NCR

System Technical Manual CP/M-86™

NCR

NCR DECISION MATE V

System Technical Manual -CP/M-86™

CP/M-80 is a trademark of Digital Research Inc.
CP/M-86 is a trademark of Digital Research Inc.
CP/M is a registered trademark of Digital Research 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.

Document Order Numbers

Augsburg Stock No.: 017-0031681
MIRS Library, Dayton: ST-2104-35

First Edition, November 1983

It is the policy of NCR Corporation 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 instances, photographs are of equipment prototypes. Therefore, before using this document, consult your dealer or 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

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

CP/M-86 SOFTWARE FOR INPUT/OUTPUT

CONTENTS

CP/M-86 SOFTWARE FOR INPUT/OUTPUT

CP/M-86 SYSTEM OVERVIEW

CP/M -86 is an operating system that loads from flexible disk into read/write memory. A set of frequently used utilities reside in memory, while others are loaded from disk as required.

Features of CP/M-86 in your NCR DECISION MATE V include field specification of one to eight logical drives (two flexible disk drives, six Winchester disk drives as three units). Any particular file can reach the full drive size. Users of CP/M-86 are physically separated by user numbers, with facilities for file copy operations from one user area to another. Powerful relative-record random access functions are present in CP/M-86 that provide direct access to any of the 65536 records of an eight-megabyte file. CP/M-86 also includes an Intel-compatible assembler (ASM86) and a debugging utility (DDT86), with which you can load, test, and save programs.

The first three tracks of surface 0 of your operating system flexible disk contain only the loader program. The operating system itself (CPM.SYS) resides on disk in much the same way as the CP/M-86 utilities. During initialization this loader is present in memory between 2000H and 5000H. The addresses below the loader are left free for ROM selection. The operating system is initially loaded to 6000H. As the final stage of the initialization procedure, the operating system is moved downwards in memory to the top of the 8086 interrupt vector area (400H), thus overwriting the loader. The BIOS program for your NCR DECISION MATE V starts 2500H bytes above 400H. The segment registers CS and DS are each set to the paragraph value 40H. With the operating system loaded, you have approximately 38KB at your disposal in a 64KB NCR DECISION MATE V.

The GENCMD utility (described in detail in the CP/M-86 Manual, which you received with your operating system flexible disk) allows you to choose from a number of memory models: you can make use of independent segments, where the "base page" of length 100H is written by the operating system to the beginning of the data segment. Alternatively, you can set up an 8080 memory model, where CS and DS address the same area in memory, so that the base page corresponds to the page zero (first 100H bytes of memory) of CP/M-80 software. However, you should note that location 5 in the base page does not contain the CP/M-80 page zero jump instruction, and that the IOBYTE is not present. (The IOBYTE is situated in the BIOS program at location 258BH relative to paragraph 40H.)

The CP/M-86 base page consists of the following elements:

Bytes 0-2:
The address in 24 bits (four uppermost bits in byte 2 = zero) of the last location in memory used by the code segment. In the 8080 memory model, this value can never exceed 0FFFFH.

Bytes 3-4:
The value in 16 bits of the base paragraph of the code segment.

Byte 5:
Value 1 to denote 8080 memory model.

Bytes 6-0AH:
Information as in bytes 0-4, this time for data segment. Byte 0BH is unused.

Bytes 0CH-0EH:
Length of area used by the extra segment.

Bytes 0FH-10H:
Base paragraph of the extra segment. Byte 11H is unused.

Bytes 12H-17H:
As in bytes 0FH-10H, this time for stack segment. (The CCP area includes a 96-byte default stack area.) Byte 17H is unused.

Bytes 18H-1DH, 1EH-23H, 24H-29H, 2AH-2FH:
Four optional groups which may be required for programs executing under the compact memory model (see CP/M-86 Manual).

Bytes 30H-5BH:
Not currently used.

Bytes 5CH-7FH:
Default FCB.

Bytes 80H-0FFH:
Default buffer.

CP/M-86 SYSTEM OVERVIEW FOR CP/M-80 PROGRAMMERS

CP/M-86 GENERAL CHARACTERISTICS

CP/M-86 contains all facilities of CP/M-80 with additional features to account for increased processor address space of up to a megabyte (1,048,576) of main memory. Further, CP/M-86 maintains file compatibility with all previous versions of CP/M. The file structure of version 2 of CP/M is used. Thus, CP/M-80 and CP/M-86 systems may exchange files without modifying the file format.

CP/M-86 resides in the file CPM.SYS, which is loaded into memory by a cold start loader during system initialization. The cold start loader resides on the first three tracks of the system disk. CPM.SYS contains three program modules:

The Console Command Processor (CCP),
the Basic Disk Operating System (BDOS),
the Basic I/O System (BIOS).

The operating system executes above the reserved interrupt locations, while the remainder of the address space is partitioned into as many as eight non-contiguous regions, as defined in a BIOS table. Unlike CP/M-80, the CCP area cannot be used as a data area subsequent to transient program load; all CPM.SYS modules remain in memory at all times, and are not reloaded at a warm start.

Similarly to CP/M-80, CP/M-86 loads and executes memory image files from disk. Memory image files are preceded by a "header record," which provides information required for proper program loading and execution. Memory image files under CP/M-86 are identified by a "CMD" file type.

Unlike CP/M-80, CP/M-86 does not use absolute locations for system entry or default variables. The BDOS entry takes place through a reserved software interrupt (INT 224), while entry to the BIOS is provided by a new BDOS call. Two variables maintained in low memory under CP/M-80, the default disk number and I/O Byte, are placed in the CCP and BIOS, respectively. Dependence upon absolute addresses is minimized in CP/M-86 by maintaining initial "base page" values, such as the default FCB and default command buffer, in the transient program data area.

The GENCMD (Generate CMD) utility replaces the LOAD program of CP/M-80, and converts the hex files produced by ASM-86 or Intel utilities into memory image format suitable for execution under CP/M-86. In addition, a variation of GENCMD, called LMCMD, converts output from the Intel LOC86 utility into CMD format.

A group consists of segments that are loaded into memory as a single unit. Since a group may consist of more than 64KB, it is the responsibility of the application program to manage segment registers when code or data beyond the first 64KB segment is accessed.

CP/M-86 supports eight program groups: the code, data, stack and extra groups as well as four auxiliary groups. When a code, data, stack or extra group is loaded, CP/M-86 sets the respective segment register (CS, DS, SS, or ES) to the base of the group. CP/M-86 can also load four auxiliary groups. A transient program manages the location of the auxiliary groups using values stored by CP/M-86 in the user's base page.

CP/M-80 AND CP/M-86 DIFFERENCES

The structure of CP/M-86 is as close to CP/M-80 as possible, in order to provide a familiar programming environment which allows application programs to be transported to the 8086 and 8088 processors with minimum effort.

Due to the nature of the 8086 processor, the fundamental difference between CP/M-80 and CP/M-86 is found in the management of the various relocatable groups. Although CP/M-80 references absolute memory locations by necessity, CP/M-86 takes advantage of the static relocation inherent in the 8086 processor. The operating system itself is loaded directly above the interrupt locations, at location 0400H, and relocatable transient programs load in the best fit memory region. Transient programs will load and run in any non-reserved region.

To make a BDOS system call, use the reserved software interrupt # 244. The jump to the BDOS at location 0005 found in CP/M-80 is not present in CP/M-86. However, the address field at offset 0006 in the base page is present so that programs which "size" available memory using this word value will operate without change. CP/M-80 BDOS functions use certain 8080 registers for entry parameters and returned values. CP/M-86 BDOS functions use a table of corresponding 8086 registers. For example, the 8086 registers CH and CL correspond to the 8080 registers B and C. Look through the list of BDOS function numbers in Figure 5.3 and you will find that function 0, as well as functions 1BH and 1FH, have changed slightly. Several new functions have been added, but they do not affect existing programs.

One major fundamental difference is that in CP/M-80, all addresses sent to the BDOS are simply 16-bit values in the range 0000H to 0FFFFH. In CP/M-86, however, the addresses are really just 16-bit offsets from the DS (Data Segment) register, which is set to the base of your data area. If you translate an existing CP/M-80 program to the CP/M-86 environment, your data segment will be less than 64KB. In this case, the DS register need not be changed following initial load, and thus all CP/M-80 addresses become simple DS-relative offsets in CP/M-86.

Under CP/M-80, programs terminate in one of three ways: by returning directly to the CCP, by calling BDOS function 0, or by transferring control to absolute location 000H. CP/M-86, however, supports only the first two methods of program termination. This has the side effect of not providing the automatic disk system reset following the jump to 0000H which, instead, is accomplished by entering a CONTROL-C at the CCP level.

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. Each surface of the flexible disk is considered as consisting of 40 concentric tracks, numbered consecutively 0 through 39. The two surfaces are designated surface 0 and surface 1. The spacing on the flexible disk is 48 tracks per inch. Each track is divided into 8 equal length 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 in the CP/M-86 operating system. Figure 5.1 presents the corresponding schematic layout. Certain elements of formatting on the flexible disk are fixed and invariable. 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.

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 (not normally required beyond Gap 1 on user tracks).

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: 01 = surface; 0,01 = surface 1.
Byte 3 Sector number 01 through 08.
Byte 4 Physical record length: 02 indicates 512 bytes per sector.

Data

The 512 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 bytes are used (i.e. A1, A1, A1, FB, and the 512 data bytes.)

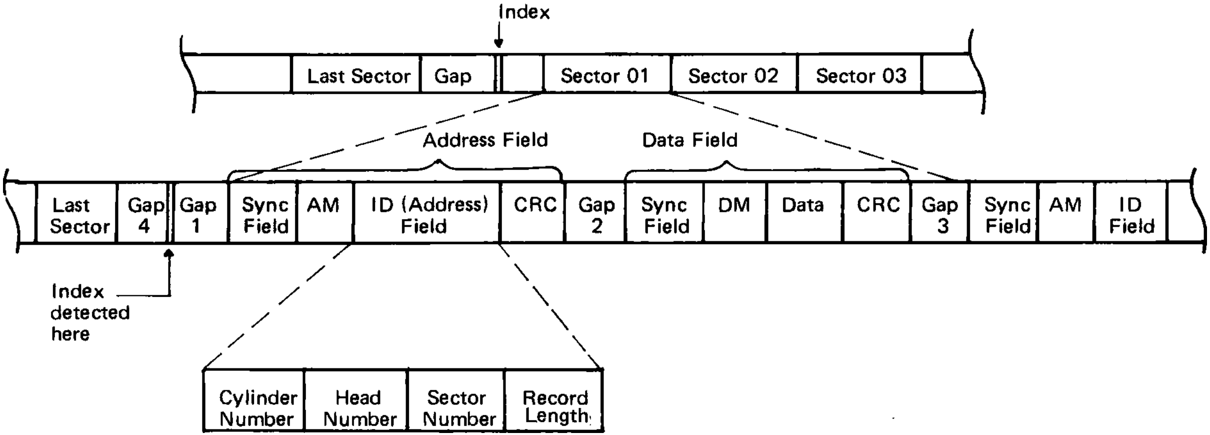


Figure 5.1

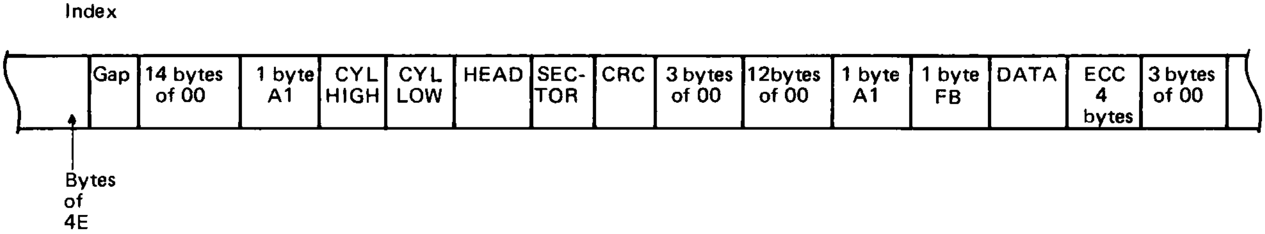


Figure 5.2

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 obligatory 6-byte disk identifier ("NCR F3") is contained at offset 10 on surface 0, track 0, sector 1.

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. Similar to the flexible disk-drive controller, the Winchester disk-drive controller uses polynomial codes (CRC and ECC) to check ID and data integrity. Figure 5.2 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 FC: cylinders 512 to 767
Value FD: cylinders 768 to 1023

CYL LOW
The eight least significant bits of the ten-bit cylinder number. (CYL HIGH contains the two most significant bits.)

HEAD
Bit 7 set indicates a bad block.

Bytes of 4E
A typical number of these bytes is 304 at 3600 r.p.m.

BDOS FUNCTIONS

A list of CP/M-86 calls is given in Figure 5.3, with an asterisk following functions which differ from, or are added to, the set of CP/M-80 Version 2 functions.

F# (Hex)	Result	F# (Hex)	Result
00	System Reset	19	Return Current Disk
01	Console Input	11	Set DMA Address
02	Console Output	1B*	Get Addr (Alloc)
03	Reader Input	1C	Write Protect Disk
04	Punch Output	1D	Get Addr (R/O Vector)
05	List Output	1E	Set File Attributes
06*	Direct Console I/O	1F*	Get Addr (Disk Parms)
07	Get I/O Byte	20	Set/Get User Code
08	Set I/O Byte	21	Read Random
09	Print String	22	Write Random
0A	Read Console Buffer	23	Compute File Size
0B	Get Console Status	24	Set Random Record
0C	Return Version Number	25*	Reset Drive
0D	Reset Disk System	28	Write Random with Zero Fill
0E	Select Disk	2F	Chain to Program
0F	Open File	31	Get Sysdat Address
10	Close File	32*	Direct BIOS Call
11	Search for First	33*	Set DMA Segment Base
12	Search for Next	34*	Get DMA Segment Base
13	Delete File	35*	Get Max Memory Available
14	Read Sequential	36*	Get Max Mem at Abs Location
15	Write Sequential	37*	Alloc Mem
16	Make File	38*	Alloc Absolute Memory Region
17	Rename File	39*	Free Memory Region
18	Return Log-in Vector	3A*	Free All Memory
		3B*	Program Load

Figure 5.3

Figure 5.4 explains briefly the nature of each function, the function number which must be loaded in Register CL, additional entry parameters and their required registers, as well as the significance of any return value. The advantage for programmers of using these entry points is that their validity is less likely to be impaired by future BIOS developments. For detailed descriptions see the CP/M-86 manual.

CP/M-86 allows dynamic allocation of memory into up to eight regions. This means that a program can be loaded into memory by another program, and this newly-loaded program can itself then load a further program, and so on. The memory areas thus allocated can be released again. Memory management functions beginning at 35H reference a Memory Control Block (MCB), defined in the calling program, which takes the form:

	16-bit	16-bit	8-bit
MCB:	M-Base	M-Length	M-Ext

where M-Base and M-Length are either input or output values expressed in 16-byte paragraph units, and M-Ext is a returned byte value, as defined specifically with each function code. An error condition is normally flagged with a 0FFH returned value in order to match the file error conventions of CP/M.

The memory management functions return information regarding: the largest available memory region, which is less than, or equal to, M-Length paragraphs; the largest possible region at the absolute paragraph boundary given by M-Base, for a maximum of M-Length paragraphs.

The MCB is also used for allocating memory. In this case, M-Length is filled by the programmer with the size of memory requested, or with the size and memory requested and an absolute base address.

Function no. in Reg. CL (Hex)	Description	Additional Entry Parameters in Reg.	Return Value in Reg.
00	System reset.	DL Abort Code	–
01	Console input – waits for character, which is echoed to console.	–	AL: ASCII character
02	Console output – tabs expanded, check for start/stop scroll.	DL: ASCII character	–
03	Reader input – waits for character	–	AL: ASCII character
04	Punch output.	DL: ASCII character	–
05	List output.	DL: ASCII character	–
06	Direct console I/O	DL: 0FFH: return key character 0FEH: return status only else: output this character	AL: ASCII-char. if ready, otherwise 0 0 = no char., < > 0 = char. ready
07	Get I/O Byte.	–	AL: IOBYTE
08	Set I/O Byte.	DL: IOBYTE	–

Figure 5.4 (1 of 5)

Function no. in Reg. CL (Hex)	Description	Additional Entry Parameters in Reg.	Return Value in Reg.
09	Print string until $ encountered — tabs and control chars. as in 02.	DX: String offset	—
0A	Read console buffer — reads console input into buffer at address DX until CR (0DH) or LF (0AH) or overflow. Other control chars. recognized.	DX: Buffer offset [DE+0]: Buffer length	[DX+1] number of characters in buffer
0B	Get console status.	—	AL: 1 if char. ready; otherwise 0
0C	Return version number.	—	BH: 00 = CP/M, BL: 00 = version before 2.0, lower nibble = release 2.n
0D	Reset disk system — all disks read/write, disk A selected	—	—
0E	Select disk.	DL: Drive A = 0 . . Drive P = 0FH	—
0F	Open file — if found, directory information copied to FCB.	DX: FCB offset	AL: 0,1,2, or 3 = found, otherwise 0FFH.
10	Close file — new FCB recorded in disk directory.	DX: FCB offset	AL: 0,1,2, or 3 = old directory entry found, otherwise 0FFH
11	Search for first file entry in directory corresponding to FCB.	DX: FCB offset	AL: 0,1,2, or 3 = found, otherwise 0FFH
12	Search for next file entry after last matched entry.	—	AL: 0,1,2, or 3 = found, otherwise 0FFH
13	Delete file matching FCB.	DX: FCB offset	AL: 0 = found, otherwise 0FFH

Figure 5.4 (2 of 5)

Function no. in Reg. CL (Hex)	Description	Additional Entry Parameters in Reg.	Return Value in Reg.
14	Read sequential record of opened file (function 0F or 16) to DMA address (function 1A).	DX: FCB offset	AL: 0 = read successful, 1 = no data exists
15	Write sequential record of opened file (function 0F or 16) from DMA address (function 1A).	DX: FCB offset	AL: 0 = write successful 1 = no available directory space 2 = no available data block
16	Make file which does not already exist.	DX: FCB offset	0,1,2, or 3 = successful, 0FFH = no directory space
17	Rename file.	DX: address of FCB inc. old name. (DE+10H): new name	0 = successful 0FFH = old name not found
18	Return Log-in vector.	–	BX: bit significance 0. . .15 corresponds to drive A. . .P, 0 bit set = drive not on line, 1 bit set = drive on line
19	Return current disk.	–	AL: 0. . .0FH corresponding to drive A. . .P
1A	Set DMA address – i.e. address of data record for read or write operation.	DX: DMA offset	–
1B	Get address of drive allocation vector.	–	BX: Alloc offset ES: segment base
1C	Temporary disk write protection.	–	–

Figure 5.4 (3 of 5)

Function no. in Reg. CL (Hex)	Description	Additional Entry Parameters in Reg.	Return Value in Reg.
1D	Get read only vector.	–	BX: bit significance 0. . .15 corresponds to drive A. . .P, bit set = R/O
1E	Set file attributes in directory in accordance with attributes in FCB.	DX: FCB offset	AL: 0 = successful 0FFH = file named in FCB not found
1F	Get address of disk parameter block	–	BX: DPB offset ES: segment base
20	Set/get user code.	DL: 0FFH = get number Otherwise, set number to register contents	AL: user number
21	Read random	DX: FCB offset	AL: 00 = successful; or error codes
22	Write random	DX: FCB offset	AL: 00 = successful; or error codes
23	Compute file size	DX: FCB offset	Random Record Field Set
24	Set random record	DX: FCB offset	Random Record Field Set
25	Reset drive	DX: Drive vector bit significance 0. . .15 corresponds to drive A. . .P, bit set = drive to be reset	AL: 00
26, 27	Not in use		
28	Write random with zero fill	DX: FCB offset	See Function 22
2F	Chain to program	DMA buffer: Command line	

Figure 5.4 (4 of 5)

Function no. in Reg. CL (Hex)	Description	Additional Entry Parameters in Reg.	Return Value in Reg.
31	Get address of System Data Area		BX: SYSDAT Address offset ES: SYSDAT Address segment
32	Direct BIOS call	DX: BIOS Descriptor	—
33	Set DMA base segment	DX: Base Address	—
34	Get DMA base segment		BX: DMA offset ES: DMA segment
35	Get largest area of memory available	DX: Offset of Memory Control Block (MCB)	AL: request 00 = successful, 0FFH = no memory available M-EXT: 0 = no additional memory available, 1 = add mem. f. allocation
36	Get largest area of memory available at paragraph boundary specified in MCB	DX: Offset of MCB	AL: 00 = successful, 0FFH = no memory available
37	Allocate memory	DX: Offset of MCB	AL: 00 = successful, 0FFH = not allocated
38	Allocate absolute memory	DX: Offset of MCB	AL: 00 = successful, 0FFH = not allocated
39	Free memory	DX: Offset of MCB	—
3A	Free all memory		
3B	Program load	DX: Offset of FCB	AX: Return Code/ Base Page Addr BX: Base Page Addr

Figure 5.4 (5 of 5)

FILE INFORMATION

CP/M-86 identifies every file by the drive specifier (1 character — optional), the file name (1-8 characters), and the file type (1-3 characters — optional). The file itself consists of byte by byte information logically divided into lines by the hexadecimal sequence 0DH, 0AH (carriage return, line feed). When reading, CP/M-86 interprets the hexadecimal value 1A as end-of-file except in machine-executable files (e.g. COM). A file is divided into 16 KB logical extents automatically accessed in both sequential and access modes.

A CP/M-86 utility or user program may make use of the default file control block (FCB) situated at offset 005CH from the DS register. The basic unit used in the reading and writing of files is the 128-byte record, for which CP/M-86 provides a default location at 0080H.

The FCB data area (i.e. from 005CH onward) uses 33 bytes for sequential, and 36 bytes (i.e. up to and including 007FH) for random file access. The FCB layout is as follows. The numbers 00 to 35 in the layout denote the offsets of the individual bytes to the FCB beginning.

dr	fl	f2	//	f8	t1	t2	t3	ex	s1	s2	rc	d0	//	dn	cr	r0	r1	r2
00	01	02	...	08	09	10	11	12	13	14	15	16	...	31	32	33	34	35

dr

drive code (0-16)
0 = use default drive for file,
1 = auto disk select drive A,
2 = auto disk select drive B.
. . .
16 = auto disk select drive P.

f1. . .f8

Contain the file name in ASCII upper case, with high bit = 0.

t1, t2, t3

Contain the file type in ASCII upper case (bit 7 = zero). The high bits t1' and t2' are used as follows:

t1' = 1: Read/Only file
t2' = 1: SYS file, no DIR list

ex

Contains the current extent number, normally set to 00 by the user, but in range 0-31 during file I/O.

s1

Reserved for internal system use.

s2

Reserved for internal system use, set to zero on call to OPEN, MAKE, SEARCH.

rc

Record count for extent "ex," takes on values from 0-128.

d0. . .dn

Reserved for system use.

cr

Current record to read or write in a sequential file operation, normally set to zero by user.

r0, r1, r2

Optional random record number in the range 0-65535, with overflow to r2. r0, r1 constitute a 16-bit value with low byte r0 and high byte r1.

FCBs are stored in a directory area of the disk and are brought into memory by BDOS Function 0F or 16 before file operations can commence. The memory copy of the FCB is updated during file operations and recorded permanently on disk when these operations are concluded (Function 10H).

CP/M-80 Version 2 and CP/M-86 perform directory operations in a reserved area of memory that does not affect write buffer content, except in the case of Search and Search Next, where the directory record is copied to the current DMA address.

Function 21H has as its entry parameter an FCB address in the register pair DX. A 16-bit value in the bytes r0 (least significant) and r1 indicates the random record to be read. The value of byte r2 must be zero. The file must already have been opened (Function 0F). If the random read is successful, the value of register AL is zero and the accessed record is at the current DMA address. If wishing to random read the next extent, the user must increment the record number, as the next read does not do this automatically. This is true also after switching to sequential read for the first read operation. Error codes returned in register AL are:

01 or 04	Read attempted beyond last file extent.
03	Cannot close current extent (bad or no FCB).
06	Random record number out of range.

For full details of error codes, refer to the NCR CP/M-86 Manual.

Function 22H is a write-random facility, using data from the current DMA address. The information given above about Function 21H applies analogously to this function. In addition, error code 05 indicates failure to write due to directory overflow.

Function 23H refers to the FCB addressed by the DX register and writes a binary value in the bytes r0 (least significant) and r1 in accordance with the highest record number. (This is not necessarily the actual number of records for files created in the random mode.) If r2 = 01, then the file contains the maximum number of records (65536). This function is useful for appending random files.

Function 24H is used to set a random record number in bytes r0 and r1 of the FCB addressed by the DX register. This FCB usually belongs to a file which has hitherto been accessed sequentially. This is useful when changing the access mode from sequential to random, or for noting the position of a record in a sequential file.

