unknown unit.
                                ;          2 = Drive not ready.
                                ;          3 = Unknown command in I/O packet.
                                ;          4 = CRC error.
                                ;          5 = Bad drive request structure length.
                                ;          6 = Seek error.
                                ;          7 = Unknown media discovered.
                                ;          8 = Sector not found.
                                ;          9 = Printer out of paper.
                                ;         10 = Write fault.
                                ;         11 = Read fault.
                                ;         12 = General failure.
                                ;
                                ;
00B7                            BUSEXIT:                           ;Device busy exit.
00B7  B4 03                            MOV    AH,00000011B         ;Set busy and done bits.
00B9  EB 15                            JMP    SHORT   EXIT1
00BB                            UNITERR:
00BB  B0 01                            MOV    AL,1
00BD  EB 0A                            JMP    SHORT  ERREXIT
00BF  B0 05                     LENERR: MOV   AL,5                 ;Bad drive request struct.length
00C1  EB 06                            JMP    SHORT  ERREXIT
00C3  B0 07                     MEDERR: MOV   AL,7                 ;Unknown Media discovered.
00C5  EB 02                            JMP    SHORT  ERREXIT
00C7  B0 03                     CMDERR: MOV   AL,3                 ;Set unknown command error #.
00C9                            ERREXIT:
00C9  B4 81                            MOV    AH,10000001B         ;Set error and done bits.
00CB  F9                               STC                         ;Set Carry bit.
00CC  EB 02                            JMP    SHORT   EXIT1        ;Quick way out.
                                ;
00CE                            EXITP  PROC   FAR
                                ;
00CE  B4 01                     EXIT:  MOV    AH,00000001B         ;Set done bit for MSDOS.
00D0  8B 1E 000E R              EXIT1: MOV    BX,WORD PTR PTRSAV
00D4  8E 06 0010 R                     MOV    ES,WORD PTR PTRSAV+2
00D8  26: 89 47 03                     MOV    ES:STATUS[BX],AX     ;Save operation complete and status.
00DC  5B                               POP    BX                   ;Restore registers.
00DD  07                               POP    ES
00DE  1F                               POP    DS
00DF  5F                               POP    DI
00E0  5A                               POP    DX
00E1  59                               POP    CX
00E2  58                               POP    AX
00E3  5E                               POP    SI
00E4  CB                               RET                         ;RESTORE REGS AND RETURN
                                ;
00E5                            EXITP  ENDP
                                ;
                                ;
                                ;    MOVE LENGTH OF DRIVE REQU. STRUCT. TO AL REG.
                                ;
00E5  8B 1E 000E R              GETLEN: MOV   BX,WORD PTR PTRSAV
00E9  8E 06 0010 R                      MOV   ES,WORD PTR PTRSAV+2
00ED  26: 8A 07                         MOV   AL,ES:BYTE PTR CMDLEN[BX]
00F0  C3                                RET
```

```
                            ;
00F1                        MEDIAC:
00F1  E8 00E5 R                     CALL    GETLEN                  ;GET DRIVE STRUCT. LENGTH
00F4  3C 0F                         CMP     AL,15
00F6  72 C7                         JB      LENERR                  ;BAD STRUCTURE LENGTH
00F8  8A 26 0000 E                  MOV     AH,WI_FLAGS             ;GET MEDIA CHECK FLAG
00FC  8A 0E 0379 R                  MOV     CL,BYTE PTR WIPAR       ;GET DISK UNIT
0100  B0 01                         MOV     AL,01H
0102  D2 E0                         SAL     AL,CL                   ;SHIFT AL BECAUSE CL=0
0104  22 C4                         AND     AL,AH
0106  75 07                         JNZ     MEDIAC1
0108  26: C6 47 0E 01               MOV     ES:BYTE PTR TRANS[BX],1 ;SET MEDIA NOT CHANGED
010D  EB BF                         JMP     EXIT
010F                        MEDIAC1:
010F  26: C6 47 0E 00               MOV     ES:BYTE PTR TRANS[BX],0 ;DON'T KNOW IF MEDIA HAS BEEN CHANGED
0114  EB B8                         JMP     EXIT
                            ;
0116                        GETBPB:
0116  E8 00E5 R                     CALL    GETLEN                  ;GET DRIVE STRUCT. LENGTH
0119  3C 16                         CMP     AL,22
011B  72 51                         JB      GETBPB2                 ;BAD STRUCT. LENGTH
011D  E8 0384 R                     CALL    FIXREADY                ;CHECK IF DRIVE READY
0120  75 4F                         JNZ     GETBPB3
0122  C6 06 037A R 10               MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
0127  C7 06 037B R 0000             MOV     WORD PTR WIPAR+2,0      ;SET FAT SECTOR
012D  E8 0397 R                     CALL    FIXDR
0130  80 3E 037D R 50               CMP     BYTE PTR WIPAR+4,50H    ;CHECK IF READY AND SEEK COMPLETE
0135  75 3A                         JNZ     GETBPB3
0137  06                            PUSH    ES
0138  8A 0E 0379 R                  MOV     CL,BYTE PTR WIPAR
013C  B0 01                         MOV     AL,01H
013E  D2 E0                         SAL     AL,CL
0140  50                            PUSH    AX
0141  C6 06 037A R 20               MOV     BYTE PTR WIPAR+1,READ   ;SET READ FUNCTION
0146  E8 0397 R                     CALL    FIXDR
0149  8E 06 037F R                  MOV     ES,WORD PTR WIPAR+6
014D  8B 3E 0381 R                  MOV     DI,WORD PTR WIPAR+8
0151  26: 80 3D FA                  CMP     BYTE PTR ES:[DI],0FAH
0155  74 1F                         JZ      GETBPB5
0157  26: 80 7D 13 00               CMP     BYTE PTR ES:19[DI],0    ;CHECK BPB  (MAX. SECTORS PER DISK)
015C  74 07                         JZ      GETBPB1
015E  26: 80 7D 15 F9               CMP     BYTE PTR ES:21[DI],0F9H ;CHECK BPB  (MEDIA DESCRIPTOR)
0163  74 17                         JZ      GETBPB6
0165                        GETBPB1:
0165  58                            POP     AX
0166  08 06 0000 E                  OR      WI_FLAGS,AL             ;SET UNKNOWN MEDIA FLAG
016A  07                            POP     ES
016B  E9 00C3 R                     JMP     MEDERR                  ;MEDIA ERROR
016E                        GETBPB2:
016E  E9 008F R                     JMP     LENERR
0171                        GETBPB3:
0171  B0 02                         MOV     AL,2                    ;DRIVE NOT READY
0173  E9 00C9 R                     JMP     ERREXIT
0176                        GETBPB5:
0176  BE 00AA R                     MOV     SI,OFFSET WIBPB1A       ;BPB ADDR. OLDSTYLE SEAGATE
0179  EB 16 90                      JMP     GETBPB8
017C                        GETBPB6:
017C  B9 0013                       MOV     CX,19
017F  83 C7 0B                      ADD     DI,11                   ;ADDR. OF BPB IN BOOT RECORD
0182  BE 0097 R                     MOV     SI,OFFSET WIBPB1C
```

```
0185                    GETBPB7:
0185  26: 8A 05                 MOV     AL,BYTE PTR ES:[DI]     ;MOVE BPB FROM BOOT RECORD
0188  88 04                     MOV     BYTE PTR [SI],AL
018A  47                        INC     DI
018B  46                        INC     SI
018C  E2 F7                     LOOP    GETBPB7
018E  BE 0097 R                 MOV     SI,OFFSET WIBPB1C
0191                    GETBPB8:
0191  58                        POP     AX
0192  F6 D0                     NOT     AL                      ;MEDIA FLAG = 0
0194  20 06 0000 E              AND     WI_FLAGS,AL             ;SET MEDIA FLAG
0198  07                        POP     ES
0199  26: 89 77 12              MOV     WORD PTR ES:COUNT [BX],SI
019D  26: 8C 4F 14              MOV     WORD PTR ES:COUNT+2[BX],CS
01A1  E9 00CE R                 JMP     EXIT
                        ;
                        ;
                        ;       READ DATA
                        ;
01A4                    WIREAD:
01A4  83 F9 00                  CMP     CX,0                    ;CHECK IF READ SEKTOR > 0
01A7  74 3B                     JZ      WIREAD1A
01A9  E8 00E5 R                 CALL    GETLEN                  ;GET DRIVE REQU.STRUCT.LENGTH
01AC  3C 16                     CMP     AL,22
01AE  72 BE                     JB      GETBPB2                 ;BAD STRUCT. LENGTH
01B0  E8 0368 R                 CALL    CHKDRIVE                ;CHECK IF UNIT AVAILABLE
01B3  72 5F                     JC      WIREAD4                 ;UNIT ERROR
01B5  E8 0384 R                 CALL    FIXREADY                ;CHECK IF DRIVE READY
01B8  75 4D                     JNZ     WIREAD3
01BA  A1 0271 R                 MOV     AX,WORD PTR RETRYDEF    ;GET NO. OF RETRIES
01BD  A3 0273 R                 MOV     WORD PTR RETRYC,AX
01C0                    WIREAD1:
01C0  C6 06 037A R 20           MOV     BYTE PTR WIPAR+1,READ   ;SET READ FUNCTION
01C5  E8 0397 R                 CALL    FIXDR                   ;READ ONE SECTOR
01C8  A0 037D R                 MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
01CB  24 F9                     AND     AL,0F9H
01CD  3C 50                     CMP     AL,50H
01CF  75 16                     JNZ     WIREAD2                 ;GO PERFORM RETRIES
01D1  A1 0271 R                 MOV     AX,WORD PTR RETRYDEF
01D4  A3 0273 R                 MOV     WORD PTR RETRYC,AX      ;SET RETRY DEFAULT VALUE
01D7  81 06 0381 R 0200         ADD     WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
01DD  FF 06 037B R              INC     WORD PTR WIPAR+2        ;SECTOR # +1
01E1  49                        DEC     CX                      ;SECTOR COUNT -1
01E2  75 DC                     JNZ     WIREAD1                 ;GO READ NEXT SECTOR
01E4                    WIREAD1A:
01E4  E9 00CE R                 JMP     EXIT
                        ;
01E7                    WIREAD2:
01E7  24 80                     AND     AL,CBUSY
01E9  75 1C                     JNZ     WIREAD3                 ;CHECK DRIVE NOT READY
01EB  FE 0E 0273 R              DEC     BYTE PTR RETRYC
01EF  79 CF                     JNS     WIREAD1                 ;PERFORM RETRY
01F1  A0 0271 R                 MOV     AL,BYTE PTR RETRYDEF
01F4  A2 0273 R                 MOV     BYTE PTR RETRYC,AL      ;SET RETRY DEFAULT VALUE
01F7  FE 0E 0274 R              DEC     BYTE PTR RETRYC+1
01FB  78 1A                     JS      WIERR                   ;CHECK ERROR TYPE
01FD  C6 06 037A R 10           MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
0202  E8 0397 R                 CALL    FIXDR                   ;RESTORE DRIVE
0205  EB B9                     JMP     WIREAD1                 ;GO READ AGAIN
```

```
                                  ;
0207                              WIREAD3:
0207  E8 00E5 R                           CALL    GETLEN
020A  26: C7 47 12 0000                   MOV     ES:WORD PTR COUNT[BX],00        ;SET NO. OF SECT.PROCESSED
0210  B0 02                               MOV     AL,2                    ;SET DRIVE NOT READY STATUS
0212  EB 1D                               JMP     SHORT   WIERR2
0214                              WIREAD4:
0214  E9 00BB R                           JMP     UNITERR
                                  ;
                                  ;
                                  ;       ERROR ROUTINE
                                  ;
0217                              WIERR:
0217  E8 00E5 R                           CALL    GETLEN
021A  26: 29 4F 12                        SUB     ES:WORD PTR COUNT[BX],CX        ;SET NO. OF PROCESSED SECT.
021E  A0 037D R                           MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS REG.
0221  D0 D8                               RCR     AL,1
0223  72 23                               JC      WIERR8                  ;CONTROLLER ERROR
0225  B1 04                               MOV     CL,4
0227  D2 D8                               RCR     AL,CL
0229  73 09                               JNC     WIERR3                  ;SEEK ERROR
022B  D0 D8                               RCR     AL,1
022D  72 09                               JC      WIERR4                  ;WRITE FAULT
022F  B0 0C                       WIERR1: MOV     AL,12                   ;GENERAL FAILURE
0231  E9 00C9 R                   WIERR2: JMP     ERREXIT                 ;STORE ERROR AND EXIT
                                  ;
0234  B0 06                       WIERR3: MOV     AL,6                    ;SEEK ERROR
0236  EB F9                               JMP     SHORT   WIERR2
0238  B0 0A                       WIERR4: MOV     AL,10                   ;WRITE FAULT
023A  EB F5                               JMP     SHORT   WIERR2
023C  B0 0B                       WIERR5: MOV     AL,11                   ;READ FAULT
023E  EB F1                               JMP     SHORT   WIERR2
0240  B0 08                       WIERR6: MOV     AL,8                    ;SECTOR NOT FOUND
0242  EB ED                               JMP     SHORT   WIERR2
0244  B0 04                       WIERR7: MOV     AL,4                    ;CRC ERROR
0246  EB E9                               JMP     SHORT   WIERR2
0248  A0 037E R                   WIERR8: MOV     AL,BYTE PTR WIPAR+5     ;GET ERROR REGISTER
024B  D0 D8                               RCR     AL,1
024D  72 ED                               JC      WIERR5                  ;ADDR. MARK NOT FOUND
024F  D0 D8                               RCR     AL,1
0251  72 DC                               JC      WIERR1                  ;TRACK 0 ERROR
0253  D0 D8                               RCR     AL,1
0255  72 D8                               JC      WIERR1                  ;ABORTED COMMAND
0257  D0 D8                               RCR     AL,1
0259  D0 D8                               RCR     AL,1
025B  72 E3                               JC      WIERR6                  ;ID NOT FOUND
025D  D0 D8                               RCR     AL,1
025F  72 E3                               JC      WIERR7                  ;CRC ERROR IN ID FIELD
0261  D0 D8                               RCR     AL,1
0263  72 D7                               JC      WIERR5                  ;UNCORRECTABLE DATA
0265  D0 D8                               RCR     AL,1
0267  72 D7                               JC      WIERR6                  ;BAD BLOCK DETECTED
0269  EB C4                               JMP     SHORT   WIERR1
                                  ;
                                  ;
026B                              WIWRTC:
026B  E9 00BF R                           JMP     LENERR
026E                              WIWRTD:
026E  E9 00BB R                           JMP     UNITERR
```

```
                              ;
0271  0000                    RETRYDEF      DW      0            ;RETRY DEFAULT VALUE
0273  0000                    RETRYC        DW      0            ;RETRY COUNT
0275  00                      WRTFLG        DB      0            ; 00=WRITE, FF=WRITE/VERIFY
                              ;
                              ;
                              ;      WRITE DATA
                              ;
0276                          WIWRT:
0276  C6 06 0275 R 00                MOV    BYTE PTR WRTFLG,0    ;SET WRITE DATA FLAG
027B                          WIWRT1:
027B  83 F9 00                       CMP    CX,0                 ;CHECK IF WRITE SECTOR > 0
027E  74 42                          JZ     WIWRT31
0280  E8 00E5 R                      CALL   GETLEN               ;GET DRIVE REQU. STRUCT. LENGTH
0283  3C 16                          CMP    AL,22
0285  72 E4                          JB     WIWRTC               ;BAD STRUCT. LENGTH
0287  E8 0368 R                      CALL   CHKDRIVE             ;CHECK IF UNIT AVAILABLE
028A  72 E2                          JC     WIWRTD               ;UNIT ERROR
028C  E8 0384 R                      CALL   FIXREADY             ;CHECK IF DRIVE READY
028F  75 34                          JNZ    WIWRT3A
0291  A1 0271 R                      MOV    AX,WORD PTR RETRYDEF
0294  A3 0273 R                      MOV    WORD PTR RETRYC,AX   ;SET RETRY COUNT
0297                          WIWRT2:
0297  C6 06 037A R 30                MOV    BYTE PTR WIPAR+1,WRITE  ;SET WRITE FUNCTION
029C  E8 0397 R                      CALL   FIXDR                ;WRITE DATA
029F  A0 037D R                      MOV    AL,BYTE PTR WIPAR+4  ;GET STATUS
02A2  24 F9                          AND    AL,0F9H
02A4  3C 50                          CMP    AL,50H               ;CHECK FOR ERROR
02A6  75 2E                          JNZ    WIWRT4               ;PERFORM RETRIES
02A8  A1 0271 R                      MOV    AX,WORD PTR RETRYDEF
02AB  A3 0273 R                      MOV    WORD PTR RETRYC,AX   ;SET RETRY DEFAULT VALUES
02AE  80 3E 0275 R 00                CMP    BYTE PTR WRTFLG,0
02B3  75 41                          JNZ    WIWRT5               ;GO VERIFY DATA
02B5                          WIWRT3:
02B5  81 06 0381 R 0200              ADD    WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
02BB  FF 06 037B R                   INC    WORD PTR WIPAR+2     ;SECTOR NUMBER +1
02BF  49                             DEC    CX                   ;SECTOR COUNT -1
02C0  75 D5                          JNZ    WIWRT2               ;GO WRITE NEXT SECTOR
02C2                          WIWRT31:
02C2  E9 00CE R                      JMP    EXIT
02C5                          WIWRT3A:
02C5  E8 00E5 R                      CALL   GETLEN
02C8  26: C7 47 12 0000              MOV    ES:WORD PTR COUNT[BX],00       ;SET NO. OF PROCESSED SECT.
02CE  B0 02                          MOV    AL,2                 ;DRIVE NOT READY
02D0  E9 00C9 R                      JMP    ERREXIT
02D3                          WIWRT3B:
02D3  E9 0217 R                      JMP    WIERR                ;CHECK STATUS TYPE
02D6                          WIWRT4:
02D6  24 80                          AND    AL,CBUSY             ;CHECK IF DRIVE NOT READY
02D8  75 EB                          JNZ    WIWRT3A
02DA  FE 0E 0273 R                   DEC    BYTE PTR RETRYC
02DE  79 B7                          JNS    WIWRT2               ;WRITE AGAIN
02E0  A0 0271 R                      MOV    AL,BYTE PTR RETRYDEF
02E3  A2 0273 R                      MOV    BYTE PTR RETRYC,AL   ;SET RETRY DEFAULT VALUE
02E6  FE 0E 0274 R                   DEC    BYTE PTR RETRYC+1
02EA  78 E7                          JS     WIWRT3B              ;WRITE ERROR
02EC  C6 06 037A R 10                MOV    BYTE PTR WIPAR+1,REST  ;SET RESTORE FUNCTION
02F1  E8 0397 R                      CALL   FIXDR                ;RESTORE DRIVE
02F4  EB A1                          JMP    WIWRT2               ;WRITE NEXT SECTOR
```

```
02F6                        WIWRT5:
02F6  8E 06 037F R                  MOV     ES,WORD PTR WIPAR+6
02FA  8B 3E 0381 R                  MOV     DI,WORD PTR WIPAR+8     ;SAVE DATA BUFFER ADDR.
02FE                        WIWRT6:
02FE  A0 0383 R                     MOV     AL,BYTE PTR WIPAR+10    ;GET ACTUAL SDH REG. CONTENTS
0301  E6 C6                         OUT     SDH,AL
0303  B0 20                         MOV     AL,READ
0305  E6 C7                         OUT     COMND,AL                ;OUTPUT READ FUNCTION
0307  E8 0420 R                     CALL    WAIT
030A  E4 C6                         IN      AL,SDH
030C  0C 18                         OR      AL,18H
030E  E6 C6                         OUT     SDH,AL                  ;CLEAR DRIVE LAMP
0310  51                            PUSH    CX
0311  B9 0200                       MOV     CX,512
0314                        WIWRT6A:
0314  E4 C0                         IN      AL,DATA                 ;GET READ DATA
0316  AE                            SCASB                           ;COMPARE WITH DATA WRITTEN
0317  E1 FB                         LOOPZ   WIWRT6A
0319  75 25                         JNZ     WIWRT7
031B                        WIWRT6B:
031B  59                            POP     CX
031C  A0 037D R                     MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
031F  24 F9                         AND     AL,0F9H
0321  3C 58                         CMP     AL,58H
0323  74 2B                         JZ      WIWRT8
0325  FE 0E 0273 R                  DEC     BYTE PTR RETRYC
0329  75 D3                         JNZ     WIWRT6                  ;PERFORM RETRY
032B  C6 06 0273 R 05               MOV     BYTE PTR RETRYC,5
0330  FE 0E 0274 R                  DEC     BYTE PTR RETRYC+1
0334  74 27                         JZ      WIWRTB                  ;READ ERROR
0336  C6 06 037A R 10               MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
033B  E8 0397 R                     CALL    FIXDR                   ;RESTORE DRIVE
033E  EB BE                         JMP     WIWRT6
0340                        WIWRT7:
0340  83 F9 00                      CMP     CX,0
0343  74 04                         JZ      WIWRT7B
0345                        WIWRT7A:
0345  E4 C0                         IN      AL,DATA
0347  E2 FC                         LOOP    WIWRT7A
0349                        WIWRT7B:
0349  80 0E 037D R 20               OR      BYTE PTR WIPAR+4,DWRFA  ;SET WRITE FAULT ERROR
034E  EB CB                         JMP     SHORT   WIWRT6B
0350                        WIWRT8:
0350  C7 06 0273 R 0505             MOV     WORD PTR RETRYC,0505H   ;SET RETRY DEFAULT VALUE
0356  E9 02B5 R                     JMP     WIWRT3
0359                        WIWRTA:
0359  59                            POP     CX
035A  E9 0238 R                     JMP     WIERR4                  ;DATA VERIFY ERROR
035D                        WIWRTB:
035D  E9 0217 R                     JMP     WIERR                   ;SET ERROR STATUS
                            ;
                            ;       WRITE DATA AND VERIFY
                            ;
0360                        WIWRTV:
0360  C6 06 0275 R FF               MOV     BYTE PTR WRTFLG,0FFH    ;SET WRITE/VERIFY FLAG
0365  E9 027B R                     JMP     WIWRT1
```

```
                                        ;
                                        ;       ROUTINE TO CHECK IF UNIT AVAILABLE
                                        ;          EXIT: CARRY ON=UNIT ERROR
                                        ;
0368                                    CHKDRIVE:
0368  50                                        PUSH    AX
0369  A0 0379 R                                 MOV     AL,BYTE PTR WIPAR       ;GET UNIT NO. TO WORK WITH
036C  8A 26 000A E                              MOV     AH,BYTE PTR WINDEV+10   ;GET NO.OF UNITS IN SYSTEM
0370  3A E0                                     CMP     AH,AL
0372  58                                        POP     AX
0373  76 02                                     JBE     CHKDRIVE1
0375  F8                                        CLC
0376  C3                                        RET
0377                                    CHKDRIVE1:
0377  F9                                        STC
0378  C3                                        RET
```

```
                        ;
                        ;*********************************************
                        ;*                                           *
                        ;*     PERIPHERAL INTERFACE MODULE (PIM)     *
                        ;*                                           *
                        ;*            WINCHESTER DISK                *
                        ;*                                           *
                        ;*********************************************
                        ;
                        ;UNIT 0= HEAD 0 AND 1
                        ;UNIT 1= HEAD 2 AND 3
                        ;
                        ;
                        ;          WINCHESTER DISK PARAMETER BLOCK
                        ;          ==============================
                        ;
                        ;
0379  00                WIPAR   DB      0               ; WIPAR + 0     DISK UNIT
037A  10                        DB      REST            ; WIPAR + 1     FUNCTION
037B  0000                      DW      0               ; WIPAR + 2     SECTOR LO
                                                        ; WIPAR + 3     SECTOR HI
037D  00                        DB      0               ; WIPAR + 4     STATUS 1
037E  00                        DB      0               ; WIPAR + 5     STATUS 2
037F  0000                      DW      0               ; WIPAR + 6     BUFFER ADDR.(SEGMENT)
0381  0000                      DW      0               ; WIPAR + 8     BUFFER ADDR.(OFFSET)
0383  00                        DB      0               ; WIPAR + 10    ACTUAL SDH REG. CONTENTS
                        ;
                        ;
                        ;******************************************
                        ;*                                        *
                        ;*   CHECK IF WINCHESTER DRIVE IS         *
                        ;*    CONNECTED AND POWERED ON.           *
                        ;*                                        *
                        ;*    EXIT: ZERO FLAG ON = DRIVE READY    *
                        ;*                                        *
                        ;******************************************
                        ;
0384                    FIXREADY:
0384  B0 55                     MOV     AL,55H
0386  E6 C4                     OUT     CYLLO,AL                ;OUTPUT PATTERN TO R/W PORT
0388  B0 AA                     MOV     AL,0AAH
038A  E6 C3                     OUT     SECNO,AL
038C  E4 C4                     IN      AL,CYLLO                ;READ PATTERN BACK AND COMPARE
038E  3C 55                     CMP     AL,55H
0390  75 04                     JNZ     FIXREADY1
0392  E4 C3                     IN      AL,SECNO
0394  3C AA                     CMP     AL,0AAH
0396                    FIXREADY1:
0396  C3                        RET
```

```
                                ;****************************************
                                ;*                                      *
                                ;*    WINCHESTER DISK DRIVER            *
                                ;*                                      *
                                ;*     ENTRY: PARAMETER BLOCK FILLED UP *
                                ;*     EXIT:  STATUS BYTES IN PARAM.    *
                                ;*            BLOCK UPDATED AND ALL     *
                                ;*            REGISTERS SAVED.          *
                                ;****************************************
0397  50                        FIXDR:  PUSH    AX
0398  53                                PUSH    BX
0399  51                                PUSH    CX
039A  52                                PUSH    DX
039B  A1 037B R                         MOV     AX,WORD PTR WIPAR+2     ;GET LOGIC SECTOR NUMBER
039E  B9 0011                           MOV     CX,17
03A1  BA 0000                           MOV     DX,0
03A4  F7 F1                             DIV     CX                      ;CALCULATE CYL/HEAD
03A6  50                                PUSH    AX
03A7  8A C2                             MOV     AL,DL
03A9  E6 C3                             OUT     SECNO,AL                ;SET SECTOR NUMBER
03AB  8A 1E 0379 R                      MOV     BL,BYTE PTR WIPAR       ;GET DISK UNIT #
03AF  8A FB                             MOV     BH,BL
03B1  81 E3 0601                        AND     BX,0601H
03B5  D0 C7                             ROL     BH,1
03B7  0A DF                             OR      BL,BH
03B9  D0 C3                             ROL     BL,1
03BB  58                                POP     AX
03BC  50                                PUSH    AX
03BD  24 01                             AND     AL,01H                  ;GET HEAD BIT
03BF  0A C3                             OR      AL,BL
03C1  0C A0                             OR      AL,SDHREG               ;ECC/CRC AND BYTES PER SECTOR
03C3  E6 C6                             OUT     SDH,AL                  ;SET ECC/CRC-BYTES/SECT-DRIVE-HEAD
03C5  A2 0383 R                         MOV     BYTE PTR WIPAR+10,AL    ;SAVE ACTUAL SDH REG. CONTENTS
03C8  58                                POP     AX
03C9  D1 C8                             ROR     AX,1
03CB  E6 C4                             OUT     CYLLO,AL                ;SET CYLINDER LOW
03CD  80 E4 03                          AND     AH,03H
03D0  8A C4                             MOV     AL,AH
03D2  E6 C5                             OUT     CYLHI,AL                ;SET CYLINDER HIGH
03D4  E4 C7                             IN      AL,STAT                 ;GET DISK STATUS
03D6  A2 037D R                         MOV     BYTE PTR WIPAR+4,AL
03D9  24 80                             AND     AL,CBUSY                ;CHECK IF CONTROLLER BUSY
03DB  75 16                             JNZ     FIXD3
03DD  A0 037A R                         MOV     AL,BYTE PTR WIPAR+1
03E0  E6 C7                             OUT     COMND,AL                ;SET FUNCTION
03E2  24 F0                             AND     AL,0F0H
03E4  3C 20                             CMP     AL,READ
03E6  74 16                             JZ      RD                      ;GO READ DATA
03E8  3C 30                             CMP     AL,WRITE
03EA  74 4E                             JZ      WR                      ;GO WRITE DATA
03EC  3C 50                             CMP     AL,FORMAT
03EE  74 46                             JZ      WR0                     ;GO FORMAT ONE TRACK
03F0  EB 5C 90                          JMP     WR2                     ;SEEK OR RESTORE
03F3  E4 C6                     FIXD3:  IN      AL,SDH
03F5  0C 10                             OR      AL,10H
03F7  E6 C6                             OUT     SDH,AL                  ;CLEAR DISK LAMP
03F9  5A                                POP     DX
03FA  59                                POP     CX
03FB  5B                                POP     BX
03FC  58                                POP     AX
03FD  C3                                RET
```