DISK INFORMATION

Tables are included in the BIOS that describe the particular characteristics of the disk subsystem used with CP/M-86. The purpose here is to describe the elements of these tables.

In general, each disk drive has an associated (16-byte) disk parameter header that contains information about the disk drive and provides a scratchpad area for certain BDOS operations. The format of the disk parameter header for each drive is shown below.

Disk Parameter Header

XLT	0000	0000	0000	DIRBUF	DPB	CSV	ALV
16b	16b	16b	16b	16b	16b	16b	16b

where each element is a 16-bit value. The meaning of each Disk Parameter Header (DPH) element is:

XLT

Always 0000H because no sector translation takes place (i.e. the physical and logical sector numbers are the same).

0000

Scratchpad values for use within the BDOS (initial value is unimportant).

DIRBUF

Offset of a 128-byte scratchpad area for directory operations within BDOS. All DPHs address the same scratchpad area.

DPB

Offset of a disk parameter block for this drive. Drives with identical disk characteristics address the same disk parameter block.

CSV

Offset of a scratchpad area used for software check for changed disks. This offset is different for each DPH.

ALV

Offset of a scratchpad area used by the BDOS to keep disk storage allocation information. This offset is different for each DPH.

Given n disk drives, the DPHs are arranged in a table whose first row of 16 bytes corresponds to drive 0, with the last row corresponding to drive n-1. The table thus appears as

DPBASE:

00	XLT 00	0000	0000	0000	DIRBUF	DBP 00	CSV 00	ALV 00
01	XLT 01	0000	0000	0000	DIRBUF	DBP 01	CSV 01	ALV 01
				and so on through				
n-1	XLTn-1	0000	0000	0000	DIRBUF	DBPn-1	CSVn-1	ALVn-1

where the label DPBASE defines the offset of the DPH table relative to the beginning of the operating system.

A responsibility of the SELDSK subroutine is to return the offset of the DPH for the selected drive. The Disk Parameter Block (DPB) for each drive is more complex. A particular DPB, which is addressed by one or more DPHs, takes the general form

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

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

SPT

The total number of sectors per track.

BSH

The data allocation block shift factor, determined by the data block allocation size. (BSH has for flexible disk a value of 4, for fixed disk a value of 6.)

BLM

The data allocation block mask (2^{BSH})-1. (BLM has for flexible disk a value of 0F, for fixed disk a value of 3F.)

EXM

The extent mask, determined by the data block allocation size and the number of disk blocks. (EXM has for flexible disk a value of 1, for fixed disk a value of 3.)

DSM

Number of allocation blocks possible on disk, minus one.

DRM

Number of directory entries that can be stored on the drive, minus one. (AL0, AL1 determine reserved directory blocks.)

CKS

The size of the directory check vector.

OFF

The number of reserved tracks at the beginning of the (logical) disk.

The value of DSM is the maximum data block number supported by this particular drive, measured in BLS (BLS for flexible disk = 2048 bytes, for fixed disk = 8192 bytes) units. The product BLS times (DSM+1) is the total number of bytes held by the drive, not counting the reserved operating system tracks.

The DRM entry is the one less than the total number of directory entries that can take on a 16-bit value. The values of AL0 and AL1, however, are determined by DRM. The values AL0 and AL1 can together be considered a string of 16 bits, as shown below.

$$\overbrace{00\ \ 01\ \ 02\ \ 03\ \ 04\ \ 05\ \ 06\ \ 07}^{\text{AL0}}\qquad\overbrace{08\ \ 09\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15}^{\text{AL1}}$$

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

Thus, if DRM = 127 (128 directory entries) and BLS = 2048, there are 64 directory entries per block, requiring 2 reserved blocks. In this case, the 2 high order bits of AL0 are set, resulting in the values AL0 = 0C0H and AL1 = 00H.

The CKS value is determined as follows: if the disk drive media is removable, then CKS = (DRM+1)/4, where DRM is the last directory entry number.

Finally, the OFF field determines the number of tracks that are skipped at the beginning of the physical disk (reserved operating system tracks). This value is automatically added whenever SETTRK (see section "The BIOS Entry Points") is called.

Returning back to the DPH for a particular drive, the two address values CSV and ALV remain. Both addresses reference an area of uninitialized memory following the BIOS. The areas must be unique for each drive, and the size of each area is determined by the values in the DPB.

The size of the area addressed by CSV is CKS bytes, which is sufficient to hold the directory check information for this particular drive. If CKS = (DRM+1)/4, one must reserve (DRM+1)/4 bytes for directory check use. If CKS = 0, no storage is reserved.

The size of the area addressed by ALV is determined by the maximum number of data blocks allowed for this particular disk and is computed as (DSM/8)+1.

LOGICAL ASSIGNMENT OF I/O DEVICES

CP/M makes use of four types of communication channel:

CONSOLE

Interactive communication with the operator.

LIST

Output channel to the principle listing device, usually a printer.

PUNCH

Punching device.

READER

Reading device.

Each of a number of physical devices is assigned to one or more of these logical devices. The physical devices are TTY (serial printer device), CRT, LPT (parallel printer). Figure 5.5 shows the possible bit settings within the IOBYTE which can be carried out by the BDOS Function 08. The Console field occupies bits 0 and 1 of the IOBYTE, the Reader field occupies bits 2 and 3, the Punch field bits 4 and 5, and the List field bits 6 and 7.

Console assigned to . . .	**Binary value of bits 0 with 1**
TTY	0 or 3
CRT	1 or 2

Figure 5.5 (1 of 4)

Reader assigned to . . .	**Binary value of bits 2 with 3**
TTY	0 or 3
CRT	1 or 2

Figure 5.5 (2 of 4)

Punch assigned to . . .	**Binary value of bits 4 with 5**
TTY	0
CRT	1 or 3
LPT	2

Figure 5.5 (3 of 4)

List assigned to . . .	**Binary value of bits 6 with 7**
TTY	0 or 3
CRT	1
LPT	2

Figure 5.5 (4 of 4)

TERMINAL FUNCTIONS

This section concerns the possibilities of software manipulation of the CRT display. CP/M-86 recognizes a number of codes up to three bytes in length 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. Figure 5.6 summarizes the function codes. With reference to this figure, 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 the key for a particular function.

The function codes are the same as those used by the Lear Siegler ADM-31™ terminal, with the following exceptions: 17H (Clear to End of Line) and 1BH 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).

The frequencies produced by the Play Music function are shown in Figure 5.7.

It is not possible to set color by means of a terminal function code. However, you can set color by means of the CRT attribute byte at the memory address 44DC. This address must, of course, be understood as an offset to the paragraph value 40H.

Foreground and background colors are determined by the six most significant bits of the attribute byte (see Figure 5.8). Bit 1 set activates video blinking.

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 (Red on color CRT)	1B 29
RESET INVERSE AND BLINKING	1B 47 30
VIDEO INVERSE ON	1B 47 34
BLINKING ON	1B 47 32
RING THE BELL	07
MUSIC	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 5.6 (1 of 2)

TERMINAL FUNCTION CODES (2)

Function

Program function key ESC, F, FN, STRING, FN

where:	ESC	=	ESCAPE character	(hex value 1B)
	F	=	Function code	(hex value 46)
	FN	=	Function number	(hex values between E0 for function key 1 and F3 for function key 20
	STRING	=	Character string	(a string of ASCII characters including control characters* hex values between 0 and 7F

Example: The followng string programs function key F2 with DIR (all values in hex): 1B 46 E1 44 49 52 0D E1

* control character 09 (Horizontal Tabulation) not allowed.

The advantage to the programmer of this method is that there is no need to return to CP/M-86 system level in order to program a Function Key via the CONFIG utility.

Figure 5.6 (2 of 2)

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 5.7

CRT ATTRIBUTES		
COLOR	Binary value in 3 bits:	
	BACKGROUND (Bits 7, 6, 5)	FOREGROUND (Bits 4, 3, 2)
White	0	7
Cyan	1	6
Magenta	2	5
Blue	3	4
Yellow	4	3
Green	5	2
Red	6	1
Black	7	0

Figure 5.8

THE BIOS PROGRAM

The BIOS portion of CP/M-86 resides in the topmost portion of the operating system (highest addresses), and takes the general form shown in Figure 5.9.

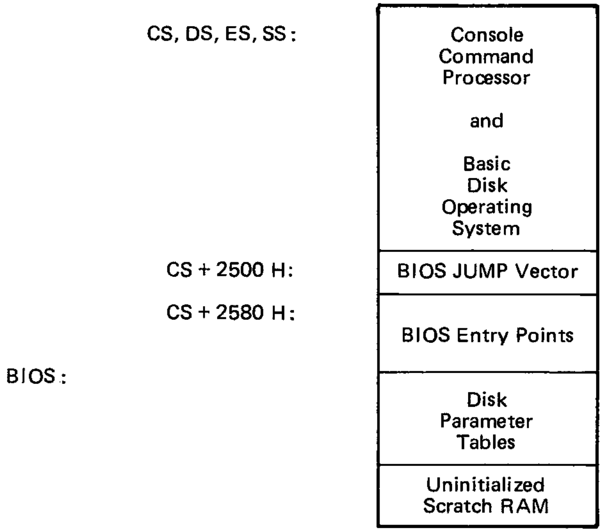


Figure 5.9 General CP/M-86 Organization

DISPLAYING THE BIOS PROGRAM ON THE SCREEN

The user can display the BIOS program on the CRT by making use of the Dynamic Debugging Tool utility (DDT86) which is provided as part of the CP/M-86 operating system. A full description of DDT86 is contained in the NCR CP/M-86 manual. It suffices here to say that with the aid of DDT86, the user can enter instructions in assembly language, produce a hexadecimal display of memory on the screen, initialize areas of memory, list the contents of memory in assembly language, transfer the contents of one area of memory to another, load disk files into memory, change the content of memory, and execute programs with or without display of CPU registers. The two DDT commands which are of interest here are the display of memory (D), and assembly language listing (L). As BIOS is already present in memory, it is only necessary to load the DDT utility.

An important note is justified here concerning the use of the L command in DDT. The disassembler interprets memory as assembler instructions. This means that areas of data storage created by the assembler directives DB, DW, or RS, or memory areas which simply are not used by the BIOS program, can lead to incorrect disassembly. Therefore, the user should ascertain that memory being disassembled contains only valid assembler instructions. The disassembler does not provide symbols.

THE BIOS JUMP VECTOR

Entry to the BIOS is through a “jump vector” located at offset 2500H from the base of the operating system. The jump vector is a sequence of 23 three-byte jump instructions which transfer program control to the individual BIOS entry points. (Figure 5.10).

Parameters for the individual subroutines in the BIOS are passed in the CX and DX registers, when required. CX receives the first parameter; DX is used for a second argument. Return values are passed in the registers according to type: Byte values are returned in AL. Word values (16 bits) are returned in BX. Specific parameters and returned values are described with each subroutine.

There are three major divisions in the BIOS jump table: system (re)initialization subroutines, simple character I/O subroutines, and disk I/O subroutines.

THE BIOS ENTRY POINTS

The earlier section “BDOS Functions” demonstrated the range of I/O functions which can be used by loading the CL and other registers with entry parameters and issuing INT 224. The BIOS

Offset to paragraph 40H	Suggested Instruction	BIOS F#	Description
2500H	JMP INIT	0	Arrive Here from Cold Boot
2503H	JMP WBOOT	1	Arrive Here for Warm Start
2506H	JMP CONST	2	Check for Console Char Ready
2509H	JMP CONIN	3	Read Console Character
250CH	JMP CONOUT	4	Write Console Character
250FH	JMP LIST	5	Write List Character
2512H	JMP PUNCH	6	Write Char to Punch Device
2515H	JMP READER	7	Read Reader Device
2518H	JMP HOME	8	Move to Track 00
251BH	JMP SELDSK	9	Select Disk Drive
251EH	JMP SETTRK	10	Set Track Number
2521H	JMP SETSEC	11	Set Sector Number
2524H	JMP SETDMA	12	Set DMA Offset Address
2527H	JMP READ	13	Read Selected Sector
252AH	JMP WRITE	14	Write Selected Sector
252DH	JMP LISTST	15	Return List Status
2530H	JMP SECTRAN	16	Sector Translate
2533H	JMP SETDMAB	17	Set DMA Segment Address
2536H	JMP GETSEGB	18	Get Offset of memory region table
2539H	JMP GETIOB	19	Get I/O Mapping Byte
253CH	JMP SETIOB	20	Set I/O Mapping Byte
253FH*	JMP SPECFUN	21	Sets up parameter for BIOS functions
2542H*	JMP SELTYP	22	Returns params for EXCHANGE utility

* NON-STANDARD BIOS FUNCTION

Figure 5.10

includes a similar vector, from which I/O functions can be activated by means of a programmed call to one of twenty-three addresses in this vector. A description of these functions follows. The hexadecimal numbers in parentheses represent the positive offset (to the BIOS starting point) of the first byte of the jump instruction which activates that function.

INIT
(0000)

This subroutine is called directly by the CP/M-86 loader after the CPM.SYS file has been read into memory. The procedure is responsible for any hardware initialization not performed by the bootstrap loader, setting initial values for BIOS variables (including IOBYTE), printing a sign-on message, and initializing the interrupt vector to point to the BDOS offset (0B06H) and base. When this routine completes, it jumps to the CCP offset (0H). All segment registers are initialized at this time to contain the base of the operating system.

WBOOT
(0003)
Warm start — BIOS is not reloaded. The routine jumps directly to the warm start entry point of the CCP (06H).

CONST
(0006)
Console status — returns 0FFH in register AL if the character is ready, otherwise 00H.

CONIN
(0009)
Console character returned in register AL. Bit 7 is reset. No return until a character is typed.

CONOUT
(000C)
Contents of register CL is sent to the console device.

LIST
(000F)
Contents of register CL is sent to the current listing device. (See section "Logical Assignment of I/O Devices.")

PUNCH
(0012)
Contents of register CL is sent to the currently assigned punch device. (See section "Logical Assignment of I/O Devices.")

READER
(0015)
Reader character returned in register AL. Bit 7 is reset. (See section "Logical Assignment of I/O Devices.")

HOME
(0018)
Disk head moves to track zero position.

SELDSK
(001B)
Selects disk drive according to contents of register CL:
0 = drive A. . . 15 = drive P. Register DL returns the address of the Disk Parameter Header (see section "Disk Information"), or zero if the drive does not exist.

SETTRK
(001E)
Selects track number contained in registers CX: 0-65535 for disk subsystems.

SETSEC
(0021)
Selects sector number contained in registers CX.

SETDMA
(0024)

Sets DMA address to contents of CX registers. The automatic warm boot setting is 0080H.

SETDMAB
(0033)

Register CX contains the segment base for subsequent DMA read or write operation. The BIOS will use the 128-byte buffer at the memory address determined by the DMA base and the DMA offset during read and write operations.

READ
(0027)

Using the set drive, track, sector, and DMA address, one disk sector is read. Normally, register AL returns zero. An error will return the value 1 and an error message. Thereupon CR will ignore the error, CONTROL-C will abort.

WRITE
(002A)

Disk sector is written. The data should be marked as "non-deleted data" to maintain compatibility with other CP/M systems. Settings and returns as in READ.

LISTST
(002D)

Returns status of list device: 0FFH in register AL indicates that the device is ready to receive a character. Useful for background printing.

SECTRAN
(0030)

Moves sector number in CX to BX.

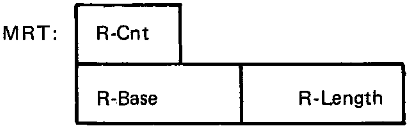
SPECFUN
(003F)

Sets up parameters for BIOS functions. Non-standard BIOS function.

GETSEGT
(0036)

Returns the address of the 5-byte Memory Region Table (MRT) in BX. The returned value is the offset of the table relative to the start of the operating system. The table defines the location and extent of physical memory which is available for transient programs.

Memory areas reserved for interrupt vectors and the CP/M-86 operating system are not included in the MRT. The Memory Region Table takes the form:



where R-Cnt is the number of Memory Region Descriptors (equal to 1), while R-Base and R-Length give the paragraph base and length of the physically contiguous area of memory.

GETIOBF
(0039)
Returns the current value of the logical to physical input/output device byte (IOBYTE) in AL. This eight-bit value is used to associate physical devices with CP/M-86's four logical devices.

SETIOBF
(003C)
Use the value in CL to set the value of the IOBYTE stored in the BIOS.

SELTYP
(0042)
Returns parameters for EXCHANGE utility. Non-standard BIOS function.

MAKING USE OF THE I/O SOFTWARE

The CP/M-86 input/output software operates from read/write memory to which the user has full access. Some advanced programmers may wish to adjust parts of BIOS to meet an exceptional requirement. In doing so, the assembler listing contained in the appendix is invaluable.

The majority of users wishing to activate I/O functions at machine code level will find the BDOS and BIOS entry points the most convenient modes of access to the I/O functions. You will notice a considerable similarity between the facitlities provided by these two modes of access. The most striking difference concerns the handling of console input and string output to the console device. The BDOS function 2 is intended for ASCII printable characters; in addition, scrolling is carried out as well as printer echo, if set. Cursor and CRT control functions, however, require the use of the BIOS function CONOUT. The other significant difference is the enhanced console printing facility from BDOS, the string function 9.

Where possible, programs should use the BDOS entry points. These have been provided in CP/M to ensure that your programs will also run with future developments of BIOS software. Remember that BIOS routines can be activated using the BDOS Function 32H. If BIOS entry points are used other than via this BDOS function, it is advisable to check the machine address of the BIOS vector before running user programs in an I/O system loaded from a different CP/M flexible disk.

SOME I/O EXAMPLES

This section contains some short examples of input/output between keyboard, CRT, loudspeaker, and printer. Your CP/M-86 system flexible disk includes a symbolic assembler which you can use for assembling these examples. When you have written your source file (e.g. TUNE) with the file extent .A86, you can proceed in accordance with the following sequence at system level:

```
ASM86 TUNE
```

Assuming no syntactical errors, enter

```
GENCMD TUNE
```

and finally load the executable machine code file into memory with

```
DDT86 TUNE
```

CP/M-86 sets segment registers for you, so when writing the examples, you should not specify segment values. However, do not forget the ORG 100H directive immediately after DSEG, as the first 256 bytes in the data segment are required by the operating system.

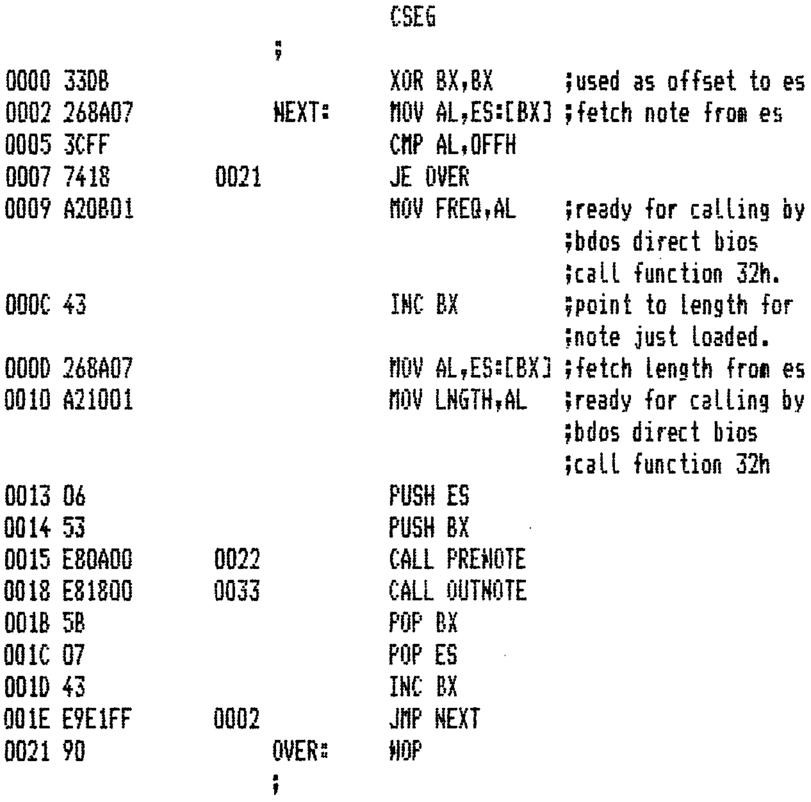
To run one of these programs, enter the G command in accordance with the description of the DDT86 utility in your NCR CP/M-86 Handbook. Do not forget to set a breakpoint immediately before the subroutines.

Your NCR CP/M-86 Handbook contains a sample program for disk access.

Tune

This is an example of how to drive the loudspeaker in your NCR DECISION MATE V. The program makes direct use of the BIOS subroutine for console output (Function 4) and the code for the Play Music terminal function (see Figure 5.7). The BIOS console output routine is accessed via the BDOS Function 32H. The BIOS function number and the CL register settings are passed via the data segment. The program uses four such 5-byte parameter blocks: the first two are for the 1B 4D sequence, the last two for frequency and length respectively. In each case the first two bytes only (BIOS function number and the value for the CL register) are used.

The data bytes for the tune itself are to be stored in the extra segment (ES). When you have loaded the program with DDT86, the operating system sets the segment registers. Using the DDT86 command SES:0 you can program your own tune, starting at ES:0. Simply enter frequency, length, frequency, length, and so on, in successive bytes. Conclude your tune with an FF byte for note. This tells the program that there are no more notes to play. You can then run the program from CS:0, with a break-point at 21H.



```
                                        CSEG

                              ;
0000 33DB                               XOR BX,BX       ;used as offset to es
0002 268A07               NEXT:         MOV AL,ES:[BX]  ;fetch note from es
0005 3CFF                               CMP AL,0FFH
0007 7418         0021                  JE OVER
0009 A20B01                             MOV FREQ,AL     ;ready for calling by
                                                        ;bdos direct bios
                                                        ;call function 32h.
000C 43                                 INC BX          ;point to length for
                                                        ;note just loaded.
000D 268A07                             MOV AL,ES:[BX]  ;fetch length from es
0010 A21001                             MOV LNGTH,AL    ;ready for calling by
                                                        ;bdos direct bios
                                                        ;call function 32h
0013 06                                 PUSH ES
0014 53                                 PUSH BX
0015 E80A00       0022                  CALL PRENOTE
0018 E81800       0033                  CALL OUTNOTE
001B 5B                                 POP BX
001C 07                                 POP ES
001D 43                                 INC BX
001E E9E1FF       0002                  JMP NEXT
0021 90                   OVER:         NOP
                              ;
```

```
                        ;s u b r o u t i n e s
                        ;
0022 8D160001           PRENOTE:  LEA DX,PRENOTE1 ;address of parameters
                                                  ;for bdos
                                                  ;direct bios call.
0026 B132                         MOV CL,32H      ;bdos direct bios
                                                  ;call function
0028 CDE0                         INT 224
002A 8D160501                     LEA DX,PRENOTE2
002E B132                         MOV CL,32H
0030 CDE0                         INT 224         ;the 1b 4d sequence has
0032 C3                           RET             ;now been transmitted
                        ;
0033 8D160A01           OUTNOTE:  LEA DX,OUTNOTE1 ;first the note
0037 B132                         MOV CL,32H
0039 CDE0                         INT 224
003B 8D160F01                     LEA DX,OUTNOTE2 ;then the length
003F B132                         MOV CL,32H
0041 CDE0                         INT 224
0043 C3                           RET
                        ;
                        ;
                                  DSEG
                                  ORG 100H
0100 041B000000         PRENOTE1  DB 4,1BH,0,0,0  ;bdos function 32h
                                                  ;parms:fn-cl-ch-dl-dh
0105 044D000000         PRENOTE2  DB 4,4DH,0,0,0  ;to activate music
                                                  ;terminal function.
010A 04                 OUTNOTE1  DB 4
010B 00000000           FREQ      DB 0,0,0,0      ;note for cl,
                                                  ;others unused
010F 04                 OUTNOTE2  DB 4
0110 00000000           LNGTH     DB 0,0,0,0      ;frequency for cl,
                                                  ;others unused
                        ;
                        ;
                                  ESEG
0000 304040403040                 DB 30H,40H,40H,40H,30H,40H,40H,40H,0FFH,0FFH
     4040FFFF                          ;write your tune in this data area,
                                       ;note-length-note-length and so on,
                                       ;finishing with 0FFH for note
                                  END
```

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, video blinking is activated temporarily. The program terminates when a dollar sign ($) is entered, and normal video is restored if necessary.

The keyboard echo to the screen is overwritten by use of the backspace terminal function. This is necessary as a character can appear on the screen only when it has been ascertained whether blinking or normal video is required.

```
 0001                CONIN     EQU 1       ;bdos keyboard input
 0002                CONOUT    EQU 2       ;bdos crt output
 0024                DOLLAR    EQU '$'
 0030                ZERO      EQU '0'
 0039                NINE      EQU '9'
 001B                VIDEO1    EQU 1BH     ;two byte sequence for
 0047                VIDEO2    EQU 47H     ;video attributes.
 0032                BLNKON    EQU 32h     ;sets blinking.
 0030                BLNKOFF   EQU 30H     ;resets blinking.
 001B                CLSCRN1   EQU 1BH     ;two byte sequence for
 003A                CLSCRN2   EQU 3AH     ;clear screen and
                                           ;cursor top left
 0008                BACKSP    EQU 8
 0020                BLANK     EQU 20H
 0032                BIOSCALL  EQU 32H     ;bdos direct bios call
                     ;
                     ;
                               CSEG
                     ;
0000 E83B00     003E           CALL CLSCRN
0003 E87000     0076 NEXTCH:   CALL VIDEORST ;ensure/reset to
                                           ;normal video
0006 E82400     002D           CALL READIN
0009 3C24                      CMP  AL,DOLLAR
000B 7418       0025           JE   DONE    ;terminate if dollar
                                           ;entered at keyboard
000D 3C30                      CMP  AL,ZERO
000F 7208       001C           JC   WRITE   ;jump if ASCII code <30h
0011 3C39                      CMP  AL,NINE
0013 7402       0017           JE   INVT    ;jump if ASCII code =39h
0015 7305       001C           JNC  WRITE   ;jump if ASCII code >39h
0017 50              INVT:     PUSH AX
```

```
0018 E84D00      0068               CALL VIDEOSET  ;change video mode
001B 58                             POP  AX
001C 8AD0             WRITE:        MOV  DL,AL
001E B102                           MOV  CL,CONOUT
0020 CDE0                           INT  224         ;write character,
                                                     ;blink if digit
0022 E9DEFF      0003               JMP  NEXTCH
0025 90               DONE:         NOP
                      ;
                      ;s u b r o u t i n e s
                      ;
0026 B132             DRCTBIOS:     MOV  CL,BIOSCALL ;bdos direct bios call.
0028 8D160001                       LEA  DX,BIOSOUT  ;address of 5 byte
                                                      ;parameter area
002C C3                             RET
                      ;
002D B101             READIN:       MOV  CL,CONIN    ;reads character
002F CDE0                           INT  224         ;from keyboard
0031 50                             PUSH AX
0032 E8F1FF      0026               CALL DRCTBIOS
0035 C606010108                     MOV  OUTCRT,BACKSP ;and places cursor
                                                       ;under it so that it
003A CDE0                           INT  224           ;will be overwritten
                                                       ;by the same character
003C 58                             POP  AX            ;after video mode
                                                       ;has been ascertained
003D C3                             RET
                      ;
003E E8E5FF      0026 CLSCRN:       CALL DRCTBIOS  ;clear screen and
                                                   ;cursor top left
0041 C60601011B                     MOV  OUTCRT,CLSCRN1
0046 CDE0                           INT  224
0048 E8DBFF      0026               CALL DRCTBIOS
004B C60601013A                     MOV  OUTCRT,CLSCRN2
0050 CDE0                           INT  224
0052 C3                             RET
                      ;
0053 E8D0FF      0026 PREVIDEO:     CALL DRCTBIOS  ;1b 47 sequence
                                                   ;to set video
0056 C60601011B                     MOV  OUTCRT,VIDEO1
005B CDE0                           INT  224
005D E8C6FF      0026               CALL DRCTBIOS
0060 C606010147                     MOV  OUTCRT,VIDEO2
0065 CDE0                           INT  224
```

```
0067 C3                          RET
                         ;
0068 E8E8FF        0053 VIDEOSET: CALL PREVIDEO
006B E8B8FF        0026           CALL DRCTBIOS
006E C606010132                   MOV  OUTCRT,BLNKON ;set to blinking
0073 CDE0                         INT  224
0075 C3                           RET
                         ;
0076 E8DAFF        0053 VIDEORST: CALL PREVIDEO
0079 E8AAFF        0026           CALL DRCTBIOS
007C C606010130                   MOV  OUTCRT.BLNKOFF   ;set to normal
0081 CDE0                         INT  224
0083 C3                           RET
                         ;
                         ;
                                  DSEG
                                  ORG  100H
                                                    ;parameters in 5 bytes
                                                    ;for bdos function number,
                                                    ;consisting of:
0100 04                 BIOSOUT   DB   4            ;bios function number
                                                    ;for console output,
0101 00000000           OUTCRT    DB   0,0,0,0      ;registers cl-ch-dl-dh.
                                                    ;Only ch used here
```

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 printing has finished, a form feed occurs. You need only press R for a further print copy. You may repeat this as often as you wish.

The program reserves 1000 bytes of uninitialized storage for your input. You can extend this storage reservation, depending on what other applications are presently in memory. Note that the last line of data and the form feed are realized at the printer only upon clearing the printer buffer.

```
  0001                     CONIN    EQU 1          ;bdos keyboard input.
  0002                     CONOUT   EQU 2          ;bdos crt output.
  0005                     OUTLIST  EQU 5          ;bdos output list device.
  0008                     BACKSP   EQU 8          ;cursor left.
  000C                     FORMFEED EQU 0CH        ;printer form feed.
  001B                     CLSCRN1  EQU 1BH        ;clear screen
  003A                     CLSCRN2  EQU 3AH        ;and cursor top left
  000D                     CR       EQU 0DH
  000A                     LF       EQU 0AH
  0020                     BLANK    EQU 20H
  0024                     DOLLAR   EQU '$'
  0052                     R        EQU 'R'
                           ;
                                    CSEG
                           ;
0000 E88300      0086               CALL CLSCRN
0003 BB0001                         MOV BX,100H    ;point to offset in ds
0006 53                    NEXT:    PUSH BX
0007 E85600      0060               CALL READIN
000A 5B                             POP BX
000B 8807                           MOV [BX],AL    ;keyboard input in memory.
000D 43                             INC BX         ;point to next
                                                   ;memory location
000E 3C08                           CMP AL,BACKSP
0010 750A        001C               JNE NOBACK
0012 4B                             DEC BX         ;if keyboard input was
0013 4B                             DEC BX         ;backspace, then remove it
                                                   ;from memory
0014 53                             PUSH BX
0015 E86100      0079               CALL ERASE
0018 5B                             POP BX
0019 E9EAFF      0006               JMP NEXT
001C 3C0D                  NOBACK:  CMP AL,CR      ;check for carriage return
001E 7508        0028               JNE NOLF
0020 53                             PUSH BX
0021 E84100      0065               CALL CRTLF     ;if carriage return,
                                                   ;then add line feed
0024 5B                             POP BX
0025 E9DEFF      0006               JMP NEXT
0028 3C24                  NOLF:    CMP AL,DOLLAR
002A 750A        0006               JNE NEXT       ;if not dollar then jump
                                                   ;to read keyboard again.
002C BB0001                PRINT:   MOV BX,100H    ;reset pointer to
                                                   ;beginning of text.
```

```
002F 8A17                NEXTP:   MOV DL,[BX]     ;fetch character
                                                  ;from memory
0031 80FA24                       CMP DL,DOLLAR
0034 7416           004C          JE DONE         ;$ concludes printing
0036 B105                         MOV CL,OUTLIST
0038 53                           PUSH BX
0039 52                           PUSH DX
003A CDE0                         INT 224         ;send character
                                                  ;to printer buffer
003C 5A                           POP DX
003D 5B                           POP BX
003E 80FA0D                       CMP DL,CR
0041 7505           0048          JNE NONLIN
0043 53                           PUSH BX
0044 E82B00         0072          CALL PRTLF      ;add line feed to cr
0047 5B                           POP BX
0048 43                  NONLIN:  INC BX          ;point to next character
0049 E9E3FF         002F          JMP NEXTP
004C E81D00         006C DONE:    CALL PRTCR      ;print remaining contents
                                                  ;of printer buffer
004F B20C                         MOV DL,FORMFEED
0051 B105                         MOV CL,OUTLIST
0053 CDE0                         INT 224         ;form feed in buffer
0055 E81400         006C          CALL PRTCR      ;and clear buffer
0058 E80500         0060          CALL READIN
005B 3C52                         CMP AL,R
005D 74CD           002C          JE PRINT        ;re-print if R pressed
005F 90                           NOP
                         ;
                         ;s u b r o u t i n e s
                         ;
0060 B101                READIN:  MOV CL,CONIN    ;read keyboard
0062 CDE0                         INT 224
0064 C3                           RET
                         ;
0065 B102                CRTLF:   MOV CL,CONOUT   ;add line feed to
                                                  ;carriage return on crt
0067 B20A                         MOV DL,LF
0069 CDE0                         INT 224
006B C3                           RET
                         ;
```

```
006C B105              PRTCR:    MOV CL,OUTLIST ;complete subroutine
006E B20D                        MOV DL,CR      ;clears printer buffer
0070 CDE0                        INT 224
0072 B105              PRTLF:    MOV CL,OUTLIST ;enter here to add
                                                ;line feed to cr
0074 B20A                        MOV DL,LF
0076 CDE0                        INT 224
0078 C3                          RET
                       ;
0079 B102              ERASE:    MOV CL,CONOUT  ;erase character on crt
007B B220                        MOV DL,BLANK
007D CDE0                        INT 224
007F B102                        MOV CL,CONOUT
0081 B208                        MOV DL,BACKSP
0083 CDE0                        INT 224
0085 C3                          RET
                       ;
0086 B102              CLSCRN:   MOV CL,CONOUT  ;clear screen and
0088 B21B                        MOV DL,CLSCRN1 ;cursor top left
008A CDE0                        INT 224
008C B102                        MOV CL,CONOUT
008E B23A                        MOV DL,CLSCRN2
0090 CDE0                        INT 224
0092 C3                          RET
                       ;
                       ;
                                 DSEG
                                 ORG 100H
0100 546869732069        DB 'This is overwriten by the text you enter'
     73206F766572
     777269747465
     6E2062792074
     686520746578
     7420796F7520
     656E746572
0129                             RS 1000
0511 4E6F206D6F72                DB 'No more, please'
     652C20706C65
     617365
                       ;
                                 END
```

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 5). To set the background color, enter $ instead of @. Enter $$ to terminate.

The program sets color by manipulating the attribute byte at the address 44DC relative to memory paragraph 40H. The paragraph value is contained in the ES register, and the attribute byte is addressed using a segment override prefix. The paragraph value in ES has to be set each time before accessing the attribute byte, as ES, unlike the other segment registers, is not restored following a BDOS call.

```
  44DC                ATTRIB    EQU 44DCH       ;crt attribute byte
  0001                CONIN     EQU 1           ;bdos - keyboard input.
  0002                CONOUT    EQU 2           ;bdos - crt output.
  0032                BIOSCALL  EQU 32H         ;bdos - direct bios call
  000D                CR        EQU 0DH
  000A                LF        EQU 0AH
  0024                DOLLAR    EQU '$'
  0040                ATSIGN    EQU '@'
  0030                ZERO      EQU '0'
  0037                SEVEN     EQU '7'
  001B                CLSCRN1   EQU 1BH         ;clear screen
  003A                CLSCRN2   EQU 3AH         ;and cursor home
  0008                CURBACK   EQU 8
  0020                BLANK     EQU 20H
  00FF                NOCHANGE  EQU 0FFH        ;no request for
                                                ;for color change.
  0000                COLCHANG  EQU 0           ;color change request.
  0001                TERMIN    EQU 1           ;end of keyboard input
                      ;
                                CSEG
                      ;
0000 E87700      007A           CALL CLRSCRN
0003 33DB                       XOR BX,BX
0005 8AFB             NEXT:     MOV BH,BL       ;last key pressed to bh
0007 53                         PUSH BX
0008 E86300      006E           CALL READIN
000B 5B                         POP BX
000C 8AD8                       MOV BL,AL       ;newly pressed key in bl
```

```
000E 80FB0D                       CMP BL,CR
0011 7508         001B            JNE NOLF        ;jump if not cr
0013 53                           PUSH BX
0014 E85C00       0073            CALL CRTLF      ;add lf to cr on crt
0017 5B                           POP BX
0018 E9EAFF       0005            JMP NEXT
001B E89000       00AE NOLF:      CALL QCOLOR
001E 3C01                         CMP AL,TERMIN
0020 7444         0066            JE DONE
0022 3C00                         CMP AL,COLCHANG
0024 75DF         0005            JNE NEXT
0026 53                           PUSH BX
0027 E86500       008F            CALL ERASE      ;erase color change
002A E86200       008F            CALL ERASE      ;sequence on crt
002D 5B                           POP BX
002E 80EB30                       SUB BL,30H      ;color 0-7 in bl
0031 80FF24                       CMP BH,DOLLAR
0034 750E         0044            JNE FOREGR      ;jump if foreground
                                                  ;color change.
0036 F6D3                         NOT BL          ;so that a number 0-7
                                                  ;produces the same
                                                  ;color,irrespective of
                                                  ;whether foreground
                                                  ;or background.
0038 B105                         MOV CL,5        ;count for shift.
003A D2E3                         SHL BL,CL       ;new background color
                                                  ;in bits 5,6,7,
                                                  ;others reset.
003C 802605011F                   AND COLBYTE,1FH ;reset bits 5,6, and
                                                   ;7,others unaffected
0041 E90C00       0050            JMP COLSET
0044 D0E3              FOREGR:    SHL BL,1
0046 D0E3                         SHL BL,1        ;new foreground color
                                                  ;in bits 2,3,4,
                                                  ;bits 0 and 1 reset.
0048 80E31F                       AND BL,1FH      ;also reset bits 5,6,7.
004B 80260501E3                   AND COLBYTE,0E3H ;reset bits 2,3,4.
0050 081E0501          COLSET:    OR COLBYTE,BL   ;new foreground or
                                                  ;background color
0054 8A1E0501                     MOV BL,COLBYTE
0058 BEDC44                       MOV SI,ATTRIB
                       ;
```

```
005B B84000                       MOV AX,40H
005E 8EC0                         MOV ES,AX      ;segment value for
                                                 ;offset of ATTRIB
                        ;
0060 26881C                       MOV ES:[SI],BL ;set ATTRIB byte in bios
0063 E99FFF        0005           JMP NEXT
0066 90                 DONE:     NOP
                        ;
                        ;
                        ; s u b r o u t i n e s
                        ;
0067 B132               DRCTBIOS: MOV CL,BIOSCALL ;bdos function number
0069 8D160001                     LEA DX,BIOSOUT  ;bios parameters' addr
006D C3                           RET
                        ;
006E B101               READIN:   MOV CL,CONIN   ;read keyboard
0070 CDE0                         INT 224
0072 C3                           RET
                        ;
0073 B102               CRTLF:    MOV CL,CONOUT  ;output to crt
0075 B20A                         MOV DL,LF
0077 CDE0                         INT 224
0079 C3                           RET
                        ;
007A E8EAFF        0067 CLRSCRN:  CALL DRCTBIOS      ;clear crt and
007D C60601011B                   MOV OUTCRT,CLSCRN1 ;cursor top left
0082 CDE0                         INT 224
0084 E8E0FF        0067           CALL DRCTBIOS
0087 C60601013A                   MOV OUTCRT,CLSCRN2
008C CDE0                         INT 224
008E C3                           RET
                        ;
008F E805FF        0067 ERASE:    CALL DRCTBIOS      ;erase last character
0092 C606010108                   MOV OUTCRT,CURBACK ;position on crt
0097 CDE0                         INT 224
0099 E8CBFF        0067           CALL DRCTBIOS
009C C606010120                   MOV OUTCRT,BLANK
00A1 CDE0                         INT 224
00A3 E8C1FF        0067           CALL DRCTBIOS
00A6 C606010108                   MOV OUTCRT,CURBACK
00AB CDE0                         INT 224
00AD C3                           RET
```

```
00AE B0FF               QCOLOR:   MOV AL,NOCHANGE ;checks for program
00B0 80FF40                       CMP BH,ATSIGN   ;terminate and color
00B3 740F        00C4             JE CHANGE       ;change request. If
00B5 80FF24                       CMP BH,DOLLAR   ;bx contains $$ then
00B8 7516        00D0             JNE ENDQ        ;terminate. @ or $ in
00BA 80FB24                       CMP BL,DOLLAR   ;bh indicates color
00BD 7505        00C4             JNE CHANGE      ;change, provided bl
00BF B001                         MOV AL,TERMIN   ;contains ASCII number
00C1 E90C00      00D0             JMP ENDQ        ;in range 0-7
00C4 80FB30             CHANGE:   CMP BL,ZERO
00C7 7207        00D0             JB ENDQ
00C9 80FB37                       CMP BL,SEVEN
00CC 7702        00D0             JA ENDQ
00CE B000                         MOV AL,COLCHANG
00D0 C3                 ENDQ:     RET
                        ;
                        ;
                                  DSEG
                                  ORG 100H
0100 04                 BIOSOUT   DB 4           ;bios function number
                                                 ;for console output.
0101 00000000           OUTCRT    DB 0,0,0,0     ;registers for bios
                                                 ;fnctn - cl ch dl dh,
                                                 ;only cl required.
0105 E8                 COLBYTE   DB 0E8H        ;intermediate storage of
                                                 ;foreground and
                                                 ;background color,
                                                 ;initialized to green
                                                 ;foreground with black
                                                 ;background
```

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."

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).

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 is 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.	

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.

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 full 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 5.11 summarizes the actual port addresses used by these interfaces.

2651 REGISTER ADDRESSING

Port (Hex)		Signals Required *				Function
K212 K213	K211	CE	BA0	BA1	BA2	
–	–	1	X	X	X	Tri-state data bus
60	70	0	0	0	0	Read receive holding register
64	74	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 - R̄/W.

Figure 5.11

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.

PORTS

The following is a summary of the available I/O ports used by the CP/M-86 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 in this section (Figure 5.12). It is important to note that certain ports, including the ports concerning this cautionary note, are reflected at other addresses.

OUT 10
Bit 0 set switches the first 2000H bytes of main memory into the address area 0-1FFFH.

OUT 11
Bit 0 set switches the firmware ROM into the address area 0-1FFFH.

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 FDC 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.

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.

LOW HIGH	0	1	2	3	4	5	6	7
0	ERROR LEDS							
1	RAMSEL	ROMSEL	SETTC	SYSSTAT	MOTOR			
2								
3	IFSEL 2A							
4	KEY: R/W DATA	KEY: R/W COMMAND						
5	FDC: R-MAIN STATUS	FDC: R/W DATA						
6	IFSEL 0 ADAPTERS K210, K212, K213							
7	IFSEL 1 ADAPTER K211							
8	TIMER: R/W COUNTER 0	TIMER: R/W COUNTER 1	TIMER: R/W COUNTER 2	TIMER: W- MODE				
9	Interrupt Controllers (Future)							
A	GDC R-STATUS W-PARAM	GDC R-DATA W-COMMAND	ZOOM					
B	IFSEL 3A							
C	IFSEL 4 WINCHESTER DISK							
D	16-BIT SWITCH							
E	64K RAM RAM BANKS 0 - 7	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM	64K RAM
F	I/O EXPANSION							

Figure 5.12 (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							
4								
5								
6	IFSEL 0							
7	IFSEL 1							
8								
9								
A								
B	IFSEL 3B NCR OMNINET							
C	IFSEL 4B							
D								
E								
F								

Figure 5.12 (2 of 2)

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 5.13 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 5.13

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 hardwarerelated 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).

Byte 0	start (L)			
Byte 1	start (M)			
Byte 2	len (L)	0	0	start (H)
Byte 3	WD	IM	len (H)	

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 (0 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.

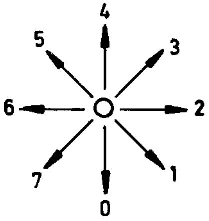
Figure – This command, using as many as 11 parameter bytes, is used for specifying whether individual dot or figure drawing is to take place, and in the latter case, it specifies the figure to be drawn. Beyond this, it is also used for determining the direction of activity for any screen writing. DMA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

Command byte: 4CH.

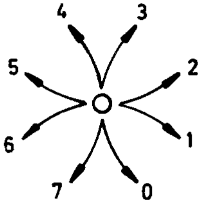
Byte 1	SL	R	A,C	G	L	Direction

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.

Your final source text immediately prior to assembly must contain the storage definitions (DB, DW, RS) in a separate data segment with the DSEG directive at its head. The stage by stage program construction in this section introduces each DB, DW, or RS at the time of discussion of the first routine which makes use of that particular storage definition. In this way, the data segment is built up gradually, as you work through the routines. Remember to include the ORG 100H directive at the beginning of the data segment.

```
                              DSEG
                              ORG 100H
0100 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. You can edit, assemble, and test the programs as described in the section "Some I/O Examples."

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.

```
                                    CSEG
0000 50                 OUTC:       PUSH AX
0001 E4A0               OUTC1:      IN AL,0A0H
0003 240A                           AND AL,0AH
0005 75FA         0001              JNZ OUTC1
0007 58                             POP AX
0008 E6A1                           OUT 0A1H,AL
000A 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.

```
000B E4A0               OUTP:       IN AL,0A0H
000D 240A                           AND AL,0AH
000F 75FA         000B              JNZ OUTP
0011 8A07               OUTP1:      MOV AL,BYTE PTR [BX]
0013 E6A0                           OUT 0A0H,AL
0015 43                             INC BX
0016 FECA                           DEC DL
0018 75F7         0011              JNZ OUTP1
001A C3                             RET
```

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

```
0102 00                 PRAMS       DB 0
0103 08                             DB 8
0104 000000590000       PRAMS1      DB 0,0,0,59H,0,0,0,59H,0FFH,0FFH,0FFH,
                                              0FFH,0FFH,0FFH,0FFH,0FFH
     0059FFFFFFFF
     FFFFFFFF
0114 000000             PRAMS2      DB 0,0,0
0117 FFFF               PRAMS3      DB 0FFH,0FFH
0119 02FF7F080008       PRAMS4      DB 2,0FFH,7FH,8,0,8,0,0FFH,3FH,0FFH,3FH
     00FF3FFF3F
0124 FFFF               PRAMS5      DB 0FFH,0FFH
0126 21                 WRLOGIC     DB 21H        ;complement
```

GINIT is the routine which transmits these parameters:

```
001B 8D1E0201          GINIT:    LEA BX,PRAMS
001F B00C                        MOV AL,0CH     ;bit 0 blanks screen
0021 E8DCFF        0000          CALL OUTC
0024 B046                        MOV AL,46H     ;set zoom to zero
0026 E8D7FF        0000          CALL OUTC
0029 B201                        MOV DL,i
002B E8DDFF        000B          CALL OUTP
002E B04B                        MOV AL,4BH     ;cursor/char
                                                ;characteristics.
0030 E8CDFF        0000          CALL OUTC
0033 B201                        MOV DL,1       ;parameter sets lines
                                                ;per row to zero.
0035 E8D3FF        000B          CALL OUTP
0038 B070                        MOV AL,70H     ;load entire
                                                ;parameter RAM.
003A E8C3FF        0000          CALL OUTC
003D B210                        MOV DL,10H
003F 8D1E0401                    LEA BX,PRAMS1
0043 E8C5FF        000B          CALL OUTP      ;sets graphics and
                                                ;400 pixels vertical.
0046 B049                        MOV AL,49H     ;set cursor pos
0048 E8B5FF        0000          CALL OUTC
004B B203                        MOV DL,3
004D 8D1E1401                    LEA BX,PRAMS2
0051 E8B7FF        000B          CALL OUTP      ;first pixel addressed
0054 B04A                        MOV AL,4AH     ;set mask
0056 E8A7FF        0000          CALL OUTC
0059 B202                        MOV DL,2
005B 8D1E1701                    LEA BX,PRAMS3
005F E8A9FF        000B          CALL OUTP
0062 B04C                        MOV AL,4CH     ;figure parameters
0064 E899FF        0000          CALL OUTC
0067 B20B                        MOV DL,0BH
0069 8D1E1901                    LEA BX,PRAMS4
006D E89BFF        000B          CALL OUTP      ;no geom. figs,
                                                ;direction east.
0070 B022                        MOV AL,22H     ;write data word high
                                                ;then low, reset to 0.
0072 E88BFF        0000          CALL OUTC
0075 B202                        MOV DL,2
0077 8D1E2401                    LEA BX,PRAMS5
007B E88DFF        000B          CALL OUTP
```

```
007E B021                     MOV AL,21H     ;write data,
                                             ;this time complement.
0080 A22601                   MOV WRLOGIC,AL
0083 E87AFF       0000        CALL OUTC
0086 B00D                     MOV AL,0DH     ;re-enable screen
0088 E875FF       0000        CALL OUTC
008B E80500       0093 WAIT:  CALL GETKEY
008E 3C24                     CMP AL,'$'
0090 75F9         008B        JNE WAIT
0092 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 byte 1. 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 WRMODE. 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.