```
                                ;
                                ;       ****************************
                                ;       *    READ ROUTINE          *
                                ;       ****************************
                                ;
03FE  E8 0420 R                 RD:     CALL    WAIT                    ;WAIT UNTIL READ COMPLETE
0401  1E                                PUSH    DS
0402  8B 1E 0381 R                      MOV     BX,WORD PTR WIPAR+8     ;GET OFFSET
0406  8E 1E 037F R                      MOV     DS,WORD PTR WIPAR+6     ;GET SEGMENT ADDR.
040A  B9 0200                           MOV     CX,512                  ;INPUT COUNT
040D  E4 C0                     RD2:    IN      AL,DATA                 ;INPUT DATA
040F  88 07                             MOV     BYTE PTR[BX],AL         ;SAVE INPUT
0411  43                                INC     BX
0412  E0 F9                             LOOPNZ  RD2             ;CONTINUE UNTIL ALL BYTES IN BUFFER
                                                                ;BUT STOP BEFORE BUFFER ADDR. WRAP AROUND
0414  83 F9 00                          CMP     CX,0
0417  74 04                             JZ      RD4
0419  E4 C0                     RD3:    IN      AL,DATA                 ;CLEAR CONTROLLER BUFFER
041B  E2 FC                             LOOP    RD3
041D  1F                        RD4:    POP     DS
041E  EB D3                             JMP     SHORT   FIXD3
                                ;
                                ;
                                ;       ****************************
                                ;       *    WAIT ROUTINE          *
                                ;       ****************************
                                ;
0420  E4 C7                     WAIT:   IN      AL,STAT                 ;GET STATUS
0422  24 80                             AND     AL,CBUSY
0424  75 FA                             JNZ     WAIT                    ;LOOP UNTIL DISK READY
0426  E4 C7                             IN      AL,STAT
0428  A2 037D R                         MOV     BYTE PTR WIPAR+4,AL     ;SAVE STATUS
042B  D0 D8                             RCR     AL,1
042D  72 01                             JC      ER1                     ;JUMP IF ERROR CONDITION
042F  C3                                RET
                                ;
0430  E4 C1                     ER1:    IN      AL,ERROR                ;GET ERROR STATUS
0432  A2 037E R                         MOV     BYTE PTR WIPAR+5,AL     ;SAVE STATUS
0435  C3                                RET
                                ;
                                ;       ****************************
                                ;       *    WRITE ROUTINE         *
                                ;       ****************************
                                ;
0436  B0 11                     WR0:    MOV     AL,17
0438  E6 C2                             OUT     SECNT,AL                ;SET SECT COUNT FOR FORMAT
                                ;
043A  1E                        WR:     PUSH    DS
043B  8B 1E 0381 R                      MOV     BX,WORD PTR WIPAR+8     ;BUFFER ADDR.(OFFSET)
043F  8E 1E 037F R                      MOV     DS,WORD PTR WIPAR+6     ;BUFFER ADDR.(SEGMENT)
0443  B9 0200                           MOV     CX,512                  ;INPUT COUNT
0446  8A 07                     WR1:    MOV     AL,BYTE PTR[BX]         ;GET BYTE FROM BUFFER
0448  E6 C0                             OUT     DATA,AL                 ;OUTPUT DATA
044A  43                                INC     BX
044B  E2 F9                             LOOP    WR1
044D  1F                                POP     DS
044E  E8 0420 R                 WR2:    CALL    WAIT                    ;WAIT UNTIL FUNCT. COMPLETE
0451  EB A0                             JMP     SHORT   FIXD3
                                ;
                                ;
                                ;
0453                            WIINIT:
0453  E8 00E5 R                         CALL    GETLEN                  ;GET DRIVE REQU. STRUCT.LENGTH
0456  3C 16                             CMP     AL,22
0458  72 5A                             JB      WIINIT1                 ;BAD STRUCT. LENGTH
045A  A0 0000 E                         MOV     AL,WI_FLAGS
045D  0C 7F                             OR      AL,7FH                  ;SET UNKNOWN DISK
045F  A2 0000 E                         MOV     WI_FLAGS,AL
0462  A0 0000 E                         MOV     AL,WINCH_DRIVES         ;GET NO. OF DRIVES ON SYSTEM
0465  D0 C0                             ROL     AL,1                    ;MAKE NO. OF UNITS
0467  8A 0E 0000 E                      MOV     CL,WI_OUT_RETRIES       ;GET RETRY (OUT) COUNTS
046B  8A 2E 0000 E                      MOV     CH,WI_IN_RETRIES        ;GET RETRY (IN) COUNTS
046F  88 2E 0271 R                      MOV     BYTE PTR RETRYDEF,CH
0473  88 0E 0272 R                      MOV     BYTE PTR RETRYDEF+1,CL  ;SAVE RETRY COUNTS
0477  A2 000A E                         MOV     BYTE PTR WINDEV+10,AL   ;SET NUMBER OF UNITS
047A  26: 88 47 0D                      MOV     ES:BYTE PTR MEDIA[BX],AL
047E  26: C7 47 0E 0453 R               MOV     ES:WORD PTR TRANS[BX],OFFSET WIINIT
0484  26: 8C 4F 10                      MOV     ES:WORD PTR TRANS+2[BX],CS
0488  26: C7 47 12 0087 R               MOV     ES:WORD PTR COUNT[BX],OFFSET WIBPB      ;ADDR. OF BPB ARRAY
048E  26: 8C 4F 14                      MOV     ES:WORD PTR COUNT+2[BX],CS
0492  C7 06 006F R 00CE R               MOV     WORD PTR WITBL,OFFSET EXIT
0498  E8 0384 R                         CALL    FIXREADY
049B  75 1F                             JNZ     WIINIT4
049D  C6 06 037A R 10                   MOV     BYTE PTR WIPAR+1,REST
04A2  80 26 0379 R 01                   AND     BYTE PTR WIPAR,01
04A7  E8 0397 R                         CALL    FIXDR                   ;RESTORE DRIVE
04AA  80 3E 037D R 50                   CMP     BYTE PTR WIPAR+4,50H
04AF  75 06                             JNZ     WIINIT2                 ;ERROR CONDITION
04B1  E9 00CE R                         JMP     EXIT
04B4                            WIINIT1:
04B4  E9 00BF R                         JMP     LENERR
04B7                            WIINIT2:
04B7  A0 037D R                         MOV     AL,BYTE PTR WIPAR+4
04BA  24 80                             AND     AL,CBUSY
04BC                            WIINIT4:
04BC  B0 02                             MOV     AL,2                    ;DRIVE NOT READY
04BE  75 02                             JNZ     WIINIT3
04C0  B0 0C                             MOV     AL,12                   ;GENERAL FAILURE
04C2                            WIINIT3:
04C2  E9 00C9 R                         JMP     ERREXIT                 ;SAVE STATUS AND EXIT
                                ;
                                ;
04C5                            WI_END:
                                ;
04C5                            CSEG    ENDS
                                ;
                                ;
                                END     BEGIN
```

```
Segments and groups:

                N a m e                 Size    align   combine class

CSEG . . . . . . . . . . . . . .        04C5    PARA    PUBLIC  'CODE'

Symbols:

                N a m e                 Type    Value   Attr

ABC. . . . . . . . . . . . . . .        Number  0004
BBD. . . . . . . . . . . . . . .        Number  0080
BEGIN. . . . . . . . . . . . . .        L NEAR  0000    CSEG
BUSEXIT. . . . . . . . . . . . .        L NEAR  00B7    CSEG
CBUSY. . . . . . . . . . . . . .        Number  0080
CDRQ . . . . . . . . . . . . . .        Number  0008
CERR . . . . . . . . . . . . . .        Number  0001
CHKDRIVE . . . . . . . . . . . .        L NEAR  0368    CSEG
CHKDRIVE1. . . . . . . . . . . .        L NEAR  0377    CSEG
CMD. . . . . . . . . . . . . . .        Number  0002
CMDERR . . . . . . . . . . . . .        L NEAR  00C7    CSEG
CMDLEN . . . . . . . . . . . . .        Number  0000
COMND. . . . . . . . . . . . . .        Number  00C7
CORRD. . . . . . . . . . . . . .        Number  0004
COUNT. . . . . . . . . . . . . .        Number  0012
CRCID. . . . . . . . . . . . . .        Number  0020
CYLHI. . . . . . . . . . . . . .        Number  00C5
CYLLO. . . . . . . . . . . . . .        Number  00C4
DAMNFD . . . . . . . . . . . . .        Number  0001
DATA . . . . . . . . . . . . . .        Alias   HBASE
DREADY . . . . . . . . . . . . .        Number  0040
DSEEC. . . . . . . . . . . . . .        Number  0010
DWRFA. . . . . . . . . . . . . .        Number  0020
ENTRY. . . . . . . . . . . . . .        L NEAR  0016    CSEG
ENTRY1 . . . . . . . . . . . . .        L NEAR  003E    CSEG
ER1. . . . . . . . . . . . . . .        L NEAR  0430    CSEG
ERREXIT. . . . . . . . . . . . .        L NEAR  00C9    CSEG
ERROR. . . . . . . . . . . . . .        Number  00C1
EXIT . . . . . . . . . . . . . .        L NEAR  00CE    CSEG
EXIT1. . . . . . . . . . . . . .        L NEAR  00D8    CSEG
EXITP. . . . . . . . . . . . . .        F PROC  00CE    CSEG    Length =0017
FIXD3. . . . . . . . . . . . . .        L NEAR  03F3    CSEG
FIXDR. . . . . . . . . . . . . .        L NEAR  0397    CSEG
FIXREADY . . . . . . . . . . . .        L NEAR  0384    CSEG
FIXREADY1. . . . . . . . . . . .        L NEAR  0396    CSEG
FORMAT . . . . . . . . . . . . .        Number  0050
GETBPB . . . . . . . . . . . . .        L NEAR  0116    CSEG
GETBPB1. . . . . . . . . . . . .        L NEAR  0165    CSEG
GETBPB2. . . . . . . . . . . . .        L NEAR  016E    CSEG
GETBPB3. . . . . . . . . . . . .        L NEAR  0171    CSEG
GETBPB5. . . . . . . . . . . . .        L NEAR  0176    CSEG
GETBPB6. . . . . . . . . . . . .        L NEAR  017C    CSEG
GETBPB7. . . . . . . . . . . . .        L NEAR  0185    CSEG
GETBPB8. . . . . . . . . . . . .        L NEAR  0191    CSEG
GETLEN . . . . . . . . . . . . .        L NEAR  00E5    CSEG
HBASE. . . . . . . . . . . . . .        Number  00C0
IDNFD. . . . . . . . . . . . . .        Number  0010
LENERR . . . . . . . . . . . . .        L NEAR  00BF    CSEG
MEDERR . . . . . . . . . . . . .        L NEAR  00C3    CSEG
MEDIA. . . . . . . . . . . . . .        Number  000D
MEDIAC . . . . . . . . . . . . .        L NEAR  00F1    CSEG
```

```
MEDIAC1. . . . . . . . . . . . .        L NEAR  010F    CSEG
PTRSAV . . . . . . . . . . . . .        L WORD  000E    CSEG
RD . . . . . . . . . . . . . . .        L NEAR  03FE    CSEG
RD2. . . . . . . . . . . . . . .        L NEAR  040D    CSEG
RD3. . . . . . . . . . . . . . .        L NEAR  0419    CSEG
RD4. . . . . . . . . . . . . . .        L NEAR  041D    CSEG
READ . . . . . . . . . . . . . .        Number  0020
REST . . . . . . . . . . . . . .        Number  0010
RETRYC . . . . . . . . . . . . .        L WORD  0273    CSEG
RETRYDEF . . . . . . . . . . . .        L WORD  0271    CSEG
SDH. . . . . . . . . . . . . . .        Number  00C6
SDHREG . . . . . . . . . . . . .        Number  00A0
SECNO. . . . . . . . . . . . . .        Number  00C3
SECNT. . . . . . . . . . . . . .        Number  00C2
SEEK . . . . . . . . . . . . . .        Number  0070
START. . . . . . . . . . . . . .        Number  0014
STAT . . . . . . . . . . . . . .        Number  00C7
STATUS . . . . . . . . . . . . .        Number  0003
STRATE . . . . . . . . . . . . .        Number  0000
STRATP . . . . . . . . . . . . .        F PROC  0003    CSEG    Length =000B
TR0. . . . . . . . . . . . . . .        Number  0002
TRANS. . . . . . . . . . . . . .        Number  000E
UNCOR. . . . . . . . . . . . . .        Number  0040
UNIT . . . . . . . . . . . . . .        Number  0001
UNITERR. . . . . . . . . . . . .        L NEAR  000B    CSEG
WAIT . . . . . . . . . . . . . .        L NEAR  0420    CSEG
WIBPB. . . . . . . . . . . . . .        L WORD  0087    CSEG
WIBPB1A. . . . . . . . . . . . .        L WORD  00AA    CSEG
WIBPB1C. . . . . . . . . . . . .        L WORD  0097    CSEG
WIERR. . . . . . . . . . . . . .        L NEAR  0217    CSEG
WIERR1 . . . . . . . . . . . . .        L NEAR  022F    CSEG
WIERR2 . . . . . . . . . . . . .        L NEAR  0231    CSEG
WIERR3 . . . . . . . . . . . . .        L NEAR  0234    CSEG
WIERR4 . . . . . . . . . . . . .        L NEAR  0238    CSEG
WIERR5 . . . . . . . . . . . . .        L NEAR  023C    CSEG
WIERR6 . . . . . . . . . . . . .        L NEAR  0240    CSEG
WIERR7 . . . . . . . . . . . . .        L NEAR  0244    CSEG
WIERR8 . . . . . . . . . . . . .        L NEAR  0248    CSEG
WIINIT . . . . . . . . . . . . .        L NEAR  0453    CSEG
WIINIT1. . . . . . . . . . . . .        L NEAR  04B4    CSEG
WIINIT2. . . . . . . . . . . . .        L NEAR  04B7    CSEG
WIINIT3. . . . . . . . . . . . .        L NEAR  04C2    CSEG
WIINIT4. . . . . . . . . . . . .        L NEAR  04BC    CSEG
WINCH_DRIVES . . . . . . . . . .        V BYTE  0000            External
WINDEV . . . . . . . . . . . . .        V WORD  0000            External
WIPAR. . . . . . . . . . . . . .        L BYTE  0379    CSEG
WIREAD . . . . . . . . . . . . .        L NEAR  01A4    CSEG
WIREAD1. . . . . . . . . . . . .        L NEAR  01C0    CSEG
WIREAD1A . . . . . . . . . . . .        L NEAR  01E4    CSEG
WIREAD2. . . . . . . . . . . . .        L NEAR  01E7    CSEG
WIREAD3. . . . . . . . . . . . .        L NEAR  0207    CSEG
WIREAD4. . . . . . . . . . . . .        L NEAR  0214    CSEG
WITBL. . . . . . . . . . . . . .        L WORD  006F    CSEG
WIWRT. . . . . . . . . . . . . .        L NEAR  0276    CSEG
WIWRT1 . . . . . . . . . . . . .        L NEAR  027B    CSEG
WIWRT2 . . . . . . . . . . . . .        L NEAR  0297    CSEG
WIWRT3 . . . . . . . . . . . . .        L NEAR  02B5    CSEG
WIWRT31. . . . . . . . . . . . .        L NEAR  02C2    CSEG
WIWRT3A. . . . . . . . . . . . .        L NEAR  02C5    CSEG
WIWRT3B. . . . . . . . . . . . .        L NEAR  02D3    CSEG
WIWRT4 . . . . . . . . . . . . .        L NEAR  02D6    CSEG
```