```
0093 53                GETKEY:  PUSH BX
0094 51                         PUSH CX
0095 52                         PUSH DX
0096 B106                       MOV CL,6
0098 B2FF                       MOV DL,0FFH
009A CDE0                       INT 224
009C 5A                         POP DX
```

```
009D 59                           POP CX
009E 5B                           POP BX
009F 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 for the data segment starting at EXPRAMS are used by the exit routine GEXIT.

```
0127 8F00               EXPRAMS    DB 8FH,0
0129 0090000100FF       EXPRAMS1   DB 0,90H,0,1,0,0FFH,0FFH,0FFH,0FFH,0FFH,
     FFFFFFFFFFFF                                               0FFH,0FFH,0FFH
     FF
 00A0 8D1E2701           GEXIT:    LEA BX,EXPRAMS
 00A4 B04B                         MOV AL.4BH
 00A6 E857FF        0000           CALL OUTC
 00A9 B201                         MOV DL,1
 00AB E85DFF        000B           CALL OUTP
 00AE B046                         MOV AL.46H
 00B0 E84DFF        0000           CALL OUTC
 00B3 B201                         MOV DL,1
 00B5 E853FF        000B           CALL OUTP
 0088 B070                         MOV AL,70H
 00BA E843FF        0000           CALL OUTC
 00BD B20D                         MOV DL,0DH
 00BF E849FF        000B           CALL OUTP
 00C2 B21A           CLSCRN:       MOV DL.1AH
 00C4 B102                         MOV CL,2
 00C6 50                           PUSH AX
 00C7 53                           PUSH BX
 00C8 51                           PUSH CX
 00C9 52                           PUSH DX
 00CA CDE0                         INT 224
 00CC 5A                           POP DX
 00CD 59                           POP CX
 00CE 5B                           POP BX
 00CF 58                           POP AX
 00D0 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 in the data segment 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.

```
0136 20                    CURPRAMS  DB 20H
0137 20                              DB 20H
0138 00                              DB 0
0DD1 B049                  CURSET:   MOV AL,49H
00D3 E82AFF        0000              CALL OUTC
00D6 8D1E3601                        LEA BX,CURPRAMS
00DA B203                            MOV DL,3
00DC E82CFF        000B              CALL OUTP
00DF C3                              RET
```

Now reserve an area for storing figure drawing parameters:

```
                           ;
0139 000000000000          FIGPRAMS  DB 0,0,0,0,0,0,0,0,0,0,0
     0000000000
```

Enter the routine for transmitting these parameters to the GDC

```
00E0 B04C                  FIGSET:   MOV AL,4CH
00E2 E81BFF        0000              CALL OUTC
00E5 8D1E3901                        LEA BX,FIGPRAMS
00E9 B20B                            MOV DL,0BH
00EB E81DFF        000B              CALL OUTP
00EE C3                              RET
```

and the command which sets drawing in progress.

```
00EF B06C                  FIGDRAW:  MOV AL,6CH
00F1 E80CFF        0000              CALL OUTC
00F4 C3                              RET
```

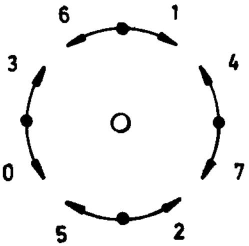
All that is now required are actual parameters for figure drawing. The following can be used for drawing a square:

```
                        :
0144 400340300030       FIGPRAM1   DB 40H,3,40H,30H,0,30H,0,0FFH,3FH,30H,0
     00FF3F3000
```

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.

```
00F5 8D1E4401                  LEA BX,FIGPRAM1
00F9 8D3E3901                  LEA DI,FIGPRAMS
00FD B10B                      MOV CL,LENGTH FIGPRAMS
00FF 8A07            NEXTPR1:  MOV AL,BYTE PTR [BX]
0101 8805                      MOV BYTE PTR [DI],AL
0103 43                        INC BX
0104 47                        INC DI
0105 FEC9                      DEC CL
0107 75F6       00FF           JNZ NEXTPR1
0109 E80FFF     001B           CALL GINIT
010C E8C2FF     00D1           CALL CURSET
010F E8CEFF     00E0           CALL FIGSET
0112 E8DAFF     00EF           CALL FIGDRAW
0115 E87BFF     0093 WAIT2:    CALL GETKEY
0118 3C78                      CMP AL,'x'
011A 75F9       0115           JNE WAIT2
011C E881FF     00A0           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:

```
014F 40BE          MIDDLE     DW 0BE40H
0151 01            MIDDLEH    DB 1
0152 32            RADIUS     DB 50
0153 0000          NORTH      DW 0
0155 00            NORTHH     DB 0
0156 0000          SOUTH      DW 0
0158 00            SOUTHH     DB 0
0159 0000          EAST       DW 0
015B 00            EASTH      DB 0
015C 0000          WEST       DW 0
015E 00            WESTH      DB 0
015F 0000          PIXEL      DW 0
0161 00            PIXELH     DB 0
0162 00            CURSL      DB 0
0163 00            CURSH      DB 0
0164 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.

```
011F 53            COMPASS:   PUSH BX
0120 51                       PUSH CX
0121 52                       PUSH DX
0122 BA8002                   MOV DX,280H                 ;pitch
0125 A14F01        CNORTH:    MOV AX,WORD PTR MIDDLE
```

```
0128 8A0E5201                     MOV CL,RADIUS
012C 32ED                         XOR CH,CH
012E 8A1E5101                     MOV BL,MIDDLEH
0132 F8                 NDCR:     CLC
0133 1BC2                         SBB AX,DX
0135 80DB00                       SBB BL,0
0138 E2F8         0132            LOOP NDCR
013A A35301                       MOV WORD PTR NORTH,AX
013D 881E5501                     MOV NORTHH,BL
0141 A14F01             CSOUTH:   MOV AX,WORD PTR MIDDLE
0144 8A0E5201                     MOV CL,RADIUS
0148 32ED                         XOR CH,CH
014A 8A1E5101                     MOV BL,MIDDLEH
014E F8                 SDCR:     CLC
014F 13C2                         ADC AX,DX
0151 80D300                       ADC BL,0
0154 E2F8         014E            LOOP SDCR
0156 A35601                       MOV WORD PTR SOUTH,AX
0159 881E5801                     MOV SOUTHH,BL
015D A14F01             CEAST:    MOV AX,WORD PTR MIDDLE
0160 8A0E5201                     MOV CL,RADIUS
0164 32ED                         XOR CH,CH
0166 8A1E5101                     MOV BL,MIDDLEH
016A F8                 EDCR:     CLC
016B 150100                       ADC AX,1
016E 80D300                       ADC BL,0
0171 E2F7         016A            LOOP EDCR
0173 A35901                       MOV WORD PTR EAST,AX
0176 881E5B01                     MOV EASTH,BL
017A A14F01             CWEST:    MOV AX,WORD PTR MIDDLE
017D 8A0E5201                     MOV CL,RADIUS
0181 32ED                         XOR CH,CH
0183 8A1E5101                     MOV BL,MIDDLEH
0187 F8                 WDCR:     CLC
0188 1D0100                       SBB AX,1
018B 80DB00                       SBB BL,0
018E E2F7         0187            LOOP WDCR
0190 A35C01                       MOV WORD PTR WEST,AX
0193 881E5E01                     MOV WESTH,BL
0197 5A                           POP DX
0198 59                           POP CX
0199 58                           POP AX
019A 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.

```
019B 53            WORDAD:    PUSH BX
019C 51                       PUSH CX
019D 52                       PUSH DX
019E A15F01                   MOV AX,WORD PTR PIXEL
01A1 8AD0                     MOV DL,AL
01A3 B104                     MOV CL,4
01A5 D3E8                     SHR AX,CL
01A7 8A366101                 MOV DH,PIXELH
01AB D2E2                     SHL DL,CL
01AD D2E6                     SHL DH,CL
01AF 0AE6                     OR AH,DH
01B1 88266301                 MOV CURSH,AH
01B5 A26201                   MOV CURSL,AL
01B8 88166401                 MOV DOTPOS,DL
01BC 5A                       POP DX
01BD 59                       POP CX
01BE 5B                       POP BX
01BF 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.

```
01C0 8D1E3601      CURTRANSF: LEA BX,CURPRAMS
01C4 A16201                   MOV AX,WORD PTR CURSL
01C7 8907                     MOV WORD PTR [BX],AX
01C9 43                       INC BX
01CA 43                       INC BX
01CB A06401                   MOV AL,DOTPOS
01CE 8807                     MOV BYTE PTR [BX],AL
01D0 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:

```
01D1 E847FE      001B           CALL GINIT
01D4 E848FF      011F           CALL COMPASS
01D7 A15301                     MOV AX,WORD PTR NORTH
01DA A35F01                     MOV WORD PTR PIXEL,AX
01DD A05501                     MOV AL,BYTE PTR NORTHH
01E0 A26101                     MOV BYTE PTR PIXELH,AL
01E3 E885FF      019B           CALL WORDAD
01E6 E8D7FF      01C0           CALL CURTRANSF
01E9 E8E5FE      00D1           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.

```
01EC 8D1E3901                   LEA BX,FIGPRAMS
01F0 C60721                     MOV BYTE PTR [BX],21H
                                          ;type of drawing = arc,
                                          ;direction = 1.
01F3 43                         INC BX
01F4 C60723                     MOV BYTE PTR [BX],23H
                                          ;rsin 45 for radius
                                          ;50 pixels
01F7 43                         INC BX
01F8 C60740                     MOV BYTE PTR [BX],40H
                                          ;graphics drawing flag
01FB 43                         INC BX
```

```
01FC C60731                         MOV BYTE PTR [BX],31H
                                            ;one less than radius
01FF 43                             INC BX
0200 C60700                         MOV BYTE PTR [BX],0
0203 43                             INC BX
0204 C60762                         MOV BYTE PTR [BX],62H
0207 43                             INC BX
0208 C60700                         MOV BYTE PTR [BX],0
020B 43                             INC BX
020C C607FF                         MOV BYTE PTR [BX],0FFH
020F 43                             INC BX
0210 C6073F                         MOV BYTE PTR [BX],3FH
0213 43                             INC BX
0214 C60700                         MOV BYTE PTR [BX],0
0217 43                             INC BX
0218 C60700                         MOV BYTE PTR [BX],0
                             :
021B E8C2FE       00E0              CALL FIGSET
021E E8CEFE       00EF              CALL FIGDRAW
```

Then follows the Direction 6 arc:

```
0221 E8ADFE       00D1              CALL CURSET
0224 C606390126                     MOV BYTE PTR FIGPRAMS,26H
0229 E8B4FE       00E0              CALL FIGSET
022C E8C0FE       00EF              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.

```
022F A15601                         MOV AX,WORD PTR SOUTH
0232 A35F01                         MOV WORD PTR PIXEL,AX
0235 A05801                         MOV AL,BYTE PTR SOUTHH
0238 A26101                         MOV BYTE PTR PIXELH,AL
023B E85DFF       019B              CALL WORDAD
023E E87FFF       01C0              CALL CURTRANSF
0241 E88DFE       00D1              CALL CURSET
0244 C606390122                     MOV BYTE PTR FIGPRAMS,22H
0249 E894FE       00E0              CALL FIGSET
024C E8A0FE       00EF              CALL FIGDRAW
024F E87FFE       00D1              CALL CURSET
```

```
0252 C606390125                     MOV BYTE PTR FIGPRAMS,25H
0257 E886FE       00E0              CALL FIGSET
025A E892FE       00EF              CALL FIGDRAW
                        ;
025D A15901                         MOV AX,WORD PTR EAST
0260 A35F01                         MOV WORD PTR PIXEL,AX
0263 A05B01                         MOV AL,BYTE PTR EASTH
0266 A26101                         MOV BYTE PTR PIXELH,AL
0269 E82FFF       019B              CALL WORDAD
026C E851FF       01C0              CALL CURTRANSF
026F E85FFE       00D1              CALL CURSET
0272 C606390124                     MOV BYTE PTR FIGPRAMS,24H
0277 E866FE       00E0              CALL FIGSET
027A E872FE       00EF              CALL FIGDRAW
027D E851FE       00D1              CALL CURSET
0280 C606390127                     MOV BYTE PTR FIGPRAMS,27H
0285 E858FE       00E0              CALL FIGSET
0288 E864FE       00EF              CALL FIGDRAW
                        ;
028B A15C01                         MOV AX,WORD PTR WEST
028E A35F01                         MOV WORD PTR PIXEL,AX
0291 A05E01                         MOV AL,BYTE PTR WESTH
0294 A26101                         MOV BYTE PTR PIXELH,AL
0297 E801FF       019B              CALL WORDAD
029A E823FF       01C0              CALL CURTRANSF
029D E831FE       00D1              CALL CURSET
02A0 C606390120                     MOV BYTE PTR FIGPRAMS,20H
02A5 E838FE       00E0              CALL FIGSET
02A8 E844FE       00EF              CALL FIGDRAW
02AB E823FE       00D1              CALL CURSET
02AE C606390123                     MOV BYTE PTR FIGPRAMS,23H
02B3 E82AFE       00E0              CALL FIGSET
02B6 E836FE       00EF              CALL FIGDRAW
```

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

```
02B9 E8D7FD       0093 WAIT3:       CALL GETKEY
02BC 3C78                           CMP AL,'x'
02BE 75F9         02B9              JNE WAIT3
02C0 E8DDFD       00A0              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 FIG-PRAMS 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 WR-MODE (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.

```
02C3 E8C0FD     0093 CALCUL:    CALL GETKEY
02C6 32D2                       XOR DL,DL
02C8 3C30                       CMP AL,'0'
02CA 7427       02F3            JE OVER
02CC 3C35                       CMP AL,'5'
02CE 743F       030F            JE ONOFF
02D0 3C31                       CMP AL,'1'
02D2 7420       02F4            JE DIR7
02D4 3C32                       CMP AL,'2'
02D6 742A       0302            JE DIR0
02D8 3C33                       CMP AL,'3'
02DA 7424       0300            JE DIR1
02DC 3C34                       CMP AL,'4'
02DE 7416       02F6            JE DIR6
02E0 3C36                       CMP AL,'6'
02E2 741A       02FE            JE DIR2
02E4 3C37                       CMP AL,'7'
02E6 7410       02F8            JE DIR5
02E8 3C38                       CMP AL,'8'
02EA 740E       02FA            JE DIR4
02EC 3C39                       CMP AL,'9'
02EE 740C       02FC            JE DIR3
02F0 E900FF     02C3            JMP CALCUL
02F3 C3              OVER:      RET
02F4 FEC2            DIR7:      INC DL
02F6 FEC2            DIR6:      INC DL
```

```
02F8 FEC2               DIR5:     INC DL
02FA FEC2               DIR4:     INC DL
02FC FEC2               DIR3:     INC DL
02FE FEC2               DIR2:     INC DL
0300 FEC2               DIR1:     INC DL
0302 88163901           DIR0:     MOV BYTE PTR FIGPRAMS,DL
0306 E8D7FD      00E0             CALL FIGSET
0309 E8E3FD      00EF             CALL FIGDRAW
030C E9B4FF      02C3             JMP CALCUL
                        ;
030F A02601             ONOFF:    MOV AL,BYTE PTR WRLOGIC
0312 3401                         XOR AL,1
0314 A22601                       MOV BYTE PTR WRLOGIC,AL
0317 E8E6FC      0000             CALL OUTC
031A E9A6FF      02C3             JMP CALCUL
```

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

```
0165 000040080008      FIGPRAM2   DB 0,0,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
     00FF3FFF3F
```

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.

```
031D 8D1E6501                     LEA BX,FIGPRAM2
0321 8D3E3901                     LEA DI,FIGPRAMS
0325 B10B                         MOV CL,LENGTH FIGPRAMS
0327 8A07               NEXTPR2:  MOV AL,BYTE PTR [BX]
0329 8805                         MOV BYTE PTR [DI],AL
032B 43                           INC BX
032C 47                           INC DI
032D FEC9                         DEC CL
032F 75F6        0327             JNZ NEXTPR2
0331 E8E7FC      001B             CALL GINIT
```

```
0334 E89AFD      00D1          CALL CURSET
0337 C606260123                MOV BYTE PTR WRLOGIC,23H
033C B023                      MOV AL,23H
033E E8BFFC      0000          CALL OUTC
0341 E87FFF      02C3          CALL CALCUL
0344 E84CFD      0093 WAIT4:   CALL GETKEY
0347 3C78                      CMP AL,'x'
0349 75F9        0344          JNE WAIT4
034B C606260121                MOV BYTE PTR WRLOGIC,21H
0350 B021                      MOV AL,21H
0352 E8ABFC      0000          CALL OUTC
                                          ;resets to complement
                                          ;from any setting.
0355 E92801      0480          JMP SAVEIT ;JMP.SAVED
0358 E845FD      00A0 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. Even the operating system is situated above this critical address (see "How to read the BIOS Program"). However, you should bear in mind that the 8086 interrupt vector is not accessible while the ROM is selected. This means that INT 224 would cause loss of program control. Therefore, you must de-select the ROM before using the BDOS functions. 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:

```
0170                 CHSTORE    RS 16
```

The following routine, ASCII, fetches a 16 x 8 bit pattern from the ROM and deposits it in the 16-byte storage area CHSTORE. 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.

```
035B 53                 ASCII:    PUSH BX
035C 51                           PUSH CX
035D 52                           PUSH DX
035E B210                         MOV DL,10H
0360 F6E2                         MUL DL          ;code already in AL
                                                  ;at calling.
0362 050010                       ADD AX,1000H     ;address of char
                                                   ;in ROM now in AX.
0365 8BD8                         MOV BX,AX
0367 8D3E7001                     LEA DI,CHSTORE
036B B91000                       MOV CX,10H
036E BA0000                       MOV DX,0
0371 8EC2                         MOV ES,DX
0373 E611                         OUT 11H,AL
0375 268A07             ROMBYTE:  MOV AL,ES:BYTE PTR [BX]
0378 8805                         MOV BYTE PTR [DI],AL
037A 43                           INC BX
037B 47                           INC DI
037C E2F7         0375            LOOP ROMBYTE
037E E610                         OUT 10H,AL
0380 5A                           POP DX
0381 59                           POP CX
0382 5B                           POP BX
0383 C3                           RET
```

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

```
0384 B037                         MOV AL,'7'
0386 E8D2FF       035B            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 comfortabel 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.

```
0389 B046                ZOOM:      MOV AL,46H
038B E872FC       0000              CALL OUTC
038E 8D1E9501                       LEA BX,ZOOMFACT
0392 B201                           MOV DL,1
0394 E874FC       000B              CALL OUTP
0397 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.

```
0180                     CHARMIR    RS 8
0188 005A427E3C24        CHARPATT   DB 0,5AH,42H,7EH,3CH,24H,24H,42H
     2442
                                                   ;random example
                         ;
0398 8D1E8801            SKEW:      LEA BX,CHARPATT
039C 83C307                         ADD BX,7
039F 8D3E8001                       LEA DI,CHARMIR
03A3 B90800                         MOV CX,8
03A6 8A07                NEXTCH:    MOV AL,BYTE PTR [BX]
03A8 E80700       03B2              CALL MIRROR
                                                   ;to cancel mirror,
                                                   ;replace CALL instruc-
                                                   ;tion by three NOPs.
03AB 8805                           MOV BYTE PTR [DI],AL
03AD 4B                             DEC BX
03AE 47                             INC DI
03AF E2F5         03A6              LOOP NEXTCH
03B1 C3                             RET
                         ;
```

```
03B2 53               MIRROR:    PUSH BX       ;the bits of the AL
03B3 51                          PUSH CX       ;register are mirrored
03B4 52                          PUSH DX       ;around an imaginary
03B5 32F6                        XOR DH,DH     ;axis between bits 3
03B7 B201                        MOV DL,1      ;and 4. Thus bits 0
03B9 B101                        MOV CL,1      ;and 7 exchange posit-
03BB 8AD8                        MOV BL,AL     ;ions, as do bits 1
03BD 32E4             NEXTSHFT:  XOR AH,AH     ;and 6, and so on.
03BF 8AC3                        MOV AL,BL
03C1 D3E0                        SHL AX,CL
03C3 22E2                        AND AH,DL
03C5 0AF4                        OR DH,AH
03C7 D0E2                        SHL DL,1
03C9 80C102                      ADD CL,2
03CC 80F911                      CMP CL,11H
03CF 75EC         03BD           JNE NEXTSHFT
03D1 8AC6                        MOV AL,DH
03D3 5A                          POP DX
03D4 59                          POP CX
03D5 5B                          POP BX
03D6 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

```
0190 1607400700        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 place by CHAROUT in ZOOMFACT.

```
0195 04                ZOOMFACT   DB 4
03D7 B078              CHAROUT:   MOV AL,78H    ;starter pRAM at parm 8.
03D9 E824FC       0000            CALL OUTC
03DC 8D1E8001                     LEA BX,CHARMIR
03E0 B208                         MOV DL,8
03E2 E826FC       000B            CALL OUTP
03E5 B04C                         MOV AL,4CH    ;figset
```

```
03E7 E816FC      0000               CALL OUTC
03EA 8D1E9001                       LEA BX,CHFGPRAM
03EE B205                           MOV DL,5
03F0 E818FC      000B               CALL OUTP
03F3 C606950104                     MOV BYTE PTR ZOOMFACT,4
03F8 E88EFF      0389               CALL ZOOM
03FB B068                           MOV AL,68H     ;draw graphic char
03FD E800FC      0000               CALL OUTC
0400 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.

```
0401 E817FC      001B               CALL GINIT
0404 E8CAFC      0001               CALL CURSET
0407 B037                           MOV AL,'7'
0409 E84FFF      035B               CALL ASCII
040C 8D1E7001                       LEA BX,CHSTORE
0410 43                             INC BX
0411 43                             INC BX
0412 43                             INC BX
0413 8D3E8801                       LEA DI,CHARPATT
0417 B90800                         MOV CX,8
041A 8A07                 NEXTCOP:  MOV AL,BYTE PTR [BX]
041C 8805                           MOV BYTE PTR [DI],AL
041E 43                             INC BX
041F 47                             INC DI
```

```
0420 E2F8          041A            LOOP NEXTCOP
0422 E873FF        0398            CALL SKEW
0425 E8AFFF        03D7 REPEAT:    CALL CHAROUT
0428 E868FC        0093 WAIT5:     CALL GETKEY
042B 3C72                          CMP AL,'r'
042D 74F6          0425            JE REPEAT
042F 3C78                          CMP AL,'x'
0431 75F5          0428            JNE WAIT5
0433 E86AFC        00A0            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:

```
0196 FFFF                 PRAMSR     DB 0FFH,0FFH
0198 FFFF                 RMASK      DB 0FFH,0FFH
019A 020840080008         FIGSR      DB 2,8,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
     00FF3FFF3F
01A5 02                   MASKFIG    DB 2
01A6                      SCREEN     RS 16000
4026 FFFF                 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.

```
0436 51                    READSCRN:  PUSH CX
0437 B90800                           MOV CX,8
043A B202                  NEXTWORD:  MOV DL,2
043C E4A0                  READYCHK:  IN AL,0A0H
043E 2401                             AND AL,1
0440 74FA          043C               JZ READYCHK
0442 E4A1                             IN AL,0A1H
0444 8805                             MOV BYTE PTR [DI],AL
0446 47                               INC DI
0447 FECA                             DEC DL
0449 75F1          043C               JNZ READYCHK
044B E2ED          043A               LOOP NEXTWORD
044D 59                               POP CX
044E 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.

```
044F B0A0                  FIFOCLR:   MOV AL,0A0H
0451 E8ACFB        0000               CALL OUTC
0454 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.

```
0455 B078               SETREAD:   MOV AL,78H
0457 E8A6FB      0000              CALL OUTC     ;set pRAM
045A 8D1E9601                      LEA BX,PRAMSR
045E B202                          MOV DL,2
0460 E8A8FB      000B              CALL OUTP
0463 B04A                          MOV AL,4AH
0465 E898FB      0000              CALL OUTC     ;set mask
0468 8D1E9801                      LEA BX,RMASK
046C B202                          MOV DL,2
046E E89AFB      000B              CALL OUTP
0471 B04C                          MOV AL,4CH
0473 E88AFB      0000              CALL OUTC     ;set fig
0476 8D1E9A01                      LEA BX,FIGSR
047A B20B                          MOV DL,0BH
047C E88CFB      000B              CALL OUTP
047F 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.

```
0480 8D1E3601           SAVEIT:    LEA BX,CURPRAMS
0484 C7070000                      MOV WORD PTR [BX],0
0488 43                            INC BX
0489 C7070000                      MOV WORD PTR [BX],0
048D E841FC      0001              CALL CURSET
0490 8D3EA601                      LEA DI,SCREEN
0494 B9E803                        MOV CX,03E8H
0497 E8BBFF      0455 NEXTSCRN:    CALL SETREAD
049A E8B2FF      044F              CALL FIFOCLR
049D E896FF      0436              CALL READSCRN
04A0 E2F5        0497              LOOP NEXTSCRN
04A2 E8FBFB      00A0              CALL GEXIT
```

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

```
04A5 E86500       050D            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.

```
04A8 8D1E6501          PAINT:     LEA BX,FIGPRAM2
04AC 8D3E3901                     LEA DI,FIGPRAMS
04B0 B90B00                       MOV CX,LENGTH FIGPRAMS
04B3 8A07              NEXTPR3:   MOV AL,BYTE PTR [BX]
04B5 8805                         MOV BYTE PTR [DI],AL
04B7 43                           INC BX
04B8 47                           INC DI
04B9 E2F8         04B3            LOOP NEXTPR3
04BB C606390102                   MOV BYTE PTR FIGPRAMS,2
04C0 E858FB       001B            CALL GINIT
04C3 E80BFC       00D1            CALL CURSET
04C6 E817FC       00E0            CALL FIGSET
04C9 8D3EA601                     LEA DI,SCREEN
04CD B9803E                       MOV CX,3E80H
04D0 51                NEWBYTE:   PUSH CX
04D1 B90800                       MOV CX,8
04D4 8A25                         MOV AH,BYTE PTR [DI]
04D6 D0EC              CHECKBIT:  SHR AH,1
04D8 7205         04DF            JC PLOT
04DA B022                         MOV AL,22H
04DC E90200       04E1            JMP LOGICSET
04DF B023              PLOT:      MOV AL,23H
04E1 E81CFB       0000 LOGICSET:  CALL OUTC
04E4 B04C                         MOV AL,4CH
04E6 E817FB       0000            CALL OUTC
```

```
04E9 8D1E3901                     LEA BX,FIGPRAMS
04ED B203                         MOV DL,3
04EF E819FB       000B            CALL OUTP
04F2 B06C                         MOV AL,6CH
04F4 E809FB       0000            CALL OUTC
04F7 E2DD         04D6            LOOP CHECKBIT
04F9 47                           INC DI
04FA 59                           POP CX
04FB E2D3         04D0            LOOP NEWBYTE
04FD B021                         MOV AL,21H
04FF E8FEFA       0000            CALL OUTC
0502 E88EFB       0093 WAIT6:     CALL GETKEY
0505 3C78                         CMP AL,'x'
0507 75F9         0502            JNE WAIT6
0509 E894FB       00A0            CALL GEXIT
050C 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.

```
050D E883FB       0093 ADJUST:    CALL GETKEY
0510 3C00                         CMP AL,0
0512 74F9         050D            JE ADJUST
0514 3C69                         CMP AL,'i'
0516 7405         051D            JE ADJUST1
0518 3C6D                         CMP AL,'m'
051A 7413         052F            JE ADJUST2
051C 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.

```
051D 8D3EA601          ADJUST1:   LEA DI,SCREEN
0521 B9401F                       MOV CX,1F40H
0524 8B05              ADJUST11:  MOV AX,WORD PTR [DI]
0526 F7D0                         NOT AX
0528 8905                         MOV WORD PTR [DI],AX
```

```
052A 47                                INC DI
052B 47                                INC DI
052C E2F6           0524               LOOP ADJUST11
052E 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 with 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.

```
052F 8D1EA601             ADJUST2:     LEA BX,SCREEN
0533 4B                                DEC BX
0534 B9C800                            MOV CX,200
0537 51                   NEXTLINE:    PUSH CX
0538 B92800                            MOV CX,40
053B BF2800                            MOV DI,40
053E BE2900                            MOV SI,41
0541 8A01                 LINESWOP:    MOV AL,BYTE PTR [BX+DI]
0543 8A20                              MOV AH,BYTE PTR [BX+SI]
0545 8821                              MOV BYTE PTR [BX+DI],AH
0547 8800                              MOV BYTE PTR [BX+SI],AL
0549 4F                                DEC DI
054A 46                                INC SI
054B E2F4           0541               LOOP LINESWOP
054D 59                                POP CX
054E 83C350                            ADD BX,80
0551 E2E4           0537               LOOP NEXTLINE
0553 8D3EA601                          LEA DI,SCREEN
0557 B9803E                            MOV CX,3E80H
055A 8A05                 ADJUST21:    MOV AL,BYTE PTR [DI]
055C E853FE         03B2               CALL MIRROR
055F 8805                              MOV BYTE PTR [DI],AL
0561 47                                INC DI
0562 E2F6           055A               LOOP ADJUST21
0564 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:

```
0565 E8A5FF      050D               CALL ADJUST
0568 E8B0FA      001B               CALL GINIT
056B B04C                           MOV AL,4CH
056D E890FA      0000               CALL OUTC
0570 8D1EA501                       LEA BX,MASKFIG
0574 B201                           MOV DL,1
0576 E892FA      000B               CALL OUTP
0579 E855FB      00D1               CALL CURSET
057C 8D1EA601                       LEA BX,SCREEN
0580 8D3E2640                       LEA DI,DUMBYTES
0584 B9401F                         MOV CX,1F40H
0587 B04A             NEXTMASK:     MOV AL,4AH
0589 E874FA      0000               CALL OUTC
058C B202                           MOV DL,2
058E E87AFA      000B               CALL OUTP
0591 B023                           MOV AL,23H
0593 E86AFA      0000               CALL OUTC
0596 87DF                           XCHG BX,DI
0598 B202                           MOV DL,2
059A E86EFA      000B               CALL OUTP
059D 4B                             DEC BX
059E 4B                             DEC BX
059F 87DF                           XCHG BX,DI
05A1 E2E4        0587               LOOP NEXTMASK
05A3 E8EDFA      0093 WAIT7:        CALL GETKEY
05A6 3C78                           CMP AL,'x'
05A8 75F9        05A3               JNE WAIT7
05AA E8F3FA      00A0               CALL GEXIT
05AD 90                             NOP
```

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 96KB, instead of the 32KB 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 32KB boundaries in the 96KB graphic display memory. Even if you wish to confine pixel writing and drawing to green on black (the first 32KB govern the green gun, the next 32KB the red gun, and the last 32KB 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.

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 D

THE BIOS PROGRAM

This Appendix contains the 8086 assembly language listing of the CP/M-86 BIOS for your NCR DECISION MATE V. Remember that the machine addresses used are offsets to a paragaraph value of 40H. This means that the first machine address of the BIOS program is 2500H bytes above the top of the 1024 byte interrupt vector. In the case of call and jump instructions, the assembler has provided the new value of the Instruction Pointer in a separate column, that is, the value of the Instruction Pointer once the jump has been taken.

Immediately following the BIOS program in this Appendix is a list of symbols, including the resolution of these symbols, and cross-references.

The following table of contents to the BIOS program provides you with a means of quick reference to the I/O functions.

```


  FFFF               TRUE       EQU     -1
  0000               FALSE      EQU     NOT TRUE
                     ;
  0000               LOADER_BIOS EQU    FALSE
                     ;
                     ;
                             IF NOT LOADER_BIOS
  0000               CCPOFFSET  EQU     0
  0B06               BDOS_OFST  EQU     0B06H   ;BDOS ENTRY POINT
  00E0               BDOS_INT   EQU     224     ;RESERVED BDOS INTERRUPT
  2500               BIOS_CODE  EQU     2500H
                             ENDIF

  00DE               BIOS_INT   EQU     222     ;BIOS INTERRUPT
                     ;
                     ;
                             IF LOADER_BIOS
                     CCPOFFSET  EQU     3
                     BDOS_OFST  EQU     0406H   ;BDOS ENTRY POINT
                     BDOS_INT   EQU     224     ;RESERVED BDOS INTERRUPT
                     BIOS_CODE  EQU     1200H
                             ENDIF
                     ;
                             CSEG
                             ORG    CCPOFFSET
                     CCP:
                             OR6    BIOS_CODE
                     ;
                     ;  BIOS JUMP VECTOR
                     ;
2500 E9D705     2ADA         JMP    INIT
2503 E98C06     2B92         JMP    WBOOT
2506 E9B306     2BBC         JMP    CONST
2509 E9CE06     2BDA         JMP    CONIN
250C E9BC06     2BCB         JMP    CONOUT
250F E90107     2C13         JMP    LISTOUT
2512 E9E706     2BFC         JMP    PUNCH
2515 E9D106     2BE9         JMP    READER
2518 E97210     358D         JMP    HOME
2518 E98210     35A0         JMP    SELDSK
251E E9BF11     36E0         JMP    SETTRK
2521 E9C111     36E5         JMP    SETSEC
2524 E9C311     36EA         JMP    SETDMA
2527 E9D811     3702         JMP    READ
252A E9EC11     3719         JMP    WRITE
2520 E9F206     2C22         JMP    LISTST
2530 E9C111     36F4         JMP    SECTRAN
2533 E98911     36EF         JMP    SETDMAB
2536 E91107     2C4A         JMP    GETSEGT
2539 E90307     2C3F         JMP    GETIOBF
253C E90507     2C44         JMP    SETIOBF

253F E98415     3AC6         JMP    SPECFUN
2542 E95316     3B98         JMP    SELTYP
```

```


                       ;******************************************************
                       ;******************************************************
                       ;******************************************************
                       ;***                                                ***
                       ;***              CONFIG AREA                       ***
                       ;***                                                ***
                       ;******************************************************
                       ;******************************************************
                       ;******************************************************

                               IF NOT LOADER_BIOS
  002B                 FILLER  EQU     2570H - OFFSET $
                               ENDIF
                       ;
                       ;
                               IF LOADER_BIOS
                       FILLER  EQU     1270H - OFFSET $
                               ENDIF
                       ;
                       ;
2545                           RS      FILLER
2570 30312E30302E      REL_ID  DB      '01.00.04'              ; RELEASE ID
     3034
2578 00                        DB      0
2579 303430383833              DB      '040883'
257F 00                DEBUG_FLG       DB      0               ; MUST BE FF IF SYSTEM LOADED WITH DDT86

2580 8A25              MMAREA  DW      SPAREA                  ; SPECIAL AREA
2582 BD25              MFNTBL  DW      FUNC_TABLE              ; START ADDRESS OF FUNCTION TABLE
2584 CE29              MCRTTBL DW      CRT_TABLE               ; START ADDRESS OF CRT TABLE
2586 BA2A              MKEYTBL DW      KBD_TT                  ; START ADDRESS OF KBD TABLE
2588 0000              MMESS   DW      0                       ; ERROR MESSAGES , NOT USED BY CPM/86

                       SPAREA:
258A 02                NBRFLEX DB      2                       ; NUMBER OF FLEX DISKS
258B 81                IOBYTE  DB      10000001B               ; IOBYTE
258C 05                RETRYC  DB      5                       ; RETRY COUNTER
258D 05                RSTC    DB      5                       ; RESTORE COUNTER
258E 00                MODEFL  DB      0                       ; MODEFLAG: 0 - NO AUTO LOAD
                                                               ;           1 - AUTO LOAD ON COLD BOOT
                                                               ;           2 - AUTO LOAD ON WARM BOOT
                                                               ;           3 - AUTO LOAD ON COLD AND
                                                               ;               WARM BOOT IF CCP BUFFER
                                                               ;               LENGTH > 0

258F 00                CONFIGFL        DB      00H             ; CONFIGURE FLAG, IF SET IGNORE FUNCT.
2590 79                M1RS232         DB      79H             ; 1 STOP BIT, EVEN PARITY, PARITY
                                                               ; ENABLED, 7 BIT CHARACTER, ASYNCHRON
2591 3E                M2RS232         DB      3EH             ; INTERNAL CLOCKS, 9600 BAUD
2592 02                CONFVER         DB      02H             ; VERSION NUMBER OF CONFIG
2593 00                PVRS232         DB      00H             ; PROTOCOL VECTOR

2594 02                NUMHDSK         DB      2               ; TOTAL NUMBER OF DISK DRIVES
2595 E8                CRT_ATTR        DB      0E8H            ; CRT ATTRIBUTE
2596 3030303030        SER_NUMBER      DB      '00000'         ; DISK SERIAL NUMBER
259B CE                CURSOR          DB      0CEH            ; CURSOR TYPE
259C                   CMD_BUF         RS      33              ; COMMAND BUFFER FOR AUTOLOAD


```

```
                                 ;***********************************************************
                                 ;*                                                         *
                                 ;*                                                         *
                                 ;*              FUNCTION KEY DEFINITION TABLE              *
                                 ;*              FIRST WORD IS THE STRING'S LENGTH          *
                                 ;*                                                         *
                                 ;*              FUNC_TABLE: FUNCTION VALUES FOR ALL        *
                                 ;*                          UNSHIFTED FUNCTION KEYS        *
                                 ;*                                                         *
                                 ;*                                                         *
                                 ;***********************************************************


  25BD                           FUNC_TABLE     EQU     $                               ; START OF FUNCT. AREA
                                 ;
                                 ;
                                         IF NOT LOADER_BIOS
 25BD 0900                       FUN1           DW      LEN1
 25BF 44495220413A                              DB      'DIR A:',CR
      0D
  0009                           LEN1           EQU     (OFFSET $ - OFFSET FUN1)

 25C6 0900                       FUN2           DW      LEN2
 25C8 464F524D4154                              DB      'FORMAT',CR
      0D
  0009                           LEN2           EQU     (OFFSET $ - OFFSET FUN2)

 25CF 0B00                       FUN3           DW      LEN3
 25D1 434F50594449                              DB      'COPYDISK',CR
      534B0D
  000B                           LEN3           EQU     (OFFSET $ - OFFSET FUN3)

 25DA 0900                       FUN4           DW      LEN4
 25DC 434F4E464947                              DB      'CONFIG',CR
      0D
  0009                           LEN4           EQU     (OFFSET $ - OFFSET FUN4)

 25E3 0900                       FUN5           DW      LEN5
 25E5 444953434954                              DB      'DISCIT',CR
      0D
  0009                           LEN5           EQU     (OFFSET $ - OFFSET FUN5)

 25EC 0B00                       FUN6           DW      LEN6
 25EE 45584348414E                              DB      'EXCHANGE',CR
      47450D
  000B                           LEN6           EQU     (OFFSET $ - OFFSET FUN6)

 25F7 0A00                       FUN7           DW      LEN7
```

```

25F9 535441542041                 DB      'STAT A:',CR
          3A0D
  000A              LEN7          EQU     (OFFSET $ - OFFSET FUN7)

2601 0000           FUN8          DW      LEN8
2603 535441542042                 DB      'STAT B:*.*',CR
          3A2A2E2A0D
  0000              LEN8          EQU     (OFFSET $ - OFFSET FUN8)

260E 1100           FUN9          DW      LEN9
2610 50495020413A                 DB      'PIP A:=B:*.*[V]'
          3D423A2A2E2A
          5B565D
  0011              LEN9          EQU     (OFFSET $ - OFFSET FUN9)

261F 0900           FUN10         DW      LEN10
2621 44495220423A                 DB      'DIR B:',CR
          0D
  0009              LEN10         EQU     (OFFSET $ - OFFSET FUN10)

2628 1100           FUN11         DW      LEN11
262A 50495020423A                 DB      'PIP B:=A:*.*[V]'
          3D413A2A2E2A
          5B565D
  0011              LEN11         EQU     (OFFSET $ - OFFSET FUN11)

2639 0500           FUN12         DW      LEN12
263B 463132                       DB      'F12'
  0005              LEN12         EQU     (OFFSET $ - OFFSET FUN12)

263E 0500           FUN13         DW      LEN13
2640 463133                       DB      'F13'
  0005              LEN13         EQU     (OFFSET $ - OFFSET FUN13)

2643 0500           FUN14         DW      LEN14
2645 463134                       DB      'F14'
  0005              LEN14         EQU     (OFFSET $ - OFFSET FUN14)

2648 0500           FUN15         DW      LEN15
264A 463135                       DB      'F15'
  0005              LEN15         EQU     (OFFSET $ - OFFSET FUN15)

2640 0500           FUN16         DW      LEN16
264F 463136                       DB      'F16'
  0005              LEN16         EQU     (OFFSET $ - OFFSET FUN16)

2652 0500           FUN17         DW      LEN17
2654 463137                       DB      'F17'
  0005              LEN17         EQU     (OFFSET $ - OFFSET FUN17)

2657 0500           FUN18         DW      LEN18
2659 463138                       DB      'F18'
```

```

  0005                 LEN18          EQU    (OFFSET $ - OFFSET FUN18)

2650 0500              FUN19          DW     LEN19
265E 463139                           DB     'F19'
  0005                 LEN19          EQU    (OFFSET $ - OFFSET FUN19)

2661 0500              FUN20          DW     LEN20
2663 463230                           DB     'F20'
  0005                 LEN20          EQU    (OFFSET $ - OFFSET FUN20)

2666 00                FUN21          DB     0                              ; END INDICATOR

  0366                 FUNFILL        EQU    1040 - (OFFSET $ - OFFSET FUNC_TABLE); FILL TO 1040 BYTES
2667                                  RS     FUNFILL
                               ENDIF
                       ;
                       ;
29CD 00                FUN_END        DB     0                       ; END OF FUNCTION TABLE
                       ;*****************************************************
                       ;**                                                 **
                       ;**          CRT TRANSLATION TABLE                  **
                       ;**                                                 **
                       ;*****************************************************
                       ;*****************************************************

                       CRT_TABLE:


29CE 03                LVAR0   DB      VAR0L
29CF 8A2E              US      DB      8AH,2EH
  0003                 VAR0L   EQU     OFFSET $ - OFFSET LVAR0

                               IF NOT LOADER_BIOS

29D1 07                LVAR1   DB      VAR1L
29D2 5E0E23038A2E      UK      DB      5EH,0EH,23H,03H,8AH,2EH
  0007                 VAR1L   EQU     OFFSET $ - OFFSET LVAR1

29D8 15                LVAR2   DB      VAR2L
29D9 5B005C085D1C      FRANCE  DB      5BH,0DH,5CH,08H,5DH,1CH,40H,0AH,7BH,14H,7CH,1AH
     400A7B147C1A
29E5 7D087E0F2303              DB      7DH,0BH,7EH,0FH,23H,03H,27H,0CH
     270C
  0015                 VAR2L   EQU     OFFSET $ - OFFSET LVAR2

29ED 13                LVAR3   DB      VAR3L
29EE 5B005C065D09      GERMANY DB      5BH,00H,5CH,06H,5DH,09H,40H,1CH,7BH,10H,7CH,16H
     401C7B107C16
29FA 7D197E1E270C              DB      7DH,19H,7EH,1EH,27H,0CH
  0013                 VAR3L   EQU     OFFSET $ - OFFSET LVAR3

2A00 13                LVAR4   DB      VAR4L
2A01 5B005C065D02      SWEDEN  DB      5BH,00H,5CH,06H,5DH,02H,24H,13H,7BH,10H,7CH,16H
     24137B107C16
2A0D 7D127E0F270C              DB      7DH,12H,7EH,0FH,27H,0CH
  0013                 VAR4L   EQU     OFFSET $ - OFFSET LVAR4

2A13 13                LVAR5   DB      VAR5L
2A14 5B015C075D02      DANSK   DB      5BH,01H,5CH,07H,5DH,02H,23H,03H,7BH,11H,7CH,17H
     23037B117C17
2A20 7D127E0F270C              DB      7DH,12H,7EH,0FH,27H,0CH
  0013                 VAR5L   EQU     OFFSET $ - OFFSET LVAR5

2A26 0D                LVAR6   DB      VAR6L
2A27 5B1F5C055D1D      KSPAIN  DB      5BH,1FH,5CH,05H,5DH,1DH,27H,0CH,7CH,15H,23H,03H
     270C7C152303

  000D                 VAR6L   EQU     OFFSET $ - OFFSET LVAR6

 2A33 17               LVAR7   DB      VAR7L
 2A34 5B0D5C085D14     ITALY   DB      5BH,0DH,5CH,08H,5DH,14H,23H,03H,40H,1CH,7BH,0AH
      2303401C7B0A
 2A40 7C187D087E1B             DB      7CH,18H,7DH,08H,7EH,1BH,60H,1AH,27H,0CH
      601A270C
  0017                 VAR7L   EQU     OFFSET $ - OFFSET LVAR7

 2A4A 15               LVAR8   DB      VAR8L
 2A4B 2303270C4008     SWISS12 DB      23H,03H,27H,0CH,40H,08H,5BH,0AH,5CH,14H,5DH,0BH
      5B0A5C145D0B
 2A57 7B107C167D19             DB      7BH,10H,7CH,16H,7DH,19H,7EH,0FH
      7E0F
  0015                 VAR8L   EQU     OFFSET $ - OFFSET LVAR8

 2A5F 01               LVAR9   DB      VAR9L
                       CANADA1:
  0001                 VAR9L   EQU     OFFSET $ - OFFSET LVAR9

 2A60 0F               LVAR10  DB      VAR10L
 2A61 270C400A5C08     CANADA2 DB      27H,0CH,40H,0AH,5CH,08H,7BH,14H,7CH,9FH,7DH,08H
      7B147C9F7D08
 2A6D 7E0F                     DB      7EH,0FH
  000F                 VAR10L  EQU     OFFSET $ - OFFSET LVAR10

 2A6F 11               LVAR11  DB      VAR11L
 2A70 270C5B835C84     SAFRICA DB      27H,0CH,5BH,83H,5CH,84H,5DH,82H,7BH,93H,7CH,94H
      5D827B937C94
 2A7C 7D927E0F                 DB      7DH,92H,7EH,0FH
  0011                 VAR11L  EQU     OFFSET $ - OFFSET LVAR11

 2A80 11               LVAR12  DB      VAR12L
 2A81 2303270C5B80     PORTUG  DB      23H,03H,27H,0CH,5BH,80H,5CH,81H,5DH,85H,7BH,90H
      5C815D857B90
 2A8D 7C917D08                 DB      7CH,91H,7DH,08H
  0011                 VAR12L  EQU     OFFSET $ - OFFSET LVAR12

 2A91 15               LVAR13  DB      VAR13L
 2A92 408C5B8B5C88     YUGOSL  DB      40H,8CH,5BH,8BH,5CH,88H,5DH,89H,5EH,8AH,60H,9CH
      5D895E8A609C
 2A9E 7B9B7C987D99             DB      7BH,9BH,7CH,98H,7DH,99H,7EH,9AH
      7E9A
  0015                 VAR13L  EQU     OFFSET $ - OFFSET LVAR13

                               ENDIF

 2AA6                          RS      20

```

```
                            ;*****************************************************
                            ;*****************************************************
                            ;**                                                 **
                            ;**         KEYBOARD TRANSLATION TABLE              **
                            ;**                                                 **
                            ;*****************************************************
                            ;*****************************************************

     2ABA 80                KBD_TT  DB      80H                     ; 80 H
     2ABB 17                        DB      17H                     ; 81 H
     2ABC 13                        DB      13H                     ; 82 H  CURSOR LEFT
     2ABD 18                        DB      18H                     ; 83 H  CURSOR DOWN
     2ABE 05                        DB      05H                     ; 84 H  CURSOR UP
     2ABF 04                        DB      04H                     ; 85 H  CURSOR RIGHT
     2AC0 18                        DB      18H                     ; 86 H  CLEAR LINE (RUBOUT)
     2AC1 87                        DB      87H                     ; 87 H
     2AC2 0D                        DB      0DH                     ; 88 H  CARRIAGE RETURN
     2AC3 89                        DB      89H                     ; 89 H
     2AC4 2C                DEC_SIGN_1  DB  2CH                     ; 8A H  COMMA (MAY BE CHANGED BY KBD_INIT
                                                                                                    routine)
     2AC5 08                        DB      08H                     ; 8B H  BACKSPACE
     2AC6 8C                        DB      8CH                     ; 8C H
     2AC7 8D                        DB      8DH                     ; 8D H
     2AC8 8E                        DB      8EH                     ; 8E H
     2AC9 8F                        DB      8FH                     ; 8F H
     2ACA 90                        DB      90H                     ; 90 H
     2ACB 17                        DB      17H                     ; 91 H
     2ACC 13                        DB      13H                     ; 92 H  CURSOR LEFT
     2ACD 18                        DB      18H                     ; 93 H  CURSOR DOWN
     2ACE 05                        DB      05H                     ; 94 H  CURSOR UP
     2ACF 04                        DB      04H                     ; 95 H  CURSOR RIGHT
     2AD0 18                        DB      18H                     ; 96 H  CLEAR LINE (RUBOUT)
     2AD1 97                        DB      97H                     ; 97 H
     2AD2 0D                        DB      0DH                     ; 98 H  CARRIAGE RETURN
     2AD3 99                        DB      99H                     ; 99 H
     2AD4 2C                DEC_SIGN_2  DB  2CH                     ; 9A H  COMMA (MAY BE CHANGED BY KBD_INIT
                                                                                                    routine)
     2AD5 08                        DB      08H                     ; 9B H  BACKSPACE
     2AD6 9C                        DB      9CH                     ; 9C H
     2AD7 9D                        DB      9DH                     ; 9D H
     2AD8 9E                        DB      9EH                     ; 9E H
     2AD9 9F                        DB      9FH                     ; 9F H
```