```
WIWRT5 . . . . . . . . . . . . . .        L NEAR  02F6    CSEG
WIWRT6 . . . . . . . . . . . . . .        L NEAR  02FE    CSEG
WIWRT6A. . . . . . . . . . . . . .        L NEAR  0314    CSEG
WIWRT6B. . . . . . . . . . . . . .        L NEAR  031B    CSEG
WIWRT7 . . . . . . . . . . . . . .        L NEAR  0340    CSEG
WIWRT7A. . . . . . . . . . . . . .        L NEAR  0345    CSEG
WIWRT7B. . . . . . . . . . . . . .        L NEAR  0349    CSEG
WIWRT8 . . . . . . . . . . . . . .        L NEAR  0350    CSEG
WIWRTA . . . . . . . . . . . . . .        L NEAR  0359    CSEG
WIWRTB . . . . . . . . . . . . . .        L NEAR  035D    CSEG
WIWRTC . . . . . . . . . . . . . .        L NEAR  026B    CSEG
WIWRTD . . . . . . . . . . . . . .        L NEAR  026E    CSEG
WIWRTV . . . . . . . . . . . . . .        L NEAR  0360    CSEG
WI_END . . . . . . . . . . . . . .        L NEAR  04C5    CSEG    Global
WI_FLAGS . . . . . . . . . . . . .        V BYTE  0000            External
WI_INTERRUPT . . . . . . . . . . .        L NEAR  0012    CSEG    Global
WI_INTERRUPT1. . . . . . . . . . .        L NEAR  006C    CSEG
WI_IN_RETRIES. . . . . . . . . . .        V BYTE  0000            External
WI_OUT_RETRIES . . . . . . . . . .        V BYTE  0000            External
WI_START . . . . . . . . . . . . .        L NEAR  0000    CSEG    Global
WI_STRATEGY. . . . . . . . . . . .        L NEAR  0003    CSEG    Global
WPC. . . . . . . . . . . . . . . .        Number  00C1
WR . . . . . . . . . . . . . . . .        L NEAR  043A    CSEG
WR0. . . . . . . . . . . . . . . .        L NEAR  0436    CSEG
WR1. . . . . . . . . . . . . . . .        L NEAR  0446    CSEG
WR2. . . . . . . . . . . . . . . .        L NEAR  044E    CSEG
WRITE. . . . . . . . . . . . . . .        Number  0030
WRTFLG . . . . . . . . . . . . . .        L BYTE  0275    CSEG
```

```
                        ;**************************************************
                        ;*                                                *
                        ;*      W I N C H E S T E R   D I S K            *
                        ;*                                                *
                        ;*             D R I V E R                        *
                        ;*                                                *
                        ;**************************************************
                        ;
                        ;
                        ;
                        ;    DEFINE OFFSETS FOR IO DATA PACKET
                        ;
= 0000                  CMDLEN  EQU     0               ;LENGTH OF THIS BLOCK
= 0001                  UNIT    EQU     1               ;SUB UNIT SPECIFIER
= 0002                  CMD     EQU     2               ;COMMAND CODE
= 0003                  STATUS  EQU     3               ;STATUS
= 000D                  MEDIA   EQU     13              ;MEDIA DESCRIPTOR
= 000E                  TRANS   EQU     14              ;TRANSFER ADDRESS
= 0012                  COUNT   EQU     18              ;COUNT OF SECTORS
= 0014                  START   EQU     20              ;FIRST BLOCK TO TRANSFER
                        ;
                        ;
                        ;
                        ;        WINCHESTER DISK DEFINITIONS
                        ;        ==========================
                        ;
                        ;*********************************************
                        ;*                                           *
                        ;*    PORT DEFINITIONS                       *
                        ;*                                           *
                        ;*********************************************
                        ;
= 00C0                  HBASE   EQU     0C0H            ;     CONTROLLER BASE ADDR.
=                       DATA    EQU     HBASE           ; R/W DATA REGISTER
= 00C1                  ERROR   EQU     HBASE+1         ; R   ERROR REGISTER
= 00C1                  WPC     EQU     HBASE+1         ;   W WRITE PRECOMP. REGISTER
= 00C2                  SECNT   EQU     HBASE+2         ; R/W SECTOR COUNT REGISTER
= 00C3                  SECNO   EQU     HBASE+3         ; R/W SECTOR NUMBER REGISTER
= 00C4                  CYLLO   EQU     HBASE+4         ; R/W CYLINDER LOW REGISTER
= 00C5                  CYLHI   EQU     HBASE+5         ; R/W CYLINDER HIGH REGISTER
= 00C6                  SDH     EQU     HBASE+6         ; R/W ECC/CRC-BYTES PÈR SECTOR-DRIVE-HEAD
= 00C7                  STAT    EQU     HBASE+7         ; R   STATUS REGISTER
= 00C7                  COMND   EQU     HBASE+7         ;   W COMMAND REGISTER
                        ;
                        ;
                        ;*********************************************
                        ;*                                           *
                        ;*    DISK FUNCTIONS                         *
                        ;*                                           *
                        ;*********************************************
                        ;
= 0000                  STRATE  EQU     0               ;STEPPING RATE TRACK TO TRACK = BUFFERED STEP
= 0010                  REST    EQU     10H OR STRATE   ;RESTORE COMMAND WITH STRATE
= 0070                  SEEK    EQU     70H OR STRATE   ;SEEK COMMAND WITH STRATE
= 0020                  READ    EQU     20H             ;READ COMMAND
= 0030                  WRITE   EQU     30H             ;WRITE COMMAND
= 0050                  FORMAT  EQU     50H             ;FORMAT COMMAND
```

```
                              ;
                              ;*********************************************
                              ;*                                           *
                              ;*      ERRROR  REGISTER  EQUATES            *
                              ;*                                           *
                              ;*********************************************
                              ;
 = 0001                       DAMNFD  EQU     01H             ; ADDR. MARK NOT FOUND
 = 0002                       TR0     EQU     02H             ; TRACK 0 ERROR
 = 0004                       ABC     EQU     04H             ; ABORTED COMMAND
 = 0010                       IDNFD   EQU     10H             ; ID NOT FOUND
 = 0020                       CRCID   EQU     20H             ; CRC-ERROR  ID-FIELD
 = 0040                       UNCOR   EQU     40H             ; UNCORRECTED DATA IN DATA FIELD
 = 0080                       BBD     EQU     80H             ; BAD BLOCK DETECTED
                              ;
                              ;*********************************************
                              ;*                                           *
                              ;*      STATUS REGISTER EQUATES              *
                              ;*                                           *
                              ;*********************************************
                              ;
 = 0001                       CERR    EQU     01H             ; CONTROLLER ERROR
 = 0004                       CORRD   EQU     04H             ; DATA CORRECTED IN DATA FIELD (ECC)
 = 0008                       CDRQ    EQU     08H             ; CONTROLLER DATA REQUEST
 = 0010                       DSEEC   EQU     10H             ; DRIVE SEEK COMPLETE
 = 0020                       DWRFA   EQU     20H             ; DRIVE WRITE FAULT
 = 0040                       DREADY  EQU     40H             ; DRIVE READY
 = 0080                       CBUSY   EQU     80H             ; CONTROLLER BUSY
                              ;
                              ;
                              ;*********************************************
                              ;*                                           *
                              ;*      SPECIALS                             *
                              ;*                                           *
                              ;*********************************************
                              ;
 = 00A8                       SDHREG  EQU     0A8H            ;ECC/512 BYTES PER SECTOR
                              ;
                              ;
                              ;
                              PUBLIC WI_STRATEGY              ;STRATEGY ENTRY POINT
                              PUBLIC WI_INTERRUPT             ;INTERRUPT ENTRY POINT
                              PUBLIC WI_START                 ;BEGIN OF DRIVER
                              PUBLIC WI_END                   ;END OF DRIVER
                              ;
                              ;
                              EXTRN WINDEV:WORD
                              EXTRN WINCH_DRIVES:BYTE         ;NO. OF WINCHESTER DRIVES
                              EXTRN WI_OUT_RETRIES:BYTE       ;NO. OF RETRIES  (OUT)
                              EXTRN WI_IN_RETRIES:BYTE        ;NO. OF RETRIES  (IN)
                              EXTRN WI_FLAGS:BYTE
```

```
                            ;
0000                        CSEG    SEGMENT PUBLIC  'CODE'
                            ASSUME  CS:CSEG,DS:CSEG,ES:CSEG,SS:CSEG

0000                                ORG     0
                            ;
0000                        BEGIN:
                            ;
                            ;*****************************************************
                            ;*                                                   *
                            ;*   S P E C I A L   D E V I C E   H E A D E R       *
                            ;*                                                   *
                            ;*****************************************************
                            ;
                            ;-----------------------------------------------+
                            ;     DWORD pointer to next device             ! 1 word offset.
                            ;        (-1,-1 if last device)                ! 1 word segement.
                            ;-----------------------------------------------+
                            ;     Device attribute WORD                    ; 1 word.
                            ;       Bit 15 = 1 for character devices.      ;
                            ;                0 for block devices.          ;
                            ;                                              ;
                            ;       Charcter devices. (Bit 15=1)           ;
                            ;         Bit 0 = 1  current sti device.       ;
                            ;         Bit 1 = 1  current sto device.       ;
                            ;         Bit 2 = 1  current NUL device.       ;
                            ;         Bit 3 = 1  current Clock device.     ;
                            ;                                              ;
                            ;         Bit 14 = 1 IOCTL control bit.        ;
                            ;-----------------------------------------------+
                            ;     Device strategy pointer.                 ; 1 word offset.
                            ;-----------------------------------------------+
                            ;     Device interrupt pointer.                ; 1 word offset.
                            ;-----------------------------------------------+
                            ;     Device name field.                       ; 8 bytes.
                            ;       Character devices are any valid name   ;
                            ;         left justified, in a space filled    ;
                            ;         field.                               ;
                            ;       Block devices contain # of units in    ;
                            ;         the first byte.                      ;
                            ;-----------------------------------------------+
                            ;
                            ;
0000                        WI_START:
                            ;
                            ;       RELEASE ID
0000  02                            DB      02              ;ISSUE
0001  01                            DB      01              ;SUB ISSUE
0002  00                            DB      00              ;PATCH LEVEL
```

```
                        ;
                        ;
                        ;   SIMPLISTIC STRATEGY ROUTINE FOR NON-MULTI-TASKING SYSTEM
                        ;
                        ;       CURRENTLY JUST SAVES I/O PACKET POINTER IN PTRSAV
                        ;       FOR LATER PROCESSING BY THE INTERRUPT ROUTINE.
                        ;
0003                    STRATP  PROC    FAR
                        ;
0003                    WI_STRATEGY:
0003  2E: 89 1E 000E R          MOV     CS:WORD PTR PTRSAV,BX
0008  2E: 8C 06 0010 R          MOV     CS:WORD PTR PTRSAV+2,ES
000D  CB                        RET
                        ;
000E                    STRATP  ENDP
                        ;
000E  0000 0000         PTRSAV  DW      0,0             ;STRATEGY POINTER SAVE
                        ;
                        ;
0012                    WI_INTERRUPT:
0012  56                        PUSH    SI
0013  BE 006F R                 MOV     SI,OFFSET WITBL
                        ;
0016  50                ENTRY:  PUSH    AX                      ;SAVE ALL NECESSARY REGISTERS.
0017  51                        PUSH    CX
0018  52                        PUSH    DX
0019  57                        PUSH    DI
001A  1E                        PUSH    DS
001B  06                        PUSH    ES
001C  53                        PUSH    BX
001D  0E                        PUSH    CS
001E  1F                        POP     DS                      ;SET DATA SEG. TO CODE SEG.
001F  8B 1E 000E R              MOV     BX,WORD PTR PTRSAV      ;Retrieve pointer to I/O Packet.
0023  8E 06 0010 R              MOV     ES,WORD PTR PTRSAV+2
0027  26: 8A 47 01              MOV     AL,ES:UNIT[BX]          ;AL = Unit code.
002B  A2 0351 R                 MOV     BYTE PTR WIPAR,AL
002E  26: 8A 67 0D              MOV     AH,ES:MEDIA[BX]         ;AH = Media descriptor.
0032  26: 80 7F 02 00           CMP     ES:BYTE PTR CMD[BX],0
0037  74 05                     JZ      ENTRY1                  ;SKIP MEDIA CHECK IF INIT FUNCT.
0039  80 FC F8                  CMP     AH,0F8H
003C  75 2E                     JNZ     WI_INTERRUPT1           ;Unknown Media descriptor
003E                    ENTRY1:
003E  26: 8B 4F 12              MOV     CX,ES:COUNT[BX]         ;CX = Contains byte/sector count.
0042  26: 8B 57 14              MOV     DX,ES:START[BX]         ;DX = Starting Logical sector.
0046  89 16 0353 R              MOV     WORD PTR WIPAR+2,DX
004A  97                        XCHG    DI,AX                   ;Move Unit/Media into DI temporarily
004B  26: 8A 47 02              MOV     AL,ES:CMD[BX]           ;Retrieve Command type.
004F  32 E4                     XOR     AH,AH                   ;Clear upper half of AX for calc.
0051  03 F0                     ADD     SI,AX                   ;Comp. entry point in funct. table.
0053  03 F0                     ADD     SI,AX
0055  3C 0B                     CMP     AL,11                   ;Not more than 11 commands.
0057  77 59                     JA      CMDERR                  ;Ah, well, error out.
0059  97                        XCHG    AX,DI                   ;Move Unit & Media back.
005A  26: 8B 7F 10              MOV     DI,ES:TRANS+2 [BX]      ;DI = addess of Transfer address.
005E  89 3E 0357 R              MOV     WORD PTR WIPAR+6,DI     ;Buffer Addr. to PIM table
0062  26: 8B 7F 0E              MOV     DI,ES:TRANS [BX]
0066  89 3E 0359 R              MOV     WORD PTR WIPAR+8,DI
006A  FF 24                     JMP     WORD PTR[SI]            ;Perform I/O packet command.
006C                    WI_INTERRUPT1:
006C  EB 40 90                  JMP     MEDERR                  ;UNKNOWN MEDIA DESCRIPTOR
```