```


                       INIT:
                               IF NOT LOADER_BIOS
2ADA E92321       4C00         JMP     MOVCPM  ; SET UP INTRPT. VECTORS,MOVE AND JUMP TO NEW O.S.
                               ENDIF
                       INIT40:         ; *** MOVCPM WILL JMPF HERE WITH A SEGMENT PARAGRAPH BASE OF 40
2ADD 8CC8                      MOV     AX,CS           ;ENTERED WITH A JMPF SO
2ADF 8ED0                      MOV     SS,AX           ; CS: AS THE INITIAL VALUE
2AE1 8ED8                      MOV     DS,AX           ; DS:,
2AE3 8EC0                      MOV     ES,AX           ; AND ES:
                               ;USE LOCAL STACK DURING INITIALIZATION
2AE5 BCFE43                    MOV SP,OFFSET STKBASE
                       ;
                       ;
                               IF NOT LOADER_BIOS
                       ;
2AE8 E88916       4174         CALL    KBD_INIT        ; GET COUNTRY CODE OF KBD
2AEB E86A00       2B58         CALL    CINIT                   ; GET FIRMWARE VERSION
2AEE 2EA09525                  MOV     AL,CRT_ATTR             ; SET CRT ATTRIBUTE
2AF2 A2DC44                    MOV     ATTRIBUTE,AL
2AF5 B90500                    MOV     CX,05H                  ; CX=COUNTER
2AF8 BE9625                    MOV     SI,OFFSET SER_NUMBER    ; MOVE SERIAL NUMBER OUT OF
2AFB BF9A43                    MOV     DI,OFFSET D_SER_NUM     ; CONFIG AREA INTO
2AFE F3A4                      REP     MOVS    AL,AL           ; SIGNON MESSAGE
2B00 BB1A43                    MOV     BX,OFFSET SIGNON        ; PRINT SIGN-ON MESSAGE
2B03 E84801       2C4E         CALL    PMSG
                               ;
2B06 E82800       2B31         CALL    PRINIT          ; INIT PRINTER
2B09 E83F0A       354B         CALL    DISKINIT        ; INIT DISK SYSTEM
2B0C 2E803E8E2501              CMP     MODEFL,1        ; LOOK FOR AUTOLOAD
2B12 740B         2B1F         JZ      G01
2B14 2E803E8E2503              CMP     MODEFL,3
2B1A 7403         2B1F         JZ      G01             ; JUMP IF AUTOLOAD
2B1C E9E4D4       0003         JMP     CCP+3
                       G01:
2B1F E81E00       2B40         CALL    AUTO_LOAD       ; MOVE COMMAND INTO CCP BUFFER
2B22 E9DBD4       0000         JMP     CCP

                               ENDIF
                       ;
                       ;
                               IF LOADER_BIOS

                               CALL    DISKINIT
                               PUSH    DS
                               MOV     AX,0
                               MOV     DS,AX
                               MOV     BDOS_OFFSET,BDOS_OFST
                               MOV     BDOS_SEGMENT,CS
                               POP     DS
                               JMP     CCP

```

```

                                  ENDIF
                          ;
                          ;
                          INT_TRAP:
2B25 FA                           CLI                       ;BLOCK INTERRUPTS
2B26 8CC8                         MOV     AX,CS
2B28 8ED8                         MOV     DS,AX             ;GET OUR DATA SEGMENT
2B2A BB0243                       MOV     BX,OFFSET INT_TRP
2B2D E81E01         2C4E          CALL    PMSG
2B30 F4                           HLT                       ;HARDSTOP
                          ;
                          ;
                          PRINIT:

                                  IF NOT LOADER_BIOS

2B31 2EA08B25                     MOV     AL,IOBYTE
2B35 E8F900         2C31          CALL    DSPACH6
2B38 AF42                         DW      SIOINIT
2B3A AF42                         DW      SIOINIT
2B3C CA42                         DW      PINIT
2B3E AF42                         DW      SIOINIT

                                  ENDIF

                                  IF LOADER_BIOS

                                  RET

                                  ENDIF


                          AUTO_LOAD:
2B40 51                           PUSH    CX
2B41 2E8A0E9C25                   MOV     CL,BYTE PTR CMD_BUF     ; READ COMMAND BUFFER LENGTH
2B46 FEC1                         INC     CL
2B48 B500                         MOV     CH,0
2B4A BE9C25                       MOV     SI,OFFSET CMD_BUF
2B4D BF0A00                       MOV     DI,OFFSET COMLEN
2B50 FC                           CLD
2B51 F3A4                         REP     MOVSB                   ; MOVE COMMAND BUFFER INTO CCP BUFFER
2B53 C60500                       MOV     BYTE PTR [DI],00H       ; HEX 0 INDICATES END OF COMMAND
2B56 59                           POP     CX
2B57 C3                           RET
```

```


                            CINIT:
2B58 06                             PUSH    ES
2B59 B80000                         MOV     AX,00H
2B5C 8EC0                           MOV     ES,AX                   ; SET ES TO 0
2B5E E611                           OUT     BYTE PTR ROMSELECT,AL   ; ENABLE FIRMWARE
2B60 BBF90F                         MOV     BX,FWVERSION+2          ; GET
2B63 268A07                         MOV     AL,ES:[BX]                ; COLOUR INDICATOR
2B66 A20544                         MOV     COLOUR_INDEX,AL         ; SAVE IT

2B69 BBF70F                         MOV     BX,FWVERSION
2B6C 268A07                         MOV     AL,ES:[BX]              ; IS LENGTH OF FIRMWARE
2B6F 3C08                           CMP     AL,08H                  ; VERSION ENTRY = 8?
2B71 7516           2B89            JNZ     OLD_FW                  ; IF NOT, WE GOT AN OLD FIRMWARE
2B73 B108                           MOV     CL,08H                  ; CL=COUNTER
2B75 BD8343                         MOV     BP,OFFSET FWMESS2       ; BP=DESTINATION OFFSET
2B78 BBF80F                         MOV     BX,FWVERSION+1          ; BX=SOURCE OFFSET
                            FW_MOVE:
2B7B 268A07                         MOV     AL,ES:[BX]
2B7E 884600                         MOV     [BP],AL
2B81 45                             INC     BP
2B82 43                             INC     BX
2B83 FEC9                           DEC     CL
2B85 75F4           2B7B            JNZ     FW_MOVE
2B87 EB05           2B8E            JMPS    RET1

                            OLD_FW:
2B89 C606A143FF                     MOV     FWMESS1,0FFH

                            RET1:
2B8E E610                           OUT     BYTE PTR RAMSELECT,AL   ; ENABLE RAM
2B90 07                             POP     ES
2B91 C3                             RET
```

```


                       ;
                       ;       WARM BOOT
                       ;
                       WBOOT:
2B92 C6063E4800                MOV     SACTIVE,00H        ; RESET PRINTER
2B97 C6063F4800                MOV     PACTIVE,00H        ; ACTIVE FLAGS
2B9C 803ED94400                CMP     GRAPHIC_FLAG,0     ; LOOK FOR GRAPHICS
2BA1 7408         2BAB         JZ      WBOOT1             ; IF NO GRAPHICS, JUMP
2BA3 C606D94400                MOV     GRAPHIC_FLAG,0     ; SET GRAPHIC MODE OFF
2BA8 E86804       3013         CALL    GRFXOFF
                       WBOOT1:
2BAB E8B709       3565         CALL    DISKWBOOT
2BAE 2E803E8E2502              CMP     MODEFL,2           ; LOOK FOR AUTOLOAD
2BB4 7C03         2BB9         JL      G02                ; JUMP IF NOT
2BB6 E887FF       2B40         CALL    AUTOLOAD           ; MOVE COMMAND INTO CCP BUFFER
                       G02:
2BB9 E94AD4       0006         JMP     CCP+6
                       ;
                       ;*** CONSOLE STATUS
                       ;
                       CONST:
2BBC 2EA08B25                  MOV     AL,IOBYTE
2BC0 E87200       2C35         CALL    DSPACHO
2BC3 5042                      DW      SPAIST
2BC5 0440                      DW      KEYST
2BC7 0440                      DW      KEYST
2BC9 5042                      DW      SPAIST
                       ;
                       ; ***
                       ;
                       CONOUT:
2BCB 2EA08B25                  MOV     AL,IOBYTE
2BCF E86300       2C35         CALL    DSPACHO     ; CALL DISPATCH (BASED ON LOWER 2 BITS OF IOBYT
2BD2 A542                      DW      SPAOUT
2BD4 8F2C                      DW      CRTMGR
2BD6 8F2C                      DW      CRTMGR
2BD8 A542                      DW      SPAOUT

                       ;***  CONIN


                       CONIN:
2BDA 2EA08B25                  MOV     AL,IOBYTE          ; LOAD IOBYTE
2BDE E85400       2C35         CALL    DSPACHO            ; COMPUTE ADDR. OF PROPER INPUT ROUTINE

2BE1 9042                      DW      SPAIN              ; TTY
2BE3 1540                      DW      KEYIN              ; CRT
2BE5 1540                      DW      KEYIN              ; CRT
2BE7 9042                      DW      SPAIN              ; TTY
```

```


                           ;*** READER

                           READER:

                                    IF NOT LOADER_BIOS

2BE9 2EA08B25                       MOV     AL,IOBYTE           ; LOAD IOBYTE
2BED B102                           MOV     CL,2
2BEF D2D8                           RCR     AL,CL               ; SHIFT RIGHT 2 BITS
2BF1 E84100         2C35            CALL    DSPACHO             ; COMPUTE ADDR. OF PROPER ROUTINE

2BF4 9042                           DW      SPAIN               ; TTY
2BF6 1540                           DW      KEYIN               ; CRT
2BF8 1540                           DW      KEYIN               ; CRT
2BFA 9042                           DW      SPAIN               ; TTY

                                    ENDIF


                                    IF LOADER_BIOS

                                    RET

                                    ENDIF

                           ;
                           ; ***
                           PUNCH:

                                    IF NOT LOADER_BIOS

2BFC 2EA08B25                       MOV     AL,IOBYTE
2C00 D0E8                           SHR     AL,1
2C02 D0E8                           SHR     AL,1        ; FOUR SHIFT RIGHTS SAVES TIME AND IS MORE
2C04 D0E8                           SHR     AL,1        ;     STRAIGHT FORWARD AS LOADING CL REGISTER
2C06 D0E8                           SHR     AL,1        ;     WITH A 4
2C08 E82A00         2C35            CALL    DSPACHO
2C0B A542                           DW      SPAOUT
2C0D 8F2C                           DW      CRTMGR
2C0F D942                           DW      P1CHROUT
2C11 8F2C                           DW      CRTMGR

                                    ENDIF


                                    IF LOADER_BIOS

                                    RET

                                    ENDIF

```

```

                          LISTOUT:

                                  IF NOT LOADER_BIOS

2C13 2EA08B25                     MOV     AL,IOBYTE
2C17 E81700       2C31            CALL    DSPACH6
2C1A 2C42                         DW      SRLOUT
2C1C 8F2C                         DW      CRTMGR
2C1E D942                         DW      P1CHROUT
2C20 2C42                         DW      SRLOUT

                                  ENDIF

                                  IF LOADER_BIOS

                                  RET

                                  ENDIF
                          ;
                          ;
                          LISTST:

                                  IF NOT LOADER_BIOS

2C22 2EA08B25                     MOV     AL,IOBYTE
2C26 E80800       2C31            CALL    DSPACH6
2C29 4A42                         DW      SRLSTAT
2C2B D440                         DW      KEYST
2C2D E342                         DW      P1STATUS
2C2F 4A42                         DW      SRLSTAT

                                  ENDIF

                                  IF LOADER_BIOS

                                  RET

                                  ENDIF

                          ;
                          ;  DISPATCHER ROUTINE  - ROUTES FUNCTION TO PROPER ROUTINE BASED ON IOBYTE
                          ;
                          DSPACH6:
2C31 D0C0                         ROL     AL,1
2C33 D0C0                         ROL     AL,1            ; ADJUST I/O BYTE FOR PRINTER
                          DSPACH0:
2C35 2403                         AND     AL,3
2C37 D0E0                         SHL     AL,1            ; 2 BYTE TABLES
2C39 5E                           POP     SI              ; RETURN ADDRESS IS REALLY TABLE BASE
2C3A 98                           CBW
2C3B 03F0                         ADD     SI,AX
2C3D FF24                         JMP     WORD PTR [SI]   ; JMP TO APPROPRIATE ROUTINE
```

```
                                ;
                                ;       GET AND SET IOBYTE ROUTINES
                                ;
                                GETIOBF:
2C3F 2EA08B25                           MOV     AL,IOBYTE       ; RETURNS IOBYTE IN REG AL
2C43 C3                                 RET
                                SETIOBF:
2C44 2E880E8B25                         MOV     IOBYTE,CL       ; EXPECTS NEW IOBYTE TO BE IN REG CL
2C49 C3                                 RET
                                ;
                                ;       ***     RETURN MEMORY REGION TABLE ADDRESS
                                ;
                                GETSEGT:
2C4A BBF042                             MOV     BX,OFFSET MRT   ; RETURN ADDRESS OF MEMORY REGION TABLE IN BX
2C4D C3                                 RET
                                ;
                                ;  UTILITY SUBROUTINE TO PRINT MESSAGES
                                ;
                                PMSG:
2C4E 8A07                               MOV     AL,[BX]         ;GET NEXT CHARACTER FROM MESSAGE
2C50 3CFF                               CMP     AL,0FFH
2C52 740B            2C5F               JZ      RETURN          ;IF ZERO RETURN
2C54 8AC8                               MOV     CL,AL
2C56 53                                 PUSH    BX              ; *** CONOUT DESTROYS BX !!
2C57 E871FF          2BCB               CALL    CONOUT          ;PRINT IT
2C5A 5B                                 POP     BX
2C5B 43                                 INC     BX
2C5C E9EFFF          2C4E               JMP     PMSG            ;NEXT CHARACTER AND LOOP
                                RETURN:
2C5F C3                                 RET
```

```


                             ;**************************************************************************
                             ;
                             ;
                             ;  BIOS INTERRUPT ROUTINE
                             ;
                             ;    THIS ROUTINE HANDLES SPECIAL SOFTWARE INTERRUPTS
                             ;
                             ;
                             ;  ENTRY VIA INT 222
                             ;       CL = 0   SET/RESET GRAPHIC FLAG
                             ;                AL = 0      CHARACTER MODE
                             ;                AL = OFFH   GRAPHIC MODE
                             ;
                             ;          = 1   SET/RESET CONFIG FLAG
                             ;                AL = 0      NORMAL OPERATION OF FUNCTION KEYS
                             ;                AL = OFFH   "CONFIG MODE" - RETURN ONLY VALUE OF FUNC KEY
                             ;
                             ;          = 2   RESERVED FOR FUTURE USE
                             ;
                             ;
                             ;  EXIT VIA IRET
                             ;       ALL REGISTERS PRESERVED
                             ;
                             ;
                             ;**************************************************************************


                             BIOS_INT_ROUTINE:
2C60 80F902                          CMP     CL,2                    ; LOOK FOR VALID FUNCTION
2C63 7F29           2C8E             JG      BIOS_INT_RET1
2C65 53                              PUSH    BX
2C66 1E                              PUSH    DS                      ; SAVE BX, DS
2C67 8CCB                            MOV     BX,CS
2C69 8EDB                            MOV     DS,BX                   ; SET DS = CS
2C6B 8AD9                            MOV     BL,CL
2C6D B700                            MOV     BH,0
2C6F 03DB                            ADD     BX,BX                   ; CALCULATE FUNCTION TABLE ENTRY
2C71 2EFFA7762C                      JMP     CS:FUNC_TAB[BX]


2C76 7C2C822C892C    FUNC_TAB        DW      FUNC0,FUNC1,FUNC2       ; JUMP TABLE

                             FUNC0:
2C7C A2D944                          MOV     GRAPHIC_FLAG,AL
2C7F E90A00         2C8C             JMP     BIOS_INT_RET

                             FUNC1:
2C82 2EA28F25                        MOV     CONFIGFL,AL
2C86 E90300         2C8C             JMP     BIOS_INT_RET

                             FUNC2:

2C89 E90000         2C8C             JMP     BIOS_INT_RET

                             BIOS_INT_RET:
2C8C 1F                              POP     DS                      ; RESTORE DS
2C8D 5B                              POP     BX                      ; AND BX

                             BIOS_INT_RET1:
2C8E CF                              IRET
```

```

=
=
=                       ;
=                       ;       CRTMGR is entered from CHARACTER OUT MANAGER
=                       ;
=                       ;               ENTRY:  CL=Character to OUTPUT
=                       ;
=                       CRTMGR:
=2C8F F6060444FF                TEST    STATUS_FLAG,0FFH        ; IF ESCAPE IN PROCESS, JUMP
=2C94 7519         2CAF         JNZ     PROC_STATUS
=2C96 8AC1                      MOV     AL,CL
=2C98 247F                      AND     AL,7FH
=2C9A 3C20                      CMP     AL,' '          ; CHECK IF CHARACTER IS A CONTROL CHARACTER
=2C9C 723E         2CDC         JB      PROC_CTL        ;    IF SO JUMP
=2C9E F606D944FF                TEST    GRAPHIC_FLAG,0FFH
=2CA3 7509         2CAE         JNZ     CRT_MGR_END     ; IF GRAPHIC MODE, RETURN
=2CA5 E81102       2EB9         CALL    CHR_TRAN        ; IF NO SPECIAL CASES, TRANSLATE CHARACTER
=2CA8 BBFE43                    MOV     BX,OFFSET CRTPB
=2CAB E8F503       30A3         CALL    HIP_OUT         ; OUTPUT CHARACTER (HIGH PERFORMANCE ROUTINE)
=                       CRT_MGR_END:
=2CAE C3                        RET                     ; RETURN TO BIOS CALLER
=                       PROC_STATUS:
=2CAF F606044402                TEST    STATUS_FLAG,ESCFLG      ; JUMP IF ESCAPE SEQUENCE IN PROCESS
=2CB4 750B         2CC1         JNZ     PROC_ESC
=                       ;
=                       ;       Otherwise the Data Request Flag must be set, so just fall through!!
=                       ;
=                       PROC_DRQ:
=2CB6 80260444FE                AND     STATUS_FLAG,NOT_DRQFLG  ; CLEAR DATA-REQUEST FLAG
=2CBB 8B1E0744                  MOV     BX,DRQ_ADRS
=2CBF FFE3                      JMP     BX              ; JUMP TO PREDETERMINED ROUTINE
=                       ;
=                       PROC_ESC:
=2CC1 80260444FD                AND     STATUS_FLAG,NOT_ESCFLG  ; CLEAR ESCAPE-IN-PROGRESS FLAG
=2CC6 BB022D                    MOV     BX,OFFSET ESC_TRANS
=                       TRANSLATE:
=2CC9 3A0F                      CMP     CL,[BX]         ; IS THIS THE CODE WE ARE LOOKING FOR?
=2CCB 740A         2CD7         JZ      TRANS_MATCH     ;     JUMP IF YES
=2CCD 803FFF                    CMP     BYTE PTR [BX],0FFH
=2CD0 7405         2CD7         JZ      TRANS_MATCH     ; ALSO JUMP IF END OF TABLE (NOT FOUND)
=2C02 83C303                    ADD     BX,3            ; (FASTER THAN THREE INCS)
=2CD5 EBF2         2CC9         JMPS    TRANSLATE       ; KEEP LOOKING
=                       TRANS_MATCH:
=2CD7 43                        INC     BX
=2CD8 8B37                      MOV     SI,WORD PTR [BX]
=2CDA FFE6                      JMP     SI
=                       ;
=                       PROC_CTL:
=2CDC BBE12C                    MOV     BX,OFFSET CTL_TRANS     ; NOTE THAT FOR PERFORMANCE REASONS
=2CDF EBE8         2CC9         JMPS    TRANSLATE               ;  WE DON'T DO THIS IN MAIN LINE CODE
```

```

 =
 =                        ;***    CONTROL CHARACTER TRANSLATION TABLE
 =                        ;
 =                        CTL_TRANS:
 =2CE1 00                         DB      00H
 =2CE2 4C2D                       DW      OFFSET(MGR_CR)
 =2CE4 0A                         DB      0AH
 =2CE5 782D                       DW      OFFSET(MGR_LF)
 =2CE7 1B                         DB      1BH
 =2CE8 F92D                       DW      OFFSET(MGR_ESC_SEQ)
 =2CEA 08                         DB      08H
 =2CEB 352D                       DW      OFFSET(MGR_BKSP)
 =2CED 1A                         DB      1AH
 =2CEE 882D                       DW      OFFSET(MGR_CLR)
 =2CF0 07                         DB      07H
 =2CF1 7E2D                       DW      OFFSET(MGR_BELL)
 =2CF3 1E                         DB      1EH
 =2CF4 602D                       DW      OFFSET(MGR_HOME)
 =2CF6 0C                         DB      0CH
 =2CF7 682D                       DW      OFFSET(MGR_NDFS)
 =2CF9 17                         DB      17H
 =2CFA 822D                       DW      OFFSET(MGR_EEOL)
 =2CFC 0B                         DB      0BH
 =2CFD 532D                       DW      OFFSET(MGR_RLF)
 =2CFF FF                         DB      0FFH
 =2D00 4B2D                       DW      OFFSET(MGR_RET)

 =
 =                        ;***    ESCAPE CODE TRANSLATION TABLE
 =                        ;
 =                        ESC_TRANS:
 =2D02 3D                         DB      '='
 =2D03 B82D                       DW      OFFSET(MGR_POSCUR)
 =2D05 29                         DB      29H
 =2D06 8C2D                       DW      OFFSET(MGR_HALF_I)
 =2D08 28                         DB      28H
 =2D09 A32D                       DW      OFFSET(MGR_FULL_I)
 =2D0B 47                         DB      'G'
 =2D0C FF2D                       DW      OFFSET(MGR_INVERSE)
 =2D0E 4D                         DB      'M'
 =2D0F 8B2E                       DW      OFFSET(MGR_MUSIC)
 =2D11 3A                         DB      03AH
 =2D12 882D                       DW      OFFSET(MGR_CLR)
 =2D14 2A                         DB      2AH
 =2D15 882D                       DW      OFFSET(MGR_CLR)
 =2D17 51                         DB      'Q'
 =2D18 E92D                       DW      OFFSET(MGR_INSCHR)
 =2D1A 57                         DB      'W'
 =2D1B ED2D                       DW      OFFSET(MGR_DELCHR)
 =2D1D 45                         DB      'E'
 =2D1E F12D                       DW      OFFSET(MGR_INSLIN)
 =2D20 52                         DB      'R'
 =2D21 F52D                       DW      OFFSET(MGR_DELLIN)
 =2D23 59                         DB      'Y'
 =2D24 E52D                       DW      OFFSET(MGR_CLEOS)
 =2D26 79                         DB      'y'
 =2D27 E52D                       DW      OFFSET(MGR_CLEOS)
 =2D29 54                         DB      'T'
 =2D2A 822D                       DW      OFFSET(MGR_EEOL)
 =2D2C 74                         DB      't'
 =2D2D 822D                       DW      OFFSET(MGR_EEOL)
 =2D2F 46                         DB      'F'
 =2D30 AB2E                       DW      OFFSET(MGR_FUNCCH)
 =2D32 FF                         DB      0FFH
 =2D33 4B2D                       DW      OFFSET(MGR_RET)
```

```

=
=                         ;
=                         ;***  BACK-SPACE CONTROL CODE
=                         ;
=                         MGR_BKSP:
=2D35 FE0EFE43                    DEC     BYTE PTR CRTPB+CPB_COL  ; DECREMENT COLUMN
=2D39 7908          2D46          JNS     MGR_WRITEPOS            ; JUMP IF COLUMN NOT NEGATIVE
=2D3B FE0EFF43                    DEC     BYTE PTR CRTPB+CPB_ROW  ; DECREMENT ROW
=2D3F 781F          2D60          JS      MGR_HOME                ; IF ROW GOES NEG, SIMPLY HOME CURSOR
=                         MGR_BKSP2:
=2D41 C606FE434F                  MOV     BYTE PTR CRTPB+CPB_COL,SCWID-1  ; COL=80 ROW IS ALRDY DECRMENTD
=                         MGR_WRITEPOS:
=2D46 B008                        MOV     AL,08             ; ESCAPE CODE: POSITION CURSOR ONLY
=2D48 E8D301        2F1E          CALL    DO_PIM_ESC
=                         MGR_RET:
=2D4B C3                          RET                       ; RETURN TO CALLER OF BIOS
=                         ;
=                         ;*** CARRIAGE RETURN CONTROL CODE
=                         ;
=                         MGR_CR:
=2D4C C606FE4300                  MOV     BYTE PTR CRTPB+CPB_COL,0        ; SIMPLY ZERO OUT COLUMN AND
                                                                                    POSITION
=2D51 EBF3          2D46          JMPS    MGR_WRITEPOS            ;       CURSOR
=                         ;
=                         ;*** REVERSE LINE FEED CONTROL CODE
=                         ;
=                         MGR_RLF:
=2D53 FE0EFF43                    DEC     BYTE PTR CRTPB+CPB_ROW  ; DECREMENT ROW
=2D57 79ED          2D46          JNS     MGR_WRITEPOS            ; IF ROW NOT NEGATIVE, POSITION CURSOR
=2D59 C606FF4300                  MOV     BYTE PTR CRTPB+CPB_ROW,0        ; DON'T LET THE ROW GO NEGATIVE!
=2D5E EBE6          2D46          JMPS    MGR_WRITEPOS
=                         ;
=                         ;*** HOME CONTROL CODE
=                         ;
=                         MGR_HOME:
=2D60 C706FE430000                MOV     WORD PTR CRTPB+CPB_COL,0        ; ZERO OUT CURSOR POSITION
=2D66 EBDE          2D46          JMPS    MGR_WRITEPOS                    ; AND WRITE CURSOR POSITION
=                         ;
=                         ;*** NON-DESTRUCTIVE FOWARD SPACE CONTROL CODE
=                         ;
=                         MGR_NDFS:
=2D68 FE06FE43                    INC     BYTE PTR CRTPB+CPB_COL          ; INCREMENT COLUMN
=2D6C 803EFE4350                  CMP     BYTE PTR CRTPB+CPB_COL,SCWID    ; IF NOT PAST LAST COLUMN
=2D71 7203          2D46          JB      MGR_WRITEPOS                    ;       ON SCREEN, WRITE CURSOR
=2D73 C606FE4300                  MOV     BYTE PTR CRTPB+CPB_COL,0        ; ELSE SET COLUMN TO ZERO AND
=                         ;   *** CAUTION NDFS ROUTINE FALLS INTO LINE FEED ROUTINE       DO LINE FEED
=                         ;
=                         ;*** LINE FEED CONTROL CODE
=                         ;
=                         MGR_LF:
=2D78 B00B                        MOV     AL,0BH            ; ESCAPE CODE: LINE FEED
=2D7A E8A101        2F1E          CALL    DO_PIM_ESC
=2D7D C3                          RET                       ; RETURN TO CALLER OF BIOS
```

```

=                         ;
=                         ;*** CONTROL CODE TO RING THE BELL
=                         ;
=                         MGR_BELL:
=2D7E E89814       4219           CALL    KBD_OUT          ; "BELL" CHAR IN CL - CALL THE KBD DRIVER
=2D81 C3                          RET                      ;        TO RING THE BELL
=                         ;
=                         ;*** ERASE TO END_OF_LINE CONTROL CODE
=                         ;
=                         MGR_EEOL:
=2D82 B003                        MOV     AL,03            ; PIM ESCAPE CODE FOR ERASE TO END OF LINE
=                         MGR_CALL_ESC:
=2D84 E89701       2F1E           CALL    DO_PIM_ESC       ; SEND ESCAPE CODE TO DRIVER
=2D87 C3                          RET
=                         ;
=                         ;*** CLEAR SCREEN CONTROL CODE
=                         ;
=                         MGR_CLR:
=2D88 B001                        MOV     AL,01            ; PIM ESCAPE CODE FOR CLEAR SCREEN
=2D8A EBF8         2D84           JMPS    MGR_CALL_ESC     ; (SAVES 2 BYTES)
=                         ;
=                         ;*** SET HALF INTENSITY ATTRIBUTE
=                         ;
=                         MGR_HALF_I:
=2D8C 803E054443                  CMP     COLOUR_INDEX,'C'                     ; LOOK FOR COLOUR
=2D91 7407         2D9A           JZ      COL_HALF_I
=2D93 800E004404                  OR      BYTE PTR CRTPB+CPB_ATTR,HALF_INTENSITY
=2D98 EB05         2D9F           JMPS    MGR_SET_ATTR
=
=                         COL_HALF_I:
=2D9A 800E004405                  OR      BYTE PTR CRTPB+CPB_ATTR,COLOUR_HALF_I
=
=                         MGR_SET_ATTR:
=2D9F B080                        MOV     AL,ATTR_MASK     ; SET ATTRIBUTE CODE
=2DA1 EBE1         2D84           JMPS    MGR_CALL_ESC     ; (SAVES 2 BYTES)
=                         ;
=                         ;*** CLEAR HALF INTENSITY ATTRIBUTE
=                         ;
=                         MGR_FULL_I:
=2DA3 803E054443                  CMP     COLOUR_INDEX,'C'                     ; LOOK FOR COLOUR
=2DA8 7407         2DB1           JZ      COL_FULL_I
=2DAA 80260044FB                  AND     BYTE PTR CRTPB+CPB_ATTR,NOT_HALF_INTENSITY
=2DAF EBEE         2D9F           JMPS    MGR_SET_ATTR
=
=                         COL_FULL_I:
=2DB1 80260044FA                  AND     BYTE PTR CRTPB+CPB_ATTR,NOT_COLOUR_HALF_I
=2DB6 EBE7         2D9F           JMPS    MGR_SET_ATTR
=                         ;
=                         ;*** POSITION CURSOR
=                         ;
=                         MGR_POSCUR:
=2DB8 C70607440120                MOV     DRQ_ADRS,OFFSET GETY    ; MOV GET COLUMN ADDRESS TO DATA REQ AD
```

```

=2DBE E94400        2E05           JMP     SET_DRQFLG             ;  ...AND WAIT FOR COL CHAR TO BE SENT
=                          GETY:
=2DC1 80E920                       SUB     CL,' '
=2DC4 80F919                       CMP     CL,ROWS+1
=2DC7 7704          2DCD           JA      GETY1
=2DC9 880EFF43                     MOV     CRTPB+CPB_ROW,CL       ; MOVE ADJUSTED CHAR SENT TO ROW
=                          GETY1:
=2DCD C7060744D62D                 MOV     DRQ_ADRS,OFFSET GETX
=2DD3 E92F00        2E05           JMP     SET_DRQFLG
=                          GETX:
=2DD6 80E920                       SUB     CL,' '
=2DD9 80F950                       CMP     CL,SCWID
=2DDC 7704          2DE2           JA      GETX1
=2DDE 880EFE43                     MOV     CRTPB+CPB_COL,CL       ; MOVE ADJUSTED CHAR SENT TO COLUMN
=                          GETX1:
=2DE2 E961FF        2D46           JMP     MGR_WRITEPOS
=                          ;
=                          ;*** CLEAR TO END-OF-SCREEN
=                          ;
=                          MGR_CLEOS:
=2DE5 B002                         MOV     AL,02   ; PIM ESCAPE CODE FOR CLEAR TO END OF SCREEN
=2DE7 EB9B          2D84           JMPS    MGR_CALL_ESC    ; JUMP TO ESCAPE SEQUENCE CALL
=                          ;
=                          ;*** INSERT CHARACTER
=                          ;
=                          MGR_INSCHR:
=2DE9 B006                         MOV     AL,06
=2DEB EB97          2D84           JMPS    MGR_CALL_ESC
=                          ;
=                          ;*** DELETE CHARACTER
=                          ;
=                          MGR_DELCHR:
=2DED B007                         MOV     AL,07
=2DEF EB93          2D84           JMPS    MGR_CALL_ESC
=                          ;
=                          ;*** INSERT LINE
=                          ;
=                          MGR_INSLIN:
=2DF1 B004                         MOV     AL,04           ; SCROLL DOWN ESCAPE CODE
=2DF3 EB8F          2D84           JMPS    MGR_CALL_ESC
=                          ;
=                          ;*** DELETE LINE
=                          ;
=                          MGR_DELLIN:
=2DF5 B005                         MOV     AL,05
=2DF7 EB8B          2D84           JMPS    MGR_CALL_ESC    ; DO A SCROLL UP
=                          ;
=                          ;*** ESCAPE CONTROL CODE
=                          ;
=                          MGR_ESC_SEQ:
=2DF9 800E044402                   OR      STATUS_FLAG,ESCFLG     ; SET ESCAPE-SEQUENCE-IN-PROGRESS FLAG
=2DFE C3                           RET
```

```

 =                          ;
 =                          ;*** SET/RESET VIDEO REVERSE ATTRIBUTE AND BLINKING
 =                          ;
 =                          MGR_INVERSE:
 =2DFF C70607440B2E                 MOV     DRQ_ADRS,UFFSET MGR_INV1
 =                          SET_DRQFLG:
 =2E05 800E044401                   OR      STATUS_FLAG,DRQFLG
 =                          MGR_RET2:
 =2E0A C3                           RET
 =                          MGR_INV1:
 =2E0B 80F930                       CMP     CL,'0'                  ; TEST FOR SET/RESET INVERSE VIDEO
 =2E0E 754A           2E5A          JNZ     MGR_INV3
 =2E10 80260044FD                   AND     BYTE PTR CRTPB+CPB_ATTR,NOT_BLINKING    ; RESET BLINKING
 =2E15 803E054443                   CMP     COLOUR_INDEX,'C'                        ; TEST FOR COLOUR
 =2E1A 7537           2E53          JNZ     MGR_INV2
 =2E1C 803E064400                   CMP     REV_VID,00H             ; GET REVERSE VIDEO ON/OFF FLAG
 =2E21 7463           2E86          JZ      MGR_SET_ATTR1           ; RETURN IF REVERSE VIDEO STILL RESET
 =2E23 C606064400                   MOV     REV_VID,00H             ; SET REVERSE VIDEO OFF
 =
 =                          MGR_COL1:
 =2E28 A00044                       MOV     AL,BYTE PTR CRTPB+CPB_ATTR
 =2E2B D0C0                         ROL     AL,1
 =2E2D D0C0                         ROL     AL,1
 =2E2F D0C0                         ROL     AL,1
 =2E31 F6D0                         NOT     AL                    ; COMPLEMENT FOREGROUND COLOUR
 =2E33 24E0                         AND     AL,0E0H               ; MASK NEW BACKGROUND COLOUR
 =2E35 8AC8                         MOV     CL,AL                 ; SAVE IT FOR LATER
 =2E37 A00044                       MOV     AL,BYTE PTR CRTPB+CPB_ATTR
 =2E3A D0C8                         ROR     AL,1
 =2E3C D0C8                         ROR     AL,1
 =2E3E D0C8                         ROR     AL,1
 =2E40 F6D0                         NOT     AL                    ; COMPLEMENT BACKGROUND COLOUR
 =2E42 241C                         AND     AL,1CH                ; MASK NEW FOREGROUND COLOUR
 =2E44 0AC8                         OR      CL,AL                 ; COMBINE WITH BACKGROUND COLOUR
 =2E46 A00044                       MOV     AL,BYTE PTR CRTPB+CPB_ATTR
 =2E49 2403                         AND     AL,03H                ; MASK BLINKING AND HALF INTENSITY
 =2E4B 0AC8                         OR      CL,AL
 =2E4D 880E0044                     MOV     BYTE PTR CRTPB+CPB_ATTR,CL
 =2E51 EB33           2E86          JMPS    MGR_SET_ATTR1
 =
 =                          MGR_INV2:
 =2E53 80260044FE                   AND     BYTE PTR CRTPB+CPB_ATTR,NOT_INVERSE     ; RESET INVERSE VIDEO
 =2E58 EB2C           2E86          JMPS    MGR_SET_ATTR1
 =
 =                          MGR_INV3:
 =2E5A 80F932                       CMP     CL,'2'  ; BLINKING?
 =2E5D 7507           2E66          JNZ     MGR_INV4
 =2E5F 800E004402                   OR      BYTE PTR CRTPB+CPB_ATTR,BLINKING        ; SET BLINKING
 =2E64 EB20           2E86          JMPS    MGR_SET_ATTR1
 =
 =                          MGR_INV4:
 =2E66 80F934                       CMP     CL,'4'  ; INVERSE VIDEO?
```

```

=2E69 759F         2E0A            JNZ     MGR_RET2       ; IF NOT DO NOTHING
=2E6B 803E054443                   CMP     COLOUR_INDEX,'C'       ; IF COLOUR
=2E70 7407         2E79            JZ      MGR_COL2               ; JUMP
=2E72 800E004401                   OR      BYTE PTR CRTPB+CPB_ATTR,INVERSE        ; SET INVERSE VIDEO
=2E77 EB0D         2E86            JMPS    MGR_SET_ATTR1
=
=                       MGR_COL2:
=2E79 803E064400                   CMP     REV_VID,00H            ; REVERSE VIDEO FLAG OFF?
=2E7E 758A         2E0A            JNZ     MGR_RET2               ; RETURN IF NOT
=2E80 FE060644                     INC     REV_VID                ; SET REVERSE VIDEO ON
=2E84 EBA2         2E28            JMPS    MGR_COL1
=
=                       MGR_SET_ATTR1:
=2E86 B080                         MOV     AL,ATTR_MASK
=2E88 E9F9FE       2D84            JMP     MGR_CALL_ESC
=                       ;
=                       ;***  PLAY MUSIC
=                       ;
=                       MGR_MUSIC:
=2E8B C7060744942E                 MOV     DRQ_ADRS,OFFSET MGR_GET_FREQ
=2E91 E971FF       2E05            JMP     SET_DRQ_FLG
=                       MGR_GET_FREQ:
=2E94 880E0244                     MOV     BYTE PTR CRTPB+CPB_FREQ,CL      ; SET FREQUENCY
=2E98 C7060744A12E                 MOV     DRQ_ADRS,OFFSET MGR_GET_FLEN
=2E9E E964FF       2E05            JMP     SET_DRQ_FLG
=                       MGR_GET_FLEN:
=2EA1 880E0344                     MOV     BYTE PTR CRTPB+CPB_FLEN,CL      ; SET FREQUENCY LENGTH
=2EA5 B009                         MOV     AL,09          ; PIM ESCAPE CODE FOR MUSIC
=2EA7 E87400       2F1E            CALL    DO_PIM_ESC
=2EAA C3                           RET
=
=                       ;***** CHANGE FUNCTION KEY DEFINITION
=
=                       MGR_FUNCCH:
=2EAB C7060744AB40                 MOV     DRQ_ADRS,OFFSET GETFCHAR
=2EB1 C6062448FF                   MOV     FNERR,0FFH
=2EB6 E94CFF       2E05            JMP     SET_DRQFLG
=
=                       ;
=                       ;*** CHRTRAN - CHARACTER TRANSLATE ROUTINE
=                       ;
=                       CHR_TRAN:
=2EB9 803E384800                   CMP     HEBREW,00H             ; LOOK FOR HEBREW
=2EBE 7718         2ED8            JA      TRAN_HEBREW
=2EC0 51                           PUSH    CX                     ; SAVE CHARACTER
=2EC1 A03C48                       MOV     AL,LANGUAGE            ; GET LANGUAGE CODE
=2EC4 3C20                         CMP     AL,20H                 ;
=2EC6 721F         2EE7            JB      TRAN_1                 ; IF ( 20 JUMP
=2EC8 3C32                         CMP     AL,32H                 ; LOOK FOR HEBREW
=2ECA 7504         2ED0            JNZ     TRAN_4                 ; IF NOT JUMP
=2ECC B000                         MOV     AL,00H
=2ECE EB21         2EF1            JMPS    TRAN_2
```

```

=
=                        TRAN_4:
=2ED0 240F                       AND     AL,0FH                  ; CLEAR BITS 8..5
=2ED2 BBB144                     MOV     BX,OFFSET LANG_T2       ; GET OFFSET OF LANGUAGE TABLE
=2ED5 D7                         XLAT    DS:LANG_T2              ; TRANSLATE
=2ED6 EB19          2EF1         JMPS    TRAN_2
=
=                        TRAN_HEBREW:
=2ED8 80F960                     CMP     CL,60H
=2EDB 7240          2F1D         JB      TRAN_END                ; NO TRANSLATION REQUIRED
=2EDD 80F97B                     CMP     CL,7BH
=2EE0 773B          2F1D         JA      TRAN_END                ; NO TRANSLATION REQUIRED
=2EE2 80E11F                     AND     CL,1FH                  ; CLEAR BITS 8,7,6
=2EE5 EB36          2F1D         JMPS    TRAN_END
=
=                        TRAN_1:
=2EE7 3C10                       CMP     AL,10H                  ; IF LANGUAGE CODE ( 10
=2EE9 7206          2EF1         JB      TRAN_2                  ; NO TRANSLATION IS NECESSARY
=2EEB 240F                       AND     AL,0FH                  ; CLEAR BITS 8..5
=2EED BBA944                     MOV     BX,OFFSET LANG_T1       ; GET OFFSET OF LANGUAGE TABLE
=2EF0 D7                         XLAT    DS:LANG_T1              ; TRANSLATE
=
=                        TRAN_2:
=2EF1 8AC8                       MOV     CL,AL
=2EF3 B500                       MOV     CH,00H
=2EF5 FEC1                       INC     CL
=2EF7 BDCE29                     MOV     BP,OFFSET CRT_TABLE     ; GET ADDRESS OF CRT TRANSLATION TABLE
=2EFA BE0000                     MOV     SI,0000H
=
=                        GET_CRT:
=2EFD 8A02                       MOV     AL,[BP+SI]              ; GET LENGTH OF TABLE ENTRY
=2EFF 98                         CBW
=2F00 03F0                       ADD     SI,AX                   ; ADD LENGTH OF ENTRY TO OFFSET POINTER
=2F02 E2F9          2EFD         LOOP    GET_CRT
=2F04 48                         DEC     AX                      ; DECREMENT LENGTH
=2F05 2BF0                       SUB     SI,AX                   ; WE NOW POINT TO THE END OF THE
=2F07 8BDE                       MOV     BX,SI                   ; ENTRY, SO SUBTRACT THE LENGTH
=2F09 03DD                       ADD     BX,BP                   ; TO GET THE START ADDRESS
=2F0B 59                         POP     CX                      ; RESTORE CHARACTER
=2F0C 40                         INC     AX
=2F0D 40                         INC     AX
=2F0E 4B                         DEC     BX
=2F0F 4B                         DEC     BX
=
=                        TRAN_3:
=2F10 43                         INC     BX
=2F11 43                         INC     BX
=2F12 48                         DEC     AX
=2F13 48                         DEC     AX                      ; DID WE REACH END OF TABLE ENTRY?
=2F14 7407          2F1D         JZ      TRAN_END                ; IF SO, RETURN
=2F16 3A0F                       CMP     CL,[BX]                 ; IS IT THE CHARACTER TO TRANSLATE
=2F18 75F6          2F10         JNE     TRAN_3                  ; IF NOT LOOP
```

```
1292
1293  =2F1A 43                        INC     BX
1294  =2F1B 8A0F                      MOV     CL,[BX]                 ; MOVE TRANSLATED CHARACTER
1295  =
1296  =                       TRAN_END:
1297  =2F1D C3                        RET
1298  =
1299  =                       ;
1300  =                       ;
1301  =                       ;
1302  =                       ;
1303  =                       ;*** ROUTINE TO CALL PIM TO PERFORM ESCAPE CODE
1304  =                       ;
1305  =                       DO_PIM_ESC:
1306  =2F1E F6060944FF                TEST    GRAPHIC_FLAG,0FFH
1307  =2F23 750F          2F34        JNZ     DO_PIM_ESC_END  ; IF GRAPHICS JUST RETURN
1308  =2F25 BBFE43                    MOV     BX,OFFSET CRTPB ; CRT PARAMETER BLOCK ADDRESS TO BX
1309  =2F28 FF7702                    PUSH    WORD PTR CPB_ATTR[BX] ; SAVE ATTR AND ESCAPE OF CRTPB ON STACK
1310  =2F2B 884703                    MOV     CPB_ESC[BX],AL  ; MOVE IN ESCAPE CODE
1311  =2F2E E81001        3041        CALL    CRTPIM          ; AND CALL DRIVER TO DO THE ESCAPE COMMAND
1312  =2F31 8F4702                    POP     WORD PTR CPB_ATTR[BX]   ; RESTORE ATTRIBUTE AND ESCAPE
1313  =                       DO_PIM_ESC_END:
1314  =2F34 C3                        RET
```

```

=
=                           ;
=                           ;
=                           ;
=                           ;*** ERROR DISPLAY ROUTINE INCLUDING GRAPHIC MODE CHECK ***
=                           ;
=                           ;
=                           ;
=                           ERR_DISP:
=2F35 53                              PUSH    BX                      ; SAVE ERROR MESSAGE ADDRESS
=2F36 FF36FE43                        PUSH    WORD PTR CRTPB          ; SAVE CURRENT CURSOR POSITION
=2F3A 53                              PUSH    BX
=2F3B 803ED94400                      CMP     GRAPHIC_FLAG,0          ; CHECK FOR GRAPHIC
=2F40 7523            2F65            JNZ     GRAPHIC                 ; IF GRAPHIC, JUMP
=2F42 BBD747                          MOV     BX,OFFSET POSMSG
=2F45 E806FD          2C4E            CALL    PMSG                    ; POSITION TO COLUMN 0, ROW 25
=2F48 5B                              POP     BX                      ; RESTORE ERROR MESSAGE ADDRESS
=2F49 E802FD          2C4E            CALL    PMSG                    ; AND DISPLAY THE MESSAGE
=2F4C E88BFC          28DA            CALL    CONIN                   ; GET THE RESPONSE
=2F4F 245F                            AND     AL,5FH                  ; CONVERT LOWER CASE TO UPPER CASE
=2F51 50                              PUSH    AX                      ; AND SAVE IT
=2F52 BBDC47                          MOV     BX,OFFSET RESMSG
=2F55 E8F6FC          2C4E            CALL    PMSG                    ; ERASE THE ERROR MESSAGE
=2F58 58                              POP     AX                      ; RESTORE RESPONSE
=2F59 5B                              POP     BX
=2F5A 50                              PUSH    AX
=2F5B 891EFE43                        MOV     WORD PTR CRTPB,BX
=2F5F E8E4FD          2046            CALL    MGR_WRITEPOS            ; RESTORE CURSOR TO PREVIOUS POSITION
=2F62 58                              POP     AX
=2F63 5B                              POP     BX
=2F64 C3                              RET
=                           ;
=                           ;
=                           ;INITIALIZE GRAPHICSCREEN FOR ERRORLINE
=                           ;
=                           ;
=                           GRAPHIC:
=2F65 51                              PUSH    CX
=2F66 C606D94400                      MOV     GRAPHIC_FLAG,0
=2F6B BBD747                          MOV     BX,OFFSET POSMSG
=2F6E E8DDFC          2C4E            CALL    PMSG                    ;POSITION TO COLUMN 0, ROW 25
=2F71 C606B04458                      MOV     GDC_LP12,25-1 OR 40H    ;CUTT ONE LINE FROM GRAPHIC SCREEN
=2F76 C7068E44803E                    MOV     GDC_SP2,400*40
=2F7C BBB944                          MOV     BX,INITSCR
=2F7F B90800                          MOV     CX,8                    ;
=2F82 E86000          2FE5            CALL    GRMOUT                  ;INIT SCREEN
=2F85 BBC244                          MOV     BX,ERROR_CUR_START
=2F88 B90300                          MOV     CX,3
=2F8B E85700          2FE5            CALL    GRMOUT                  ;SET CURSOR TO START OF ERROR LINE
=2F8E BBC644                          MOV     BX,MASK_OUT
=2F91 B90200                          MOV     CX,2
=2F94 E84E00          2FE5            CALL    GRMOUT                  ;SET MASK REGISTER TO FFFF
```

```

=2F97 BBC944                         MOV     BX,FIGS_OUT
=2F9A B90200                         MOV     CX,2
=2F9D E84500       2FE5              CALL    GRMOUT                  ;SET LENGTH TO CLEAR
=2FA0 BBC044                         MOV     BX,WDAT_OUT
=2FA3 B90200                         MOV     CX,2
=2FA6 E83C00       2FE5              CALL    GRMOUT                  ;SET CLEAR PATTERN
=2FA9 BBC244                         MOV     BX,ERROR_CUR_START
=2FAC B90300                         MOV     CX,3
=2FAF E83300       2FE5              CALL    GRMOUT                  ; SET CURSOR TO START OF ERROR LINE
=2FB2 BBC644                         MOV     BX,MASK_OUT
=2FB5 B90200                         MOV     CX,2
=2FB8 E82A00       2FE5              CALL    GRMOUT                  ; SET MASK REGISTERS TO FFFF
=2FBB 59                             POP     CX
=2FBC 5B                             POP     BX                      ; RESTORE ERROR MESSAGE ADDRESS
=2FBD E88EFC       2C4E              CALL    PMSG                    ; AND DISPLAY THE MESSAGE
=2FC0 E817FC       2BDA              CALL    CONIN                   ; GET THE RESPONSE
=2FC3 245F                           AND     AL,5FH                  ; CONVERT LOWER CASE TO UPPER CASE
=2FC5 5B                             POP     BX
=2FC6 5B                             POP     BX
=2FC7 C6060944FF                     MOV     GRAPHIC_FLAG,0FFH       ; SET GRAPHIC MODE
=2FCC C3                             RET
=
=
=                         ERR_DISP1:
=2FCD 803E094400                     CMP     GRAPHIC_FLAG,0          ; LOOK FOR GRAPHIC MODE
=2FD2 7410         2FE4              JZ      ERR_DISP_END            ; IF NOT JUMP
=                         ;
=                         ;
=                         ; CLEAR ERROR LINE
=                         ;
=                         ;
=2FD4 51                             PUSH    CX
=2FD5 C606BD4459                     MOV     GDC_LP12,25 OR 40H
=2FDA BBB944                         MOV     BX,INITSCR
=2FDD B90400                         MOV     CX,4
=2FE0 E80200       2FE5              CALL    GRMOUT                  ;INIT PAGE 1 TO FULL GRAPHIC SCREEN
=2FE3 59                             POP     CX
=
=                         ERR_DISP_END:
=2FE4 C3                             RET
=                         ;
=                         ;
=                         ; SUBROUTINES
=                         ;
=                         ;
=                         GRMOUT:
=2FE5 E82400       300C              CALL    GRGDCC1         ; GDC STATUS CHECK
=2FE8 8A07                           MOV     AL,[BX]         ;
=2FEA E6A1                           OUT     GRCMD,AL        ; COMMAND OUTPUT
=2FEC 83F900                         CMP     CX,0            ; IF NO PARAMETER
=2FEF 740A         2FFB              JE      GRMOUTRET       ;
=                         GRMOUT010:
```