```
                                ;
                                ;     WINCHESTER DISK FUNCTION TABLE
                                ;
006F  0427 R                    WITBL   DW      WIINIT          ;0  - Initialize Driver.
0071  00DC R                            DW      MEDIAC          ;1  - Return current media code.
0073  0101 R                            DW      GETBPB          ;2  - Get Bios Parameter Block.
0075  00B2 R                            DW      CMDERR          ;3  - Reserved. (currently returns error)
0077  017C R                            DW      WIREAD          ;4  - Block read.
0079  00A2 R                            DW      BUSEXIT         ;5  - (Not used, return busy flag)
007B  00B9 R                            DW      EXIT            ;6  - Return status. (Not used)
007D  00B9 R                            DW      EXIT            ;7  - Flush input buffer. (Not used.)
007F  024E R                            DW      WIWRT           ;8  - Block write.
0081  033B R                            DW      WIWRTV          ;9  - Block write with verify.
0083  00B9 R                            DW      EXIT            ;10 - Return output status.
0085  00B9 R                            DW      EXIT            ;11 - Flush output buffer. (Not used.)
                                ;
                                ;
                                ;      BIOS PARAMETER BLOCK ARRAY
                                ;
0087  008F R                    WIBPB   DW      WIBPB1
0089  008F R                            DW      WIBPB1
008B  008F R                            DW      WIBPB1
008D  008F R                            DW      WIBPB1
                                ;
                                ;      SEAGATE WITHOUT BOOT RECORD
                                ;
                                ;
                                ;
008F  0200                      WIBPB1  DW      512             ;BYTES PER SECTOR
0091  08                                DB      8               ;SECTOR PER ALLOCATION UNIT
0092  0001                              DW      1               ;RESERVED SECTORS
0094  02                                DB      2               ;NUMBER OF FAT'S
0095  0200                              DW      512             ;NUMBER OF ROOT DIRECT. ENTRIES
0097  5104                              DW      20740           ;NUMBER OF SECTORS PER DISK
0099  F8                                DB      0F8H            ;MEDIA DESCRIPTOR
009A  0008                              DW      8               ;NUMBER OF FAT SECTORS
009C  0011                              DW      17              ;SECTORS PER TRACK
009E  0004                              DW      4               ;NUMBER OF HEADS
00A0  0000                              DW      0               ;HIDDEN SECTORS
                                ;
                                ; COMMON ERROR PROCESSING ROUTINE.
                                ;  AL = ERROR CODE.
                                ;
                                ;  Error # 0 = Write Protect violation.
                                ;          1 = Unknown unit.
                                ;          2 = Drive not ready.
                                ;          3 = Unknown command in I/O packet.
                                ;          4 = CRC error.
                                ;          5 = Bad drive request structure length.
                                ;          6 = Seek error.
                                ;          7 = Unknown media discovered.
                                ;          8 = Sector not found.
                                ;          9 = Printer out of paper.
                                ;         10 = Write fault.
                                ;         11 = Read fault.
                                ;         12 = General failure.
```

```
                                    ;
00A2                                BUSEXIT:                          ;Device busy exit.
00A2  B4 03                                 MOV     AH,00000011B      ;Set busy and done bits.
00A4  EB 15                                 JMP     SHORT   EXIT1
00A6                                UNITERR:
00A6  B0 01                                 MOV     AL,1
00A8  EB 0A                                 JMP     SHORT   ERREXIT
00AA  B0 05                         LENERR: MOV     AL,5              ;Bad drive request struct.length
00AC  EB 06                                 JMP     SHORT   ERREXIT
00AE  B0 07                         MEDERR: MOV     AL,7              ;Unknown Media discovered.
00B0  EB 02                                 JMP     SHORT   ERREXIT
00B2  B0 03                         CMDERR: MOV     AL,3              ;Set unknown command error #.
00B4                                ERREXIT:
00B4  B4 81                                 MOV     AH,10000001B      ;Set error and done bits.
00B6  F9                                    STC                       ;Set Carry bit.
00B7  EB 02                                 JMP     SHORT   EXIT1     ;Quick way out.
                                    ;
00B9                                EXITP   PROC    FAR
                                    ;
00B9  B4 01                         EXIT:   MOV     AH,00000001B      ;Set done bit for MSDOS.
00BB  8B 1E 000E R                  EXIT1:  MOV     BX,WORD PTR PTRSAV
00BF  8E 06 0010 R                          MOV     ES,WORD PTR PTRSAV+2
00C3  26: 89 47 03                          MOV     ES:STATUS[BX],AX  ;Save operation complete and status.
00C7  5B                                    POP     BX                ;Restore registers.
00C8  07                                    POP     ES
00C9  1F                                    POP     DS
00CA  5F                                    POP     DI
00CB  5A                                    POP     DX
00CC  59                                    POP     CX
00CD  58                                    POP     AX
00CE  5E                                    POP     SI
00CF  CB                                    RET                       ;RESTORE REGS AND RETURN
                                    ;
00D0                                EXITP   ENDP
                                    ;
                                    ;    MOVE LENGTH OF DRIVE REQU. STRUCT. TO AL REG.
                                    ;
00D0  8B 1E 000E R                  GETLEN: MOV     BX,WORD PTR PTRSAV
00D4  8E 06 0010 R                          MOV     ES,WORD PTR PTRSAV+2
00D8  26: 8A 07                             MOV     AL,ES:BYTE PTR CMDLEN[BX]
00DB  C3                                    RET
                                    ;
                                    ;
00DC                                MEDIAC:
00DC  E8 00D0 R                             CALL    GETLEN            ;GET DRIVE STRUCT. LENGTH
00DF  3C 0F                                 CMP     AL,15
00E1  72 C7                                 JB      LENERR            ;BAD STRUCTURE LENGTH
00E3  8A 26 0000 E                          MOV     AH,WI_FLAGS       ;GET MEDIA CHECK FLAG
00E7  8A 0E 0351 R                          MOV     CL,BYTE PTR WIPAR ;GET DISK UNIT
00EB  B0 01                                 MOV     AL,01H
00ED  D2 E0                                 SAL     AL,CL             ;SHIFT AL BECAUSE CL=0
00EF  22 C4                                 AND     AL,AH
00F1  75 07                                 JNZ     MEDIAC1
00F3  26: C6 47 0E 01                       MOV     ES:BYTE PTR TRANS[BX],1 ;SET MEDIA NOT CHANGED
00F8  EB BF                                 JMP     EXIT
00FA                                MEDIAC1:
00FA  26: C6 47 0E 00                       MOV     ES:BYTE PTR TRANS[BX],0 ;DON'T KNOW IF MEDIA HAS BEEN CHANGED
00FF  EB B8                                 JMP     EXIT
```

```
                                ;
0101                            GETBPB:
0101  E8 00D0 R                         CALL    GETLEN                  ;GET DRIVE STRUCT. LENGTH
0104  3C 16                             CMP     AL,22
0106  72 44                             JB      GETBPB2                 ;BAD STRUCT. LENGTH
0108  E8 035C R                         CALL    FIXREADY                ;CHECK IF DRIVE READY
010B  75 42                             JNZ     GETBPB3
010D  C6 06 0352 R 10                   MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
0112  C7 06 0353 R 0000                 MOV     WORD PTR WIPAR+2,0      ;SET FAT SECTOR
0118  E8 036F R                         CALL    FIXDR
011B  80 3E 0355 R 50                   CMP     BYTE PTR WIPAR+4,50H    ;CHECK IF READY AND SEEK COMPLETE
0120  75 2D                             JNZ     GETBPB3
0122  06                                PUSH    ES
0123  8A 0E 0351 R                      MOV     CL,BYTE PTR WIPAR
0127  B0 01                             MOV     AL,01H
0129  D2 E0                             SAL     AL,CL
012B  50                                PUSH    AX
012C  C6 06 0352 R 20                   MOV     BYTE PTR WIPAR+1,READ   ;SET READ FUNCTION
0131  E8 036F R                         CALL    FIXDR
0134  8E 06 0357 R                      MOV     ES,WORD PTR WIPAR+6
0138  8B 3E 0359 R                      MOV     DI,WORD PTR WIPAR+8
013C  26: 80 7D 15 F8                   CMP     BYTE PTR ES:21[DI],0F8H
0141  74 11                             JZ      GETBPB6
0143                            GETBPB1:
0143  58                                POP     AX
0144  08 06 0000 E                      OR      WI_FLAGS,AL             ;SET UNKNOWN MEDIA FLAG
0148  07                                POP     ES
0149  E9 00AE R                         JMP     MEDERR                  ;MEDIA ERROR
014C                            GETBPB2:
014C  E9 00AA R                         JMP     LENERR
014F                            GETBPB3:
014F  B0 02                             MOV     AL,2                    ;DRIVE NOT READY
0151  E9 00B4 R                         JMP     ERREXIT
0154                            GETBPB6:
0154  B9 0013                           MOV     CX,19
0157  83 C7 0B                          ADD     DI,11                   ;ADDR. OF BPB IN BOOT RECORD
015A  BE 000F R                         MOV     SI,OFFSET WIBPB1
015D                            GETBPB7:
015D  26: 8A 05                         MOV     AL,BYTE PTR ES:[DI]     ;MOVE BPB FROM BOOT RECORD
0160  88 04                             MOV     BYTE PTR [SI],AL
0162  47                                INC     DI
0163  46                                INC     SI
0164  E2 F7                             LOOP    GETBPB7
0166  BE 000F R                         MOV     SI,OFFSET WIBPB1
0169                            GETBPB8:
0169  58                                POP     AX
016A  F6 D0                             NOT     AL                      ;MEDIA FLAG = 0
016C  20 06 0000 E                      AND     WI_FLAGS,AL             ;SET MEDIA FLAG
0170  07                                POP     ES
0171  26: 89 77 12                      MOV     WORD PTR ES:COUNT [BX],SI
0175  26: 8C 4F 14                      MOV     WORD PTR ES:COUNT+2[BX],CS
0179  E9 00B9 R                         JMP     EXIT
```

```
                               ;
                               :     READ DATA
                               ;
017C                           WIREAD:
017C  83 F9 00                       CMP    CX,0                ;CHECK IF SECTOR COUNTER > 0
017F  74 3B                          JZ     WIREAD1A
0181  E8 00D8 R                      CALL   GETLEN              ;GET DRIVE REQU.STRUCT.LENGTH
0184  3C 16                          CMP    AL,22
0186  72 C4                          JB     GETBPB2             ;BAD STRUCT. LENGTH
0188  E8 0340 R                      CALL   CHKDRIVE            ;CHECK IF UNIT AVAILABLE
018B  72 5F                          JC     WIREAD4             ;UNIT ERROR
018D  E8 035C R                      CALL   FIXREADY            ;CHECK IF DRIVE READY
0190  75 4D                          JNZ    WIREAD3
0192  A1 0249 R                      MOV    AX,WORD PTR RETRYDEF    ;GET NO. OF RETRIES
0195  A3 024B R                      MOV    WORD PTR RETRYC,AX
0198                           WIREAD1:
0198  C6 06 0352 R 20                MOV    BYTE PTR WIPAR+1,READ   ;SET READ FUNCTION
019D  E8 036F R                      CALL   FIXDR               ;READ ONE SECTOR
01A0  A0 0355 R                      MOV    AL,BYTE PTR WIPAR+4     ;GET STATUS
01A3  24 F9                          AND    AL,0F9H
01A5  3C 50                          CMP    AL,50H
01A7  75 16                          JNZ    WIREAD2             ;GO PERFORM RETRIES
01A9  A1 0249 R                      MOV    AX,WORD PTR RETRYDEF
01AC  A3 024B R                      MOV    WORD PTR RETRYC,AX  ;SET RETRY DEFAULT VALUE
01AF  81 06 0359 R 0200              ADD    WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
01B5  FF 06 0353 R                   INC    WORD PTR WIPAR+2    ;SECTOR # +1
01B9  49                             DEC    CX                  ;SECTOR COUNT -1
01BA  75 DC                          JNZ    WIREAD1             ;GO READ NEXT SECTOR
01BC                           WIREAD1A:
01BC  E9 00B9 R                      JMP    EXIT
                               ;
01BF                           WIREAD2:
01BF  24 80                          AND    AL,CBUSY
01C1  75 1C                          JNZ    WIREAD3             ;CHECK DRIVE NOT READY
01C3  FE 0E 024B R                   DEC    BYTE PTR RETRYC
01C7  79 CF                          JNS    WIREAD1             ;PERFORM RETRY
01C9  A0 0249 R                      MOV    AL,BYTE PTR RETRYDEF
01CC  A2 024B R                      MOV    BYTE PTR RETRYC,AL  ;SET RETRY DEFAULT VALUE
01CF  FE 0E 024C R                   DEC    BYTE PTR RETRYC+1
01D3  78 1A                          JS     WIERR               ;CHECK ERROR TYPE
01D5  C6 06 0352 R 10                MOV    BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
01DA  E8 036F R                      CALL   FIXDR               ;RESTORE DRIVE
01DD  EB B9                          JMP    WIREAD1             ;GO READ AGAIN
                               ;
01DF                           WIREAD3:
01DF  E8 00D8 R                      CALL   GETLEN
01E2  26: C7 47 12 0000              MOV    ES:WORD PTR COUNT[BX],00      ;SET NO. OF SECT.PROCESSED
01E8  B0 02                          MOV    AL,2                ;SET DRIVE NOT READY STATUS
01EA  EB 1D                          JMP    SHORT   WIERR2
01EC                           WIREAD4:
01EC  E9 00A6 R                      JMP    UNITERR
```

```
                                ;
                                ;       ERROR ROUTINE
                                ;
01EF                            WIERR:
01EF  E8 00D8 R                         CALL    GETLEN
01F2  26: 29 4F 12                      SUB     ES:WORD PTR COUNT[BX],CX        ;SET NO. OF PROCESSED SECT.
01F6  A0 0355 R                         MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS REG.
01F9  D0 D8                             RCR     AL,1
01FB  72 23                             JC      WIERR8                  ;CONTROLLER ERROR
01FD  B1 04                             MOV     CL,4
01FF  D2 D8                             RCR     AL,CL
0201  73 09                             JNC     WIERR3                  ;SEEK ERROR
0203  D0 D8                             RCR     AL,1
0205  72 09                             JC      WIERR4                  ;WRITE FAULT
0207  B0 0C                     WIERR1: MOV     AL,12                   ;GENERAL FAILURE
0209  E9 00B4 R                 WIERR2: JMP     ERREXIT                 ;STORE ERROR AND EXIT
                                ;
020C  B0 06                     WIERR3: MOV     AL,6                    ;SEEK ERROR
020E  EB F9                             JMP     SHORT   WIERR2
0210  B0 0A                     WIERR4: MOV     AL,10                   ;WRITE FAULT
0212  EB F5                             JMP     SHORT   WIERR2
0214  B0 0B                     WIERR5: MOV     AL,11                   ;READ FAULT
0216  EB F1                             JMP     SHORT   WIERR2
0218  B0 08                     WIERR6: MOV     AL,8                    ;SECTOR NOT FOUND
021A  EB ED                             JMP     SHORT   WIERR2
021C  B0 04                     WIERR7: MOV     AL,4                    ;CRC ERROR
021E  EB E9                             JMP     SHORT   WIERR2
0220  A0 0356 R                 WIERR8: MOV     AL,BYTE PTR WIPAR+5     ;GET ERROR REGISTER
0223  D0 D8                             RCR     AL,1
0225  72 ED                             JC      WIERR5                  ;ADDR. MARK NOT FOUND
0227  D0 D8                             RCR     AL,1
0229  72 DC                             JC      WIERR1                  ;TRACK 0 ERROR
022B  D0 D8                             RCR     AL,1
022D  72 D8                             JC      WIERR1                  ;ABORTED COMMAND
022F  D0 D8                             RCR     AL,1
0231  D0 D8                             RCR     AL,1
0233  72 E3                             JC      WIERR6                  ;ID NOT FOUND
0235  D0 D8                             RCR     AL,1
0237  72 E3                             JC      WIERR7                  ;CRC ERROR IN ID FIELD
0239  D0 D8                             RCR     AL,1
023B  72 D7                             JC      WIERR5                  ;UNCORRECTABLE DATA
023D  D0 D8                             RCR     AL,1
023F  72 D7                             JC      WIERR6                  ;BAD BLOCK DETECTED
0241  EB C4                             JMP     SHORT   WIERR1
                                ;
                                ;
0243                            WIWRTC:
0243  E9 00AA R                         JMP     LENERR
0246                            WIWRTD:
0246  E9 00A6 R                         JMP     UNITERR
                                ;
                                ;
0249  0000                      RETRYDEF        DW      0               ;RETRY DEFAULT VALUE
024B  0000                      RETRYC          DW      0               ;RETRY COUNT
024D  00                        WRTFLG          DB      0               ; 00=WRITE, FF=WRITE/VERIFY
```

```
                            ;
                            ;       WRITE DATA
                            ;
024E                        WIWRT:
024E  C6 06 024D R 00               MOV     BYTE PTR WRTFLG,0       ;SET WRITE DATA FLAG
0253                        WIWRT1:
0253  83 F9 00                      CMP     CX,0                    ;CHECK IF SECTOR COUNTER > 0
0256  74 42                         JZ      WIWRT31
0258  E8 00D0 R                     CALL    GETLEN                  ;GET DRIVE REQU. STRUCT. LENGTH
025B  3C 16                         CMP     AL,22
025D  72 E4                         JB      WIWRTC                  ;BAD STRUCT. LENGTH
025F  E8 0340 R                     CALL    CHKDRIVE                ;CHECK IF UNIT AVAILABLE
0262  72 E2                         JC      WIWRTD                  ;UNIT ERROR
0264  E8 035C R                     CALL    FIXREADY                ;CHECK IF DRIVE READY
0267  75 34                         JNZ     WIWRT3A
0269  A1 0249 R                     MOV     AX,WORD PTR RETRYDEF
026C  A3 024B R                     MOV     WORD PTR RETRYC,AX      ;SET RETRY COUNT
026F                        WIWRT2:
026F  C6 06 0352 R 30               MOV     BYTE PTR WIPAR+1,WRITE  ;SET WRITE FUNCTION
0274  E8 036F R                     CALL    FIXDR                   ;WRITE DATA
0277  A0 0355 R                     MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
027A  24 F9                         AND     AL,0F9H
027C  3C 50                         CMP     AL,50H                  ;CHECK FOR ERROR
027E  75 2E                         JNZ     WIWRT4                  ;PERFORM RETRIES
0280  A1 0249 R                     MOV     AX,WORD PTR RETRYDEF
0283  A3 024B R                     MOV     WORD PTR RETRYC,AX      ;SET RETRY DEFAULT VALUES
0286  80 3E 024D R 00               CMP     BYTE PTR WRTFLG,0
028B  75 41                         JNZ     WIWRT5                  ;GO VERIFY DATA
028D                        WIWRT3:
028D  81 06 0359 R 0200             ADD     WORD PTR WIPAR+8,0200H  ;BUFFER ADDR. +200H
0293  FF 06 0353 R                  INC     WORD PTR WIPAR+2        ;SECTOR NUMBER +1
0297  49                            DEC     CX                      ;SECTOR COUNT -1
0298  75 D5                         JNZ     WIWRT2                  ;GO WRITE NEXT SECTOR
029A                        WIWRT31:
029A  E9 00B9 R                     JMP     EXIT
029D                        WIWRT3A:
029D  E8 00D0 R                     CALL    GETLEN
02A0  26: C7 47 12 0000             MOV     ES:WORD PTR COUNT[BX],00        ;SET NO. OF PROCESSED SECT.
02A6  B0 02                         MOV     AL,2                    ;DRIVE NOT READY
02A8  E9 00B4 R                     JMP     ERREXIT
02AB                        WIWRT3B:
02AB  E9 01EF R                     JMP     WIERR                   ;CHECK STATUS TYPE
02AE                        WIWRT4:
02AE  24 80                         AND     AL,CBUSY                ;CHECK IF DRIVE NOT READY
02B0  75 EB                         JNZ     WIWRT3A
02B2  FE 0E 024B R                  DEC     BYTE PTR RETRYC
02B6  79 B7                         JNS     WIWRT2                  ;WRITE AGAIN
02B8  A0 0249 R                     MOV     AL,BYTE PTR RETRYDEF
02BB  A2 024B R                     MOV     BYTE PTR RETRYC,AL      ;SET RETRY DEFAULT VALUE
02BE  FE 0E 024C R                  DEC     BYTE PTR RETRYC+1
02C2  78 E7                         JS      WIWRT3B                 ;WRITE ERROR
02C4  C6 06 0352 R 10               MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
02C9  E8 036F R                     CALL    FIXDR                   ;RESTORE DRIVE
02CC  EB A1                         JMP     WIWRT2                  ;WRITE NEXT SECTOR
02CE                        WIWRT5:
02CE  8E 06 0357 R                  MOV     ES,WORD PTR WIPAR+6
02D2  8B 3E 0359 R                  MOV     DI,WORD PTR WIPAR+8     ;SAVE DATA BUFFER ADDR.
```

```
02D6                      WIWRT6:
02D6  A0 035B R                  MOV     AL,BYTE PTR WIPAR+10    ;GET ACTUAL SDH REG. CONTENTS
02D9  E6 C6                      OUT     SDH,AL
02DB  B0 20                      MOV     AL,READ
02DD  E6 C7                      OUT     COMND,AL                ;OUTPUT READ FUNCTION
02DF  E8 03F4 R                  CALL    WAIT
02E2  E4 C6                      IN      AL,SDH
02E4  0C 18                      OR      AL,18H
02E6  E6 C6                      OUT     SDH,AL                  ;CLEAR DRIVE LAMP
02E8  51                         PUSH    CX
02E9  B9 0200                    MOV     CX,512
02EC                      WIWRT6A:
02EC  E4 C0                      IN      AL,DATA                 ;GET READ DATA
02EE  AE                         SCASB                           ;COMPARE WITH DATA WRITTEN
02EF  E1 FB                      LOOPZ   WIWRT6A
02F1  75 25                      JNZ     WIWRT7
02F3                      WIWRT6B:
02F3  59                         POP     CX
02F4  A0 0355 R                  MOV     AL,BYTE PTR WIPAR+4     ;GET STATUS
02F7  24 F9                      AND     AL,0F9H
02F9  3C 50                      CMP     AL,50H
02FB  74 2B                      JZ      WIWRT8
02FD  FE 0E 024B R               DEC     BYTE PTR RETRYC
0301  75 D3                      JNZ     WIWRT6                  ;PERFORM RETRY
0303  C6 06 024B R 05            MOV     BYTE PTR RETRYC,5
0308  FE 0E 024C R               DEC     BYTE PTR RETRYC+1
030C  74 27                      JZ      WIWRTB                  ;READ ERROR
030E  C6 06 0352 R 10            MOV     BYTE PTR WIPAR+1,REST   ;SET RESTORE FUNCTION
0313  E8 036F R                  CALL    FIXDR                   ;RESTORE DRIVE
0316  EB BE                      JMP     WIWRT6
0318                      WIWRT7:
0318  83 F9 00                   CMP     CX,0
031B  74 04                      JZ      WIWRT7B
031D                      WIWRT7A:
031D  E4 C0                      IN      AL,DATA
031F  E2 FC                      LOOP    WIWRT7A
0321                      WIWRT7B:
0321  80 0E 0355 R 20            OR      BYTE PTR WIPAR+4,DWRFA  ;SET WRITE FAULT ERROR
0326  EB CB                      JMP     SHORT  WIWRT6B
0328                      WIWRT8:
0328  C7 06 024B R 0505          MOV     WORD PTR RETRYC,0505H   ;SET RETRY DEFAULT VALUE
032E  E9 028D R                  JMP     WIWRT3
0331                      WIWRTA:
0331  59                         POP     CX
0332  E9 0210 R                  JMP     WIERR4                  ;DATA VERIFY ERROR
0335                      WIWRTB:
0335  E9 01EF R                  JMP     WIERR                   ;SET ERROR STATUS
                          ;
                          ;      WRITE DATA AND VERIFY
                          ;
0338                      WIWRTV:
0338  C6 06 024D R FF            MOV     BYTE PTR WRTFLG,0FFH    ;SET WRITE/VERIFY FLAG
033D  E9 0253 R                  JMP     WIWRT1
```

```
                                        ;
                                        ;     ROUTINE TO CHECK IF UNIT AVAILABLE
                                        ;        EXIT: CARRY ON=UNIT ERROR
                                        ;
0340                                    CHKDRIVE:
0340  50                                      PUSH    AX
0341  A0 0351 R                               MOV     AL,BYTE PTR WIPAR       ;GET UNIT NO. TO WORK WITH
0344  8A 26 000A E                            MOV     AH,BYTE PTR WINDEV+10   ;GET NO.OF UNITS IN SYSTEM
0348  3A E0                                   CMP     AH,AL
034A  58                                      POP     AX
034B  76 02                                   JBE     CHKDRIVE1
034D  F8                                      CLC
034E  C3                                      RET
034F                                    CHKDRIVE1:
034F  F9                                      STC
0350  C3                                      RET
```

```
                        ;
                        ;*********************************************
                        ;*                                           *
                        ;*      PERIPHERAL INTERFACE MODULE (PIM)    *
                        ;*                                           *
                        ;*             WINCHESTER DISK               *
                        ;*                                           *
                        ;*********************************************
                        ;
                        ;UNIT 0= HEAD 0 AND 1
                        ;UNIT 1= HEAD 2 AND 3
                        ;
                        ;
                        ;         WINCHESTER DISK PARAMETER BLOCK
                        ;         ===============================
                        ;
                        ;
0351  00                WIPAR   DB      0               ; WIPAR + 0     DISK UNIT
0352  10                        DB      REST            ; WIPAR + 1     FUNCTION
0353  0000                      DW      0               ; WIPAR + 2     SECTOR LO
                                                        ; WIPAR + 3     SECTOR HI
0355  00                        DB      0               ; WIPAR + 4     STATUS 1
0356  00                        DB      0               ; WIPAR + 5     STATUS 2
0357  0000                      DW      0               ; WIPAR + 6     BUFFER ADDR.(SEGMENT)
0359  0000                      DW      0               ; WIPAR + 8     BUFFER ADDR.(OFFSET)
035B  00                        DB      0               ; WIPAR + 10    ACTUAL SDH REG. CONTENTS
                        ;
                        ;
                        ;
                        ;****************************************
                        ;*                                      *
                        ;*  CHECK IF WINCHESTER DRIVE IS        *
                        ;*   CONNECTED AND POWERED ON.          *
                        ;*                                      *
                        ;*   EXIT: ZERO FLAG ON = DRIVE READY   *
                        ;*                                      *
                        ;****************************************
                        ;
035C                    FIXREADY:
035C  B0 55                     MOV     AL,55H
035E  E6 C4                     OUT     CYLLO,AL                ;OUTPUT PATTERN TO R/W PORT
0360  B0 AA                     MOV     AL,0AAH
0362  E6 C3                     OUT     SECNO,AL
0364  E4 C4                     IN      AL,CYLLO                ;READ PATTERN BACK AND COMPARE
0366  3C 55                     CMP     AL,55H
0368  75 04                     JNZ     FIXREADY1
036A  E4 C3                     IN      AL,SECNO
036C  3C AA                     CMP     AL,0AAH
036E                    FIXREADY1:
036E  C3                        RET
```

```
                                ;*******************************************
                                ;*                                         *
                                ;*    WINCHESTER DISK DRIVER               *
                                ;*                                         *
                                ;*     ENTRY: PARAMETER BLOCK FILLED UP    *
                                ;*     EXIT:  STATUS BYTES IN PARAM.       *
                                ;*            BLOCK UPDATED AND ALL        *
                                ;*            REGISTERS SAVED.             *
                                ;*******************************************
                                ;
036F  50                        FIXDR:  PUSH    AX
0370  53                                PUSH    BX
0371  51                                PUSH    CX
0372  52                                PUSH    DX
0373  A1 0353 R                         MOV     AX,WORD PTR WIPAR+2     ;GET LOGIC SECTOR NUMBER
0376  B9 0011                           MOV     CX,17
0379  BA 0000                           MOV     DX,0
037C  F7 F1                             DIV     CX                      ;CALCULATE CYL/HEAD
037E  50                                PUSH    AX
037F  8A C2                             MOV     AL,DL
0381  E6 C3                             OUT     SECNO,AL                ;SET SECTOR NUMBER
0383  8A 1E 0351 R                      MOV     BL,BYTE PTR WIPAR       ;GET DISK UNIT #
0387  D0 E3                             SHL     BL,1
0389  D0 E3                             SHL     BL,1
038B  D0 E3                             SHL     BL,1                    ;DRIVE SELECT
038D  58                                POP     AX
038E  50                                PUSH    AX
038F  24 03                             AND     AL,03H                  ;GET HEAD SELECT
0391  0A C3                             OR      AL,BL
0393  0C A0                             OR      AL,SDHREG               ;ECC/CRC AND BYTES PER SECTOR
0395  E6 C6                             OUT     SDH,AL                  ;SET ECC/CRC-BYTES/SECT-DRIVE-HEAD
0397  A2 035B R                         MOV     BYTE PTR WIPAR+10,AL    ;SAVE ACTUAL SDH REG. CONTENTS
039A  58                                POP     AX
039B  D1 E8                             SHR     AX,1
039D  D1 E8                             SHR     AX,1
039F  E6 C4                             OUT     CYLLO,AL                ;SET CYLINDER LOW
03A1  80 E4 03                          AND     AH,03H
03A4  8A C4                             MOV     AL,AH
03A6  E6 C5                             OUT     CYLHI,AL                ;SET CYLINDER HIGH
03A8  E4 C7                             IN      AL,STAT                 ;GET DISK STATUS
03AA  A2 0355 R                         MOV     BYTE PTR WIPAR+4,AL
03AD  24 80                             AND     AL,CBUSY                ;CHECK IF CONTROLLER BUSY
03AF  75 16                             JNZ     FIXD3
03B1  A0 0352 R                         MOV     AL,BYTE PTR WIPAR+1
03B4  E6 C7                             OUT     COMND,AL                ;SET FUNCTION
03B6  24 F0                             AND     AL,0F0H
03B8  3C 20                             CMP     AL,READ
03BA  74 16                             JZ      RD                      ;GO READ DATA
03BC  3C 30                             CMP     AL,WRITE
03BE  74 4E                             JZ      WR                      ;GO WRITE DATA
03C0  3C 50                             CMP     AL,FORMAT
03C2  74 46                             JZ      WR0                     ;GO FORMAT ONE TRACK
03C4  EB 5C 90                          JMP     WR2                     ;SEEK OR RESTORE
03C7  E4 C6                     FIXD3:  IN      AL,SDH
03C9  0C 10                             OR      AL,10H
03CB  E6 C6                             OUT     SDH,AL                  ;CLEAR DISK LAMP
03CD  5A                                POP     DX
03CE  59                                POP     CX
03CF  5B                                POP     BX
03D0  58                                POP     AX
03D1  C3                                RET
```

```
                        ;
                        ;       ****************************
                        ;       *    READ ROUTINE          *
                        ;       ****************************
                        ;
03D2  E8 03F4 R         RD:     CALL    WAIT                    ;WAIT UNTIL READ COMPLETE
03D5  1E                        PUSH    DS
03D6  8B 1E 0359 R              MOV     BX,WORD PTR WIPAR+8     ;GET OFFSET
03DA  8E 1E 0357 R              MOV     DS,WORD PTR WIPAR+6     ;GET SEGMENT ADDR.
03DE  B9 0200                   MOV     CX,512                  ;INPUT COUNT
03E1  E4 C0             RD2:    IN      AL,DATA                 ;INPUT DATA
03E3  88 07                     MOV     BYTE PTR[BX],AL         ;SAVE INPUT
03E5  43                        INC     BX
03E6  E0 F9                     LOOPNZ  RD2             ;CONTINUE UNTIL ALL BYTES IN BUFFER
                                                        ;BUT STOP BEFORE BUFFER ADDR. WRAP AROUND
03E8  83 F9 00                  CMP     CX,0
03EB  74 04                     JZ      RD4
03ED  E4 C0             RD3:    IN      AL,DATA                 ;CLEAR CONTROLLER BUFFER
03EF  E2 FC                     LOOP    RD3
03F1  1F                RD4:    POP     DS
03F2  EB D3                     JMP     SHORT   FIXD3
                        ;
                        ;
                        ;       ****************************
                        ;       *    WAIT ROUTINE          *
                        ;       ****************************
                        ;
03F4  E4 C7             WAIT:   IN      AL,STAT                 ;GET STATUS
03F6  24 80                     AND     AL,CBUSY
03F8  75 FA                     JNZ     WAIT                    ;LOOP UNTIL DISK READY
03FA  E4 C7                     IN      AL,STAT
03FC  A2 0355 R                 MOV     BYTE PTR WIPAR+4,AL     ;SAVE STATUS
03FF  D0 D8                     RCR     AL,1
0401  72 01                     JC      ER1                     ;JUMP IF ERROR CONDITION
0403  C3                        RET
                        ;
0404  E4 C1             ER1:    IN      AL,ERROR                ;GET ERROR STATUS
0406  A2 0356 R                 MOV     BYTE PTR WIPAR+5,AL     ;SAVE STATUS
0409  C3                        RET
                        ;
                        ;       ****************************
                        ;       *    WRITE ROUTINE         *
                        ;       ****************************
                        ;
040A  B0 11             WR0:    MOV     AL,17
040C  E6 C2                     OUT     SECNT,AL                ;SET SECT COUNT FOR FORMAT
                        ;
040E  1E                WR:     PUSH    DS
040F  8B 1E 0359 R              MOV     BX,WORD PTR WIPAR+8     ;BUFFER ADDR.(OFFSET)
0413  8E 1E 0357 R              MOV     DS,WORD PTR WIPAR+6     ;BUFFER ADDR.(SEGMENT)
0417  B9 0200                   MOV     CX,512                  ;INPUT COUNT
041A  8A 07             WR1:    MOV     AL,BYTE PTR[BX]         ;GET BYTE FROM BUFFER
041C  E6 C0                     OUT     DATA,AL                 ;OUTPUT DATA
041E  43                        INC     BX
041F  E2 F9                     LOOP    WR1
0421  1F                        POP     DS
0422  E8 03F4 R         WR2:    CALL    WAIT                    ;WAIT UNTIL FUNCT. COMPLETE.
0425  EB A8                     JMP     SHORT   FIXD3
```

```
                                   ;
0427                               WIINIT:
0427  E8 00D8 R                         CALL    GETLEN                  ;GET DRIVE REQU. STRUCT.LENGTH
042A  3C 16                             CMP     AL,22
042C  72 58                             JB      WIINIT1                 ;BAD STRUCT. LENGTH
042E  A0 0000 E                         MOV     AL,WI_FLAGS
0431  0C 7F                             OR      AL,7FH                  ;SET UNKNOWN DISK
0433  A2 0000 E                         MOV     WI_FLAGS,AL
0436  A0 0000 E                         MOV     AL,WINCH_DRIVES         ;GET NO. OF DRIVES ON SYSTEM
0439  8A 0E 0000 E                      MOV     CL,WI_OUT_RETRIES       ;GET RETRY (OUT) COUNTS
043D  8A 2E 0000 E                      MOV     CH,WI_IN_RETRIES        ;GET RETRY (IN) COUNTS
0441  88 2E 0249 R                      MOV     BYTE PTR RETRYDEF,CH
0445  88 0E 024A R                      MOV     BYTE PTR RETRYDEF+1,CL  ;SAVE RETRY COUNTS
0449  A2 000A E                         MOV     BYTE PTR WINDEV+10,AL   ;SET NUMBER OF UNITS
044C  26: 88 47 0D                      MOV     ES:BYTE PTR MEDIA[BX],AL
0450  26: C7 47 0E 0427 R               MOV     ES:WORD PTR TRANS[BX],OFFSET WIINIT
0456  26: 8C 4F 10                      MOV     ES:WORD PTR TRANS+2[BX],CS
045A  26: C7 47 12 0087 R               MOV     ES:WORD PTR COUNT[BX],OFFSET WIBPB      ;ADDR. OF BPB ARRAY
0460  26: 8C 4F 14                      MOV     ES:WORD PTR COUNT+2[BX],CS
0464  C7 06 006F R 00B9 R               MOV     WORD PTR WITBL,OFFSET EXIT
046A  E8 035C R                         CALL    FIXREADY
046D  75 1F                             JNZ     WIINIT4
046F  C6 06 0352 R 10                   MOV     BYTE PTR WIPAR+1,REST
0474  80 26 0351 R 01                   AND     BYTE PTR WIPAR,01
0479  E8 036F R                         CALL    FIXDR                   ;RESTORE DRIVE
047C  80 3E 0355 R 50                   CMP     BYTE PTR WIPAR+4,50H
0481  75 06                             JNZ     WIINIT2                 ;ERROR CONDITION
0483  E9 00B9 R                         JMP     EXIT
0486                               WIINIT1:
0486  E9 00AA R                         JMP     LENERR
0489                               WIINIT2:
0489  A0 0355 R                         MOV     AL,BYTE PTR WIPAR+4
048C  24 80                             AND     AL,CBUSY
048E                               WIINIT4:
048E  B0 02                             MOV     AL,2                    ;DRIVE NOT READY
0490  75 02                             JNZ     WIINIT3
0492  B0 0C                             MOV     AL,12                   ;GENERAL FAILURE
0494                               WIINIT3:
0494  E9 00B4 R                         JMP     ERREXIT                 ;SAVE STATUS AND EXIT
                                   ;
                                   ;
0497                               WI_END:
                                   ;
0497                               CSEG   ENDS
                                   ;
                                   ;
                                   END     BEGIN
```

```
Segments and groups:

                N a m e             Size   align  combine class

CSEG . . . . . . . . . . . . . .    0497   PARA   PUBLIC  'CODE'

Symbols:

                N a m e             Type    Value  Attr

ABC. . . . . . . . . . . . . . .    Number  0004
BBD. . . . . . . . . . . . . . .    Number  0080
BEGIN. . . . . . . . . . . . . .    L NEAR  0000   CSEG
BUSEXIT. . . . . . . . . . . . .    L NEAR  00A2   CSEG
CBUSY. . . . . . . . . . . . . .    Number  0080
CDRQ . . . . . . . . . . . . . .    Number  0008
CERR . . . . . . . . . . . . . .    Number  0001
CHKDRIVE . . . . . . . . . . . .    L NEAR  0340   CSEG
CHKDRIVE1. . . . . . . . . . . .    L NEAR  034F   CSEG
CMD. . . . . . . . . . . . . . .    Number  0002
CMDERR . . . . . . . . . . . . .    L NEAR  00B2   CSEG
CMDLEN . . . . . . . . . . . . .    Number  0000
COMND. . . . . . . . . . . . . .    Number  00C7
CORRD. . . . . . . . . . . . . .    Number  0004
COUNT. . . . . . . . . . . . . .    Number  0012
CRCID. . . . . . . . . . . . . .    Number  0020
CYLHI. . . . . . . . . . . . . .    Number  00C5
CYLLO. . . . . . . . . . . . . .    Number  00C4
DAMNFD . . . . . . . . . . . . .    Number  0001
DATA . . . . . . . . . . . . . .    Alias   HBASE
DREADY . . . . . . . . . . . . .    Number  0040
DSEEC. . . . . . . . . . . . . .    Number  0010
DWRFA. . . . . . . . . . . . . .    Number  0020
ENTRY. . . . . . . . . . . . . .    L NEAR  0016   CSEG
ENTRY1 . . . . . . . . . . . . .    L NEAR  003E   CSEG
ER1. . . . . . . . . . . . . . .    L NEAR  0404   CSEG
ERREXIT. . . . . . . . . . . . .    L NEAR  00B4   CSEG
ERROR. . . . . . . . . . . . . .    Number  00C1
EXIT . . . . . . . . . . . . . .    L NEAR  00B9   CSEG
EXIT1. . . . . . . . . . . . . .    L NEAR  00BB   CSEG
EXITP. . . . . . . . . . . . . .    F PROC  00B9   CSEG    Length =0017
FIXD3. . . . . . . . . . . . . .    L NEAR  03C7   CSEG
FIXDR. . . . . . . . . . . . . .    L NEAR  036F   CSEG
FIXREADY . . . . . . . . . . . .    L NEAR  035C   CSEG
FIXREADY1. . . . . . . . . . . .    L NEAR  036E   CSEG
FORMAT . . . . . . . . . . . . .    Number  0050
GETBPB . . . . . . . . . . . . .    L NEAR  0101   CSEG
GETBPB1. . . . . . . . . . . . .    L NEAR  0143   CSEG
GETBPB2. . . . . . . . . . . . .    L NEAR  014C   CSEG
GETBPB3. . . . . . . . . . . . .    L NEAR  014F   CSEG
GETBPB6. . . . . . . . . . . . .    L NEAR  0154   CSEG
GETBPB7. . . . . . . . . . . . .    L NEAR  015D   CSEG
GETBPB8. . . . . . . . . . . . .    L NEAR  0169   CSEG
GETLEN . . . . . . . . . . . . .    L NEAR  00D0   CSEG
HBASE. . . . . . . . . . . . . .    Number  00C0
IDNFD. . . . . . . . . . . . . .    Number  0010
LENERR . . . . . . . . . . . . .    L NEAR  00AA   CSEG
MEDERR . . . . . . . . . . . . .    L NEAR  00AE   CSEG
MEDIA. . . . . . . . . . . . . .    Number  000D
MEDIAC . . . . . . . . . . . . .    L NEAR  00DC   CSEG
```

```
MEDIAC1. . . . . . . . . . . . . .      L NEAR  00FA    CSEG
PTRSAV . . . . . . . . . . . . . .      L WORD  000E    CSEG
RD . . . . . . . . . . . . . . . .      L NEAR  03D2    CSEG
RD2. . . . . . . . . . . . . . . .      L NEAR  03E1    CSEG
RD3. . . . . . . . . . . . . . . .      L NEAR  03ED    CSEG
RD4. . . . . . . . . . . . . . . .      L NEAR  03F1    CSEG
READ . . . . . . . . . . . . . . .      Number  0020
REST . . . . . . . . . . . . . . .      Number  0010
RETRYC . . . . . . . . . . . . . .      L WORD  024B    CSEG
RETRYDEF . . . . . . . . . . . . .      L WORD  0249    CSEG
SDH. . . . . . . . . . . . . . . .      Number  00C6
SDHREG . . . . . . . . . . . . . .      Number  00A0
SECNO. . . . . . . . . . . . . . .      Number  00C3
SECNT. . . . . . . . . . . . . . .      Number  00C2
SEEK . . . . . . . . . . . . . . .      Number  0070
START. . . . . . . . . . . . . . .      Number  0014
STAT . . . . . . . . . . . . . . .      Number  00C7
STATUS . . . . . . . . . . . . . .      Number  0003
STRATE . . . . . . . . . . . . . .      Number  0000
STRATP . . . . . . . . . . . . . .      F PROC  0003    CSEG    Length =000B
TR0. . . . . . . . . . . . . . . .      Number  0002
TRANS. . . . . . . . . . . . . . .      Number  000E
UNCOR. . . . . . . . . . . . . . .      Number  0040
UNIT . . . . . . . . . . . . . . .      Number  0001
UNITERR. . . . . . . . . . . . . .      L NEAR  00A6    CSEG
WAIT . . . . . . . . . . . . . . .      L NEAR  03F4    CSEG
WIBPB. . . . . . . . . . . . . . .      L WORD  0007    CSEG
WIBPBI . . . . . . . . . . . . . .      L WORD  000F    CSEG
WIERR. . . . . . . . . . . . . . .      L NEAR  01EF    CSEG
WIERR1 . . . . . . . . . . . . . .      L NEAR  0207    CSEG
WIERR2 . . . . . . . . . . . . . .      L NEAR  0209    CSEG
WIERR3 . . . . . . . . . . . . . .      L NEAR  020C    CSEG
WIERR4 . . . . . . . . . . . . . .      L NEAR  0210    CSEG
WIERR5 . . . . . . . . . . . . . .      L NEAR  0214    CSEG
WIERR6 . . . . . . . . . . . . . .      L NEAR  0218    CSEG
WIERR7 . . . . . . . . . . . . . .      L NEAR  021C    CSEG
WIERR8 . . . . . . . . . . . . . .      L NEAR  0220    CSEG
WIINIT . . . . . . . . . . . . . .      L NEAR  0427    CSEG
WIINIT1. . . . . . . . . . . . . .      L NEAR  0486    CSEG
WIINIT2. . . . . . . . . . . . . .      L NEAR  0489    CSEG
WIINIT3. . . . . . . . . . . . . .      L NEAR  0494    CSEG
WIINIT4. . . . . . . . . . . . . .      L NEAR  049E    CSEG
WINCH_DRIVES . . . . . . . . . . .      V BYTE  0000            External
WINDEV . . . . . . . . . . . . . .      V WORD  0000            External
WIPAR. . . . . . . . . . . . . . .      L BYTE  0351    CSEG
WIREAD . . . . . . . . . . . . . .      L NEAR  017C    CSEG
WIREAD1. . . . . . . . . . . . . .      L NEAR  0198    CSEG
WIREAD1A . . . . . . . . . . . . .      L NEAR  01BC    CSEG
WIREAD2. . . . . . . . . . . . . .      L NEAR  01BF    CSEG
WIREAD3. . . . . . . . . . . . . .      L NEAR  01DF    CSEG
WIREAD4. . . . . . . . . . . . . .      L NEAR  01EC    CSEG
WITBL. . . . . . . . . . . . . . .      L WORD  006F    CSEG
WIWRT. . . . . . . . . . . . . . .      L NEAR  024E    CSEG
WIWRT1 . . . . . . . . . . . . . .      L NEAR  0253    CSEG
WIWRT2 . . . . . . . . . . . . . .      L NEAR  026F    CSEG
WIWRT3 . . . . . . . . . . . . . .      L NEAR  028D    CSEG
WIWRT31. . . . . . . . . . . . . .      L NEAR  029A    CSEG
WIWRT3A. . . . . . . . . . . . . .      L NEAR  029D    CSEG
WIWRT3B. . . . . . . . . . . . . .      L NEAR  02AB    CSEG
WIWRT4 . . . . . . . . . . . . . .      L NEAR  02AE    CSEG
WIWRT5 . . . . . . . . . . . . . .      L NEAR  02CE    CSEG
```

```
WIWRT6 . . . . . . . . . . . . . .      L NEAR  02D6    CSEG
WIWRT6A. . . . . . . . . . . . . .      L NEAR  02EC    CSEG
WIWRT6B. . . . . . . . . . . . . .      L NEAR  02F3    CSEG
WIWRT7 . . . . . . . . . . . . . .      L NEAR  0318    CSEG
WIWRT7A. . . . . . . . . . . . . .      L NEAR  031D    CSEG
WIWRT7B. . . . . . . . . . . . . .      L NEAR  0321    CSEG
WIWRT8 . . . . . . . . . . . . . .      L NEAR  0328    CSEG
WIWRTA . . . . . . . . . . . . . .      L NEAR  0331    CSEG
WIWRTB . . . . . . . . . . . . . .      L NEAR  0335    CSEG
WIWRTC . . . . . . . . . . . . . .      L NEAR  0243    CSEG
WIWRTD . . . . . . . . . . . . . .      L NEAR  0246    CSEG
WIWRTV . . . . . . . . . . . . . .      L NEAR  0338    CSEG
WI_END . . . . . . . . . . . . . .      L NEAR  0497    CSEG    Global
WI_FLAGS . . . . . . . . . . . . .      V BYTE  0000            External
WI_INTERRUPT . . . . . . . . . . .      L NEAR  0012    CSEG    Global
WI_INTERRUPT1. . . . . . . . . . .      L NEAR  006C    CSEG
WI_IN_RETRIES. . . . . . . . . . .      V BYTE  0000            External
WI_OUT_RETRIES . . . . . . . . . .      V BYTE  0000            External
WI_START . . . . . . . . . . . . .      L NEAR  0000    CSEG    Global
WI_STRATEGY. . . . . . . . . . . .      L NEAR  0003    CSEG    Global
WPC. . . . . . . . . . . . . . . .      Number  00C1
WR . . . . . . . . . . . . . . . .      L NEAR  040E    CSEG
WR0. . . . . . . . . . . . . . . .      L NEAR  040A    CSEG
WR1. . . . . . . . . . . . . . . .      L NEAR  041A    CSEG
WR2. . . . . . . . . . . . . . . .      L NEAR  0422    CSEG
WRITE. . . . . . . . . . . . . . .      Number  0038
WRTFLG . . . . . . . . . . . . . .      L BYTE  024D    CSEG
```

BIOS LINK SEQUENCE

The BIOS modules are linked in the following order:

```
IOBASE
KBDCRT
COMDRV
LPDRV
TIMDRV
DSKDRV
WIDRV10 or WIDRV5
BASINIT
SYSINIT
SYSIMES
```

Despite their different lengths of code, WIDRV5 and WIDRV10 occupy the same memory region.

LINK MAP OF IO.SYS

```
BASE SEGMENT : 40H (LOCATED AT ABSOLUTE ADDRESS 400H)

Warning: No STACK segment

Start   Stop    Length  Name                    Class

00000H  041C8H  41C9H   CSEG                    CODE
041D0H  049D4H  0805H   SYSINITSEG              SYSTEM_INIT

Origin          Group
Address         Publics by Name

0000:0488       AUXDEV
0000:0046       AUXTBL
0000:10D7       BACKSP
041D:07D8       BADCOM
041D:07EC       BADCOUNTRY
041D:07A2       BADLD_POST
041D:07C6       BADLD_PRE
041D:077E       BADOPM
041D:07A2       BADSIZ_POST
041D:07A5       BADSIZ_PRE
0000:0BC3       BELL
0000:1485       BLEOS
041D:0012       BUFFERS
0000:01EF       BUF_A
0000:01F1       BUF_E
0000:119E       CARRET
0000:0223       CLEAR_1
0000:0233       CLEAR_2
0000:023D       CMDTABL
0000:04BE       COM1DEV
0000:04D0       COM2DEV
0000:0051       COM2TBL
0000:04E2       COM3DEV
0000:005C       COM3TBL
0000:04F4       COM4DEV
0000:0067       COM4TBL
0000:0506       COM5DEV
0000:0072       COM5TBL
0000:0046       COMM_TBL
0000:0266       COMTBL
0000:194E       COM_INT1
0000:1954       COM_INT2
0000:195A       COM_INT3
0000:1960       COM_INT4
0000:1966       COM_INT5
0000:0476       CONDEV
0000:0023       CONFIG_FLAGS
0000:001E       CONSOLE_FLAGS
0000:08F5       CON_INT
041D:07A2       CRLFM
0000:0376       CRTACTTBL
0000:001D       CRT_ROWS
0000:1026       CUB
0000:0FF3       CUD
```

```
Origin          Group
Address         Publics by Name

0000:1012       CUF
0000:1038       CUP
041D:0005       CURRENT_DOS_LOCATION
0000:001F       CURSOR_TYPE
0000:0FE0       CUU
0000:15AB       DCHR
0000:0227       DEC_SIGN_1
0000:0237       DEC_SIGN_2
0000:035F       DEC_S_1
0000:036F       DEC_S_2
041D:0011       DEFAULT_DRIVE
0000:0D56       DEFFK
0000:1507       DELLIN
041D:000B       DEVICE_LIST
0000:0476       DEVSTART
0000:3441       DREND
0000:0572       DSKDEV
0000:2E4C       DSK_INT
0000:3DB5       DUMMY_END
0000:3BF0       DUMMY_START
0000:1430       ED
0000:1457       EL
0000:02A8       ESCTBL
0000:02A6       ETBLENT
041D:0013       FILES
041D:0009       FINAL_DOS_LOCATION
0000:0013       FLOPPY_DRIVES
0000:0011       FLTAB
0000:0025       FL_FLAGS
0000:0015       FL_IN_RETRIES
0000:0014       FL_OUT_RETRIES
0000:0610       FUNCTBL
0000:002B       GW_COMM
0000:0EB1       H2
0000:0373       HEB_SW
0000:3FB0       HWINIT
0000:0CC4       I16_HANDLER
0000:0E37       I29_HANDLER
0000:1557       ICHR
0000:019F       INP_BUF
0000:14BC       INSLIN
0000:05A6       INT_TRAP
0000:0355       KBDTBL
0000:0C4B       KBD_BUFF_IN
0000:20D2       KBD_CC
0000:09F0       KBD_FL
0000:02D5       KBD_RDEF_TBL
0000:021D       KBD_TT
0000:0189       LASTPRM
0000:0089       LINBUF
0000:1108       LINEFD
0000:0518       LPT1DEV
0000:1E66       LPT1_ACT
0000:052A       LPT2DEV
0000:0034       LPT2TBL
0000:1E70       LPT2_ACT
```

```
Origin           Group
Address          Publics by Name

0000:053C        LPT3DEV
0000:0038        LPT3TBL
0000:1E7A        LPT3_ACT
0000:054E        LPT4DEV
0000:003C        LPT4TBL
0000:1E84        LPT4_ACT
0000:0560        LPT5DEV
0000:0040        LPT5TBL
0000:1E8E        LPT5_ACT
0000:001A        M1RS232
0000:001B        M2RS232
041D:000F        MEMORY_SIZE
0000:0083        MEMORY_SIZEK
0000:05A1        MEM_SIZE_DET
0000:1188        NDFS
0000:0089        PARMS
0000:0BE9        PLAY_MUSIC
0000:109E        POSIT
0000:1052        PRCP
0000:0019        PRINTER_IF_TYPE
0000:0030        PRINTER_TBL
0000:049A        PRNDEV
0000:0030        PRNTBL
0000:104B        PSCP
0000:0085        PTRSAV
0000:001C        PVRS232
0000:13B3        REVERSE
0000:0596        RE_INIT
0000:1375        RHALF_INT
0000:1170        RLF
0000:1DBE        RS232_INT
0000:1223        SGR
0000:1316        SHALF_INT
0000:0EFE        ST12
0000:0602        STACKTOP
041D:0000        SYSINIT
041D:0805        SYSSIZE
0000:217B        TIM2_ISR
0000:04AC        TIMDEV
0000:219A        TIMER_RET
0000:2141        TIM_INT
0000:11A6        VHOME
0000:0016        WINCH_DRIVES
0000:0584        WINDEV
0000:38E7        WI_END
0000:0028        WI_FLAGS
0000:3462        WI_INTERRUPT
0000:0018        WI_IN_RETRIES
0000:0017        WI_OUT_RETRIES
0000:3450        WI_START
0000:3453        WI_STRATEGY
0000:105B        XDSR
```

```
Origin          Group
Address         Publics by Value

0000:0011       FLTAB
0000:0013       FLOPPY_DRIVES
0000:0014       FL_OUT_RETRIES
0000:0015       FL_IN_RETRIES
0000:0016       WINCH_DRIVES
0000:0017       WI_OUT_RETRIES
0000:0018       WI_IN_RETRIES
0000:0019       PRINTER_IF_TYPE
0000:001A       M1RS232
0000:001B       M2RS232
0000:001C       PVRS232
0000:001D       CRT_ROWS
0000:001E       CONSOLE_FLAGS
0000:001F       CURSOR_TYPE
0000:0023       CONFIG_FLAGS
0000:0025       FL_FLAGS
0000:0028       WI_FLAGS
0000:002B       GW_COMM
0000:0030       PRINTER_TBL
0000:0030       PRNTBL
0000:0034       LPT2TBL
0000:0038       LPT3TBL
0000:003C       LPT4TBL
0000:0040       LPT5TBL
0000:0046       COMM_TBL
0000:0046       AUXTBL
0000:0051       COM2TBL
0000:005C       COM3TBL
0000:0067       COM4TBL
0000:0072       COM5TBL
0000:0083       MEMORY_SIZEK
0000:0085       PTRSAV
0000:0089       PARMS
0000:0089       LINBUF
0000:0189       LASTPRM
0000:019F       INP_BUF
0000:01EF       BUF_A
0000:01F1       BUF_E
0000:021D       KBD_TT
0000:0223       CLEAR_1
0000:0227       DEC_SIGN_1
0000:0233       CLEAR_2
0000:0237       DEC_SIGN_2
0000:023D       CMDTABL
0000:0266       COMTBL
0000:02A6       ETBLENT
0000:02A8       ESCTBL
0000:02D5       KBD_RDEF_TBL
0000:0355       KBDTBL
0000:035F       DEC_S_1
0000:036F       DEC_S_2
0000:0373       HEB_SW
0000:0376       CRTACTTBL
0000:0476       DEVSTART
0000:0476       CONDEV
0000:0488       AUXDEV
```

```
Origin          Group
Address         Publics by Value

0000:049A       PRNDEV
0000:04AC       TIMDEV
0000:04BE       COM1DEV
0000:04D0       COM2DEV
0000:04E2       COM3DEV
0000:04F4       COM4DEV
0000:0506       COM5DEV
0000:0518       LPT1DEV
0000:052A       LPT2DEV
0000:053C       LPT3DEV
0000:054E       LPT4DEV
0000:0560       LPT5DEV
0000:0572       DSKDEV
0000:0584       WINDEV
0000:0596       RE_INIT
0000:05A1       MEM_SIZE_DET
0000:05A6       INT_TRAP
0000:0602       STACKTOP
0000:0610       FUNCTBL
0000:08F5       CON_INT
0000:09F0       KBD_FL
0000:0BC3       BELL
0000:0BE9       PLAY_MUSIC
0000:0C4B       KBD_BUFF_IN
0000:0CC4       I16_HANDLER
0000:0D56       DEFFK
0000:0E37       I29_HANDLER
0000:0EB1       H2
0000:0EFE       ST12
0000:0FE0       CUU
0000:0FF3       CUD
0000:1012       CUF
0000:1026       CUB
0000:1038       CUP
0000:104B       PSCP
0000:1052       PRCP
0000:105B       XDSR
0000:109E       POSIT
0000:10D7       BACKSP
0000:1108       LINEFD
0000:1170       RLF
0000:1188       NDFS
0000:119E       CARRET
0000:11A6       VHOME
0000:1223       SGR
0000:1316       SHALF_INT
0000:1376       RHALF_INT
0000:13B3       REVERSE
0000:1430       ED
0000:1457       EL
0000:1485       BLEOS
0000:14BC       INSLIN
0000:1507       DELLIN
0000:1557       ICHR
0000:15AB       DCHR
0000:194E       COM_INT1
```

```
Origin          Group
Address         Publics by Value

0000:1954       COM_INT2
0000:195A       COM_INT3
0000:1960       COM_INT4
0000:1966       COM_INT5
0000:1DBE       RS232_INT
0000:1E66       LPT1_ACT
0000:1E70       LPT2_ACT
0000:1E7A       LPT3_ACT
0000:1E84       LPT4_ACT
0000:1E8E       LPT5_ACT
0000:20D2       KBD_CC
0000:2141       TIM_INT
0000:217B       TIM2_ISR
0000:219A       TIMER_RET
0000:2E4C       DSK_INT
0000:3441       DREND
0000:3450       WI_START
0000:3453       WI_STRATEGY
0000:3462       WI_INTERRUPT
0000:38E7       WI_END
0000:38F0       DUMMY5_START
0000:3DB5       DUMMY5_END
0000:3FB0       HWINIT
041D:0000       SYSINIT
041D:0005       CURRENT_DOS_LOCATION
041D:0009       FINAL_DOS_LOCATION
041D:000B       DEVICE_LIST
041D:000F       MEMORY_SIZE
041D:0011       DEFAULT_DRIVE
041D:0012       BUFFERS
041D:0013       FILES
041D:077E       BADOPM
041D:07A2       BADSIZ_POST
041D:07A2       BADLD_POST
041D:07A2       CRLFM
041D:07A5       BADSIZ_PRE
041D:07C6       BADLD_PRE
041D:07DB       BADCOM
041D:07EC       BADCOUNTRY
041D:0805       SYSSIZE

Program entry point at       0000:0000
```