```

=2FF1 43                               INC     BX                ;
=2FF2 8A07                             MOV     AL,[BX]           ;
=2FF4 E6A0                             OUT     GRPARA,AL         ; PARAMETER OUTPUT
=2FF6 E81300        300C               Call    GRGDCC1           ; wait till empty
=2FF9 E2F6          2FF1               LOOP    GRMOUT010         ;
=                           GRMOUTRET:
=2FFB C3                               RET                       ; RETURN
=                           GRSTART:
=2FFC E80D00        300C               CALL    GRGDCC1           ; GDC FIFO EMPTY CHECK
=2FFF B00D                             MOV     AL,STARTCMD       ;
=3001 E6A1                             OUT     GRCMD,AL          ; DISPLAY ENABLE
=3003 C3                               RET
=                           ;**********************************************************************
=                           GRSTOP:
=3004 E80500        300C               CALL    GRGDCC1           ; GDC FIFO EMPTY CHECK
=3007 B00C                             MOV     AL,STOPCMD        ;
=3009 E6A1                             OUT     GRCMD,AL          ; DISPLAY DISABLE
=300B C3                               RET
=                           GRGDCC1:
=300C E4A0                             IN      AL,GRSTATUS       ; GDC STATUS READ
=300E A804                             TEST    AL,04H            ;   FIFO EMPTY (DB2)
=3010 74FA          300C               JZ      GRGDCC1           ; IF NOT EMPTY
=3012 C3                               RET                       ; RETURN IF GDC FIFO IS EMPTY
=                           GRFXOFF:
=3013 E8EEFF        3004               Call    GRSTOP            ;DISABLE DISPLAY
=3016 BBD044                           Mov     BX,offset ALPHA_PARTITION
=3019 B90800                           Mov     CX,8              ;number of arguments
=301C E8C6FF        2FE5               Call    GRMOUT
=301F E866F0        2D88               CALL    MGR_CLR           ;CLEAR SCREEN (CHARACER MODE)
=3022 E80A00        302F               CALL    DELAY
=                           GRFXOFF1:
=3025 E4A0                             IN      AL,GRSTATUS       ;GDC STATUS READ
=3027 A820                             TEST    AL,20H
=3029 74FA          3025               JZ      GRFXOFF1
=302B E8CEFF        2FFC               Call    GRSTART           ;ENABLE DISPLAY
=302E C3                               Ret
=                           ;
=                           DELAY:
=302F B90400                           MOV     CX,4
=                           DELAY1:
=3032 E4A0                             IN      AL,GRSTATUS
=3034 A820                             TEST    AL,20H
=3036 74FA          3032               JZ      DELAY1
=                           DELAY2:
=3038 E4A0                             IN      AL,GRSTATUS
=303A A820                             TEST    AL,20H
=303C 75FA          3038               JNZ     DELAY2
=303E E2F2          3032               LOOP    DELAY1
=3040 C3                               RET

```

```


=                              INCLUDE C:CRTPIMC.SEG
=
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=
=                       ;************************************************************************
=                       ;*                                                                      *
=                       ;*             CRT Peripheral Interface Module                          *
=                       ;*                                                                      *
=                       ;************************************************************************
=                       ;
=                       ;  This Module is a hardware dependent, Operating System independent driver
=                       ;  for CRT display output
=                       ;
=                       ;  Entry Parameters:
=                       ;       CL = Character to be OUTPUT
=                       ;       BX = Address of CRT Parameter Block
=                       ;
=                       ;  Exit:  All registers unchanged
=                       ;
=                       CRTPIM:
=3041 50                        PUSH    AX
=3042 53                        PUSH    BX
=3043 51                        PUSH    CX
=3044 52                        PUSH    DX      ; SAVE ALL OF THE REGISTERS WE WILL BE WORKING WITH
=3045 56                        PUSH    SI
```

```

=3046 880EDD44                 MOV    OUTCHAR,CL      ; SAVE OUT CHARACTER IN MEMORY FOR LATER REF
=304A 8B07                     MOV    AX,CPB_COL[BX]
=304C A3DA44                   MOV    WORD PTR CURCOL,AX      ; ALSO SAVE ROW/COLUMN IN MEMORY
=304F 8A4703                   MOV    AL,CPB_ESC[BX]
=3052 A8FF                     TEST   AL,0FFH         ;
=3054 7423          3079       JZ     DO_OUTCHAR              ; IF ESCAPE = 0 THEN JUST OUTPUT CHARACTER
=3056 A880                     TEST   AL,ATTR_MASK    ;
=3058 7407          3061       JZ     DO_ESC          ; JUMP IF NO SET ATTRIBUTE SPECIFIED
=305A 8A6702                   MOV    AH,CPB_ATTR[BX] ;
=305D 8826DC44                 MOV    ATTRIBUTE,AH    ; SET ATTRIBUTE BYTE
=                       DO_ESC:
=3061 240F                     AND    AL,ESC_MASK
=3063 740E          3073       JZ     TEST_VID_OUT    ; SKIP ESCAPE PROCESSING IF NO ESCAPE FUNCTION
=3065 D0E0                     SHL    AL,1            ; FOR TABLE REFERENCING
=3067 98                       CBW                    ; EXPAND AL INTO AH
=3068 BED330                   MOV    SI,OFFSET ESC_TABLE
=306B 03F0                     ADD    SI,AX           ; AX = ADDRESS OF ESCAPE ROUTINE ADDRESS
=306D 53                       PUSH   BX              ; SAVE CRT PARAMETER BLOCK ADDRESS
=306E 51                       PUSH   CX              ; SAVE CHARACTER TO OUTPUT
=306F FF14                     CALL   WORD PTR [SI]   ; PERFORM ESCAPE FUNCTION
=3071 59                       POP    CX              ; RESTORE CHARACTER AND CRTPB ADDRESS
=3072 5B                       POP    BX
=                       TEST_VID_OUT:
=3073 F6470340                 TEST   BYTE PTR CPB_ESC[BX],CL_MASK
=3077 741F          3098       JZ     CRT_EXIT
=                       DO_OUTCHAR:
=3079 803EDA4450               CMP    CURCOL,SCWID    ; COLUMN > 80?
=307E 7503          3083       JNZ    01              ; JUMP IF NO
=3080 E83604        3489       CALL   SCLUP4          ; ELSE SCROLL UP SCREEN
=                       01:
=3083 8B16DC44                 MOV    DX,WORD PTR ATTRIBUTE   ; DH=OUTCAR DL=ATTRIBUTE
=3087 E86A00        30F4       CALL   WRGCHR
=308A FE06DA44                 INC    CURCOL
=308E 803EDA4450               CMP    CURCOL,SCWID
=3093 7203          3098       JB     CRT_EXIT
=3095 E8AF02        3347       CALL   BMPCR1          ; IF CURCOL>80, BUMP CUR
=                       CRT_EXIT:
=3098 5E                       POP    SI
=3099 5A                       POP    DX
=309A 59                       POP    CX
=309B 5B                       POP    BX
=309C A1DA44                   MOV    AX,WORD PTR CURCOL
=309F 8907                     MOV    CPB_COL[BX],AX  ; Restore CRTPB COL/ROW to latest state
=30A1 58                       POP    AX
=30A2 C3                       RET
=                       ;
=                       ;***  High Performance Screen Write     HIP_OUT
=                       ;
=                       ;      Entry Conditions - BX = CRTPB Address
=                       ;                         CL = Character to OUTPUT
=                       ;      Exit Conditions  - BX - Preserved
=                       ;                         AX, CX, DX - Destroyed

=                       ;                         CPB_COL and CPB_ROW fields of CRTPB updated
=                       ;
=                       HIP_OUT:
=30A3 8B07                      MOV     AX,CPB_COL[BX]
=30A5 A3DA44                    MOV     WORD PTR CURCOL,AX     ; Set-up CURCOL, CURROW, OUTCHAR fields
=30A8 880EDD44                  MOV     OUTCHAR,CL
=30AC 53                        PUSH    BX
=30AD 803EDA4450                CMP     CURCOL,SCWID           ; COLUMN > 80?
=30B2 7503         30B7         JNZ     H1             ; JUMP IF NO
=30B4 E80204       34B9         CALL    SCLUP4         ; ELSE SCROLL UP SCREEN
=                       H1:
=30B7 8B160C44                  MOV     DX,WORD PTR ATTRIBUTE  ; DH=OUTCAR DL=ATTRIBUTE
=30BB E83600       30F4         CALL    WRGCHR
=30BE FE06DA44                  INC     CURCOL
=30C2 803EDA4450                CMP     CURCOL,SCWID
=30C7 7203         30CC         JB      H2
=30C9 E87B02       3347         CALL    BMPCR1         ; IF CURCOL>80, BUMP CUR
=                       H2:
=30CC 5B                        POP     BX
=30CD A1DA44                    MOV     AX,WORD PTR CURCOL     ; Update CRTPB with CURCOL and CURROW
=30D0 8907                      MOV     CPB_COL[BX],AX
=30D2 C3                        RET
=                       ;
=                       ;***  Escape Table - Routines will be called indirect using the escape code * 2
=                       ;                    as an offset to the routine address
=                       ;
=                       ESC_TABLE:
=30D3 F330                      DW      OFFSET(NO_OP)
=30D5 5934                      DW      OFFSET(VCLEAR)
=30D7 1634                      DW      OFFSET(CLEOS)
=30D9 F533                      DW      OFFSET(ICLEOL)
=30DB E734                      DW      OFFSET(SCROLLDN)
=30DD 8B34                      DW      OFFSET(SCROLLUP)
=30DF 8433                      DW      OFFSET(INSCHR)
=30E1 C033                      DW      OFFSET(DELCHR)
=30E3 5C33                      DW      OFFSET(WRITEPOS)
=30E5 7233                      DW      OFFSET(MUSIC)
=30E7 F330                      DW      OFFSET(NO_OP)
=30E9 4634                      DW      OFFSET(ILF)
=30EB F330                      DW      OFFSET(NO_OP)
=30ED F330                      DW      OFFSET(NO_OP)
=30EF F330                      DW      OFFSET(NO_OP)
=30F1 F330                      DW      OFFSET(NO_OP)
=                       ;
=                       ;*** NO_OP   SIMPLY RETURNS IF ESCAPE CODE NOT IMPLEMENTED
=                       ;
=                       NO_OP:
=30F3 C3                        RET
=                       ;
=                       ;
=                       ;    WRGCHR, RDGCHR   WRITE AND READ GRAPHICS CHARACTER ROUTINES
=                       ;
```

```
1631
1632  =                       ;          WRITE OR READ ONE CHARACTER TO/FROM GDC IN MIXED MODE
1633  =                       ;
1634  =                       ;
1635  =                       ;***   WRGCHR  - Write Graphics Character
1636  =                       ;                ENTRY - DL = ATTRIBUTE
1637  =                       ;                        DH = CHARACTER
1638  =                       ;
1639  =                       WRGCHR:
1640  =30F4 E4A0              XX1:    IN      AL,GDCSTA
1641  =30F6 2402                      AND     AL,FIFULL
1642  =30F8 75FA        30F4          JNZ     XX1             ;LOOP UNTIL FIFO NOT FULL
1643  =30FA B020                      MOV     AL,WDAT OR TYWORD OR MOREPL
1644  =30FC E6A1                      OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
1645  =30FE E4A0              XX16:   IN      AL,GDCSTA
1646  =3100 2402                      AND     AL,FIFULL
1647  =3102 75FA        30FE          JNZ     XX16            ;LOOP UNTIL FIFO NOT FULL
1648  =3104 8AC6                      MOV     AL,DH
1649  =3106 E6A0                      OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1650  =3108 E4A0              XX17:   IN      AL,GDCSTA
1651  =310A 2402                      AND     AL,FIFULL
1652  =310C 75FA        3108          JNZ     XX17            ;LOOP UNTIL FIFO NOT FULL
1653  =310E 8AC2                      MOV     AL,DL
1654  =3110 E6A0                      OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1655  =3112 C3                        RET
1656  =                       ;
1657  =                       ;***   RDGCHR  - Read Graphics Character
1658  =                       ;               ENTRY - NONE
1659  =                       ;               EXIT  - DL = ATTRIBUTE
1660  =                       ;                       DH = CHARACTER
1661  =                       ;                       AL destroyed
1662  =                       ;
1663  =                       RDGCHR:
1664  =3113 E4A0              XX2:    IN      AL,GDCSTA
1665  =3115 2402                      AND     AL,FIFULL
1666  =3117 75FA        3113          JNZ     XX2             ;LOOP UNTIL FIFO NOT FULL
1667  =3119 B04C                      MOV     AL,FIGS         ;FIGURE DRAWING PARAMETER
1668  =311B E6A1                      OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
1669  =311D E4A0              XX18:   IN      AL,GDCSTA
1670  =311F 2402                      AND     AL,FIFULL
1671  =3121 75FA        311D          JNZ     XX18            ;LOOP UNTIL FIFO NOT FULL
1672  =3123 B002                      MOV     AL,2            ;DIRECTION = 2
1673  =3125 E6A0                      OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1674  =3127 E4A0              XX19:   IN      AL,GDCSTA
1675  =3129 2402                      AND     AL,FIFULL
1676  =312B 75FA        3127          JNZ     XX19            ;LOOP UNTIL FIFO NOT FULL
1677  =312D B001                      MOV     AL,1            ;DC = 1
1678  =312F E6A0                      OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
1679  =3131 E4A0              XX3:    IN      AL,GDCSTA
1680  =3133 2402                      AND     AL,FIFULL
1681  =3135 75FA        3131          JNZ     XX3             ;LOOP UNTIL FIFO NOT FULL
1682  =3137 B0A0                      MOV     AL,RDAT OR TYWORD       ;READ WORD FROM DISPLAY MEMORY
1683  =3139 E6A1                      OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
```

```

=313B E86A00        31A8          CALL   INPAR          ; GET ASCII CHARACTER
=313E 8AF0                        MOV    DH,AL
=3140 E86500        31A8          CALL   INPAR          ; GET ATTRIBUTE
=3143 8AD0                        MOV    DL,AL
=3145 C3                          RET
=                         ;
=                         ;*** SPCLEAR1   ENTRY: BX = Cursor Posistion
=                         ;                      CX = No. of bytes to clear
=                         ;
=                         SPCLEAR1:
=3146 03D9                        ADD    BX,CX
=3148 81FBD007                    CMP    BX,07D0H
=314C 760E          315C          JBE    SPCLEAR2       ; JUMP IF ENTIRE REGION TO CLEAR WITHIN 1ST PG
=314E 81EBD007                    SUB    BX,07D0H
=3152 E80700        315C          CALL   SPCLEAR2
=3155 8BC8                        MOV    CX,BX
=3157 33DB                        XOR    BX,BX   ;ZERO OUT BX
=3159 E87100        31CD          CALL   SETCUR1
=                         SPCLEAR2:
=315C 49                          DEC    CX
=3150 E89900        31F9          CALL   SETMSK
=3160 E4A0                XX4:    IN     AL,GDCSTA
=3162 2402                        AND    AL,FIFULL
=3164 75FA          3160          JNZ    XX4            ;LOOP UNTIL FIFO NOT FULL
=3166 B04C                        MOV    AL,FIGS
=3168 E6A1                        OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=316A E4A0                XX20:   IN     AL,GDCSTA
=316C 2402                        AND    AL,FIFULL
=316E 75FA          316A          JNZ    XX20           ;LOOP UNTIL FIFO NOT FULL
=3170 B002                        MOV    AL,2
=3172 E6A0                        OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3174 E4A0                XX21:   IN     AL,GDCSTA
=3176 2402                        AND    AL,FIFULL
=3178 75FA          3174          JNZ    XX21           ;LOOP UNTIL FIFO NOT FULL
=317A 8AC1                        MOV    AL,CL
=317C E6A0                        OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=317E E4A0                XX22:   IN     AL,GDCSTA
=3180 2402                        AND    AL,FIFULL
=3182 75FA          317E          JNZ    XX22           ;LOOP UNTIL FIFO NOT FULL
=3184 8AC5                        MOV    AL,CH
=3186 E6A0                        OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3188 E4A0                XX5:    IN     AL,GDCSTA
=318A 2402                        AND    AL,FIFULL
=318C 75FA          3188          JNZ    XX5            ;LOOP UNTIL FIFO NOT FULL
=318E B020                        MOV    AL,WDAT OR TYWORD OR MOREPL
=3190 E6A1                        OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=3192 E4A0                XX23:   IN     AL,GDCSTA
=3194 2402                        AND    AL,FIFULL
=3196 75FA          3192          JNZ    XX23           ;LOOP UNTIL FIFO NOT FULL
=3198 8020                        MOV    AL,020H
=319A E6A0                        OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=319C E4A0                XX24:   IN     AL,GDCSTA
```

```
=319E 2402                         AND   AL,FIFULL
=31A0 75FA        319C             JNZ   XX24           ;LOOP UNTIL FIFO NOT FULL
=31A2 A0DC44                       MOV   AL,ATTRIBUTE   ;*** WHAT ABOUT COLOR? ***
=31A5 E6A0                         OUT   GDCPAR,AL      ;SEND PARAMETER TO GDC
=31A7 C3                           RET
=                        ;
=                        INPAR:
=31A8 E4A0                         IN    AL,GDCSTA               ; READ GDC STATUS
=31AA 2401                         AND   AL,DATRDY
=31AC 74FA        31A8             JZ    INPAR          ; AND WAIT IF NO CHARACTER READY
=31AE E4A1                         IN    AL,FIFO
=31B0 C3                           RET
=                        ;
=                        ;*** SENPAR  SEND PARAMETERS TO SCREEN
=                        ;     ENTRY: BX = ADDRESS OF PARAMETER
=                        ;            CX = LENGTH
=                        ;     EXIT:  AL,BX,CX ARE DESTROYED
=                        ;            AH,DX  ARE PRESERVED
=                        ;
=                        SENPAR:
=31B1 E4A0               XX25:     IN    AL,GDCSTA
=31B3 2402                         AND   AL,FIFULL
=31B5 75FA        31B1             JNZ   XX25           ;LOOP UNTIL FIFO NOT FULL
=31B7 8A07                         MOV   AL,0[BX]
=31B9 E6A0                         OUT   GDCPAR,AL      ;SEND PARAMETER TO GDC
=31BB 43                           INC   BX             ; BUMP TO NEXT PARAMETER
=31BC E2F3        31B1             LOOP  SENPAR         ; LOOP UNTIL CX PARAMETERS HAVE BEEN SENT
=31BE C3                           RET
=                        ;
=                        ;*** SETCUR  - SET CURSOR
=                        ;                 ENTRY:   BX=GDC CURSOR POSITION
=                        ;                 EXIT:    AL,BX destroyed
=                        ;                          CX,DX preserved
=                        ;
=                        SETCUR:
=31BF 031EDE44                     ADD   BX,SP1
=31C3 81FB0007                     CMP   BX,0700H
=31C7 7204        31CD             JB    SETCUR1
=31C9 81EB0007                     SUB   BX,0700H
=                        SETCUR1:
=31CD E4A0               XX6:      IN    AL,GDCSTA
=31CF 2402                         AND   AL,FIFULL
=31D1 75FA        31CD             JNZ   XX6            ;LOOP UNTIL FIFO NOT FULL
=31D3 B049                         MOV   AL,CURS
=31D5 E6A1                         OUT   GDCCOM,AL      ;SEND COMMAND TO GDC
=31D7 E4A0               XX26:     IN    AL,GDCSTA
=31D9 2402                         AND   AL,FIFULL
=31DB 75FA        31D7             JNZ   XX26           ;LOOP UNTIL FIFO NOT FULL
=31DD 8AC3                         MOV   AL,BL
=31DF E6A0                         OUT   GDCPAR,AL      ;SEND PARAMETER TO GDC
=31E1 E4A0               XX27:     IN    AL,GDCSTA
=31E3 2402                         AND   AL,FIFULL
```

```

=31E5 75FA      31E1           JNZ    XX27           ;LOOP UNTIL FIFO NOT FULL
=31E7 8AC7                     MOV    AL,BH
=31E9 E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=31EB E4A0             XX28:   IN     AL,GDCSTA
=31ED 2402                     AND    AL,FIFULL
=31EF 75FA      31EB           JNZ    XX28           ;LOOP UNTIL FIFO NOT FULL
=31F1 32C0                     XOR    AL,AL
=31F3 E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=31F5 E80100    31F9           CALL   SETMSK
=31F8 C3                       RET
=                      ;
=                      ;*** SETMASK ROUTINE   (AL destroyed, all other registers preserved)
=                      ;
=                      SETMSK:
=31F9 E4A0             XX7:    IN     AL,GDCSTA
=31FB 2402                     AND    AL,FIFULL
=31FD 75FA      31F9           JNZ    XX7            ;LOOP UNTIL FIFO NOT FULL
=31FF B04A                     MOV    AL,MASKREG
=3201 E6A1                     OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=3203 E4A0             XX29:   IN     AL,GDCSTA
=3205 2402                     AND    AL,FIFULL
=3207 75FA      3203           JNZ    XX29           ;LOOP UNTIL FIFO NOT FULL
=3209 B0FF                     MOV    AL,-1
=320B E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=320D E4A0             XX30:   IN     AL,GDCSTA
=320F 2402                     AND    AL,FIFULL
=3211 75FA      320D           JNZ    XX30           ;LOOP UNTIL FIFO NOT FULL
=3213 B0FF                     MOV    AL,-1
=3215 E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3217 C3                       RET
=                      ;
=                      ;*** RDLIN      READ 1 ROW INTO LINBUF
=                      ;
=                      ;          Entry registers: none
=                      ;          Exit registers: AL, BX, CX destroyed
=                      ;                               DX preserved
=                      ;
=                      RDLIN:
=3218 E4A0             XX8:    IN     AL,GDCSTA
=321A 2402                     AND    AL,FIFULL
=321C 75FA      3218           JNZ    XX8            ;LOOP UNTIL FIFO NOT FULL
=321E B04C                     MOV    AL,FIGS
=3220 E6A1                     OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=3222 E4A0             XX31:   IN     AL,GDCSTA
=3224 2402                     AND    AL,FIFULL
=3226 75FA      3222           JNZ    XX31           ;LOOP UNTIL FIFO NOT FULL
=3228 B002                     MOV    AL,2           ;DIRECTION = 2
=322A E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=322C E4A0             XX32:   IN     AL,GDCSTA
=322E 2402                     AND    AL,FIFULL
=3230 75FA      322C           JNZ    XX32           ;LOOP UNTIL FIFO NOT FULL
=3232 B050                     MOV    AL,80          ;LENGTH = 80 WORDS [CHAR + ATTR]
```

```

=3234 E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3236 E4A0            XX33:    IN     AL,GDCSTA
=3238 2402                     AND    AL,FIFULL
=323A 75FA       3236          JNZ    XX33           ;LOOP UNTIL FIFO NOT FULL
=323C 32C0                     XOR    AL,AL
=323E E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3240 E4A0            XX9:     IN     AL,GDCSTA
=3242 2402                     AND    AL,FIFULL
=3244 75FA       3240          JNZ    XX9            ;LOOP UNTIL FIFO NOT FULL
=3246 B0A0                     MOV    AL,RDAT
=3248 E6A1                     OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=324A BB0944                   MOV    BX,OFFSET LINBUF
=324D B9A000                   MOV    CX,160         ; FOR READ LOOP
=                     RDLIN1:
=3250 E855FF     31A8          CALL   INPAR
=3253 8807                     MOV    0[BX],AL
=3255 43                       INC    BX
=3256 E2F8       3250          LOOP   RDLIN1
=3258 C3                       RET
=                     ;
=                     ;*** WRLIN  WRITE 1 ROW INTO GDC
=                     ;
=                     ;        Entry registers: none
=                     ;        Exit:            AL, BX, CX destoyed
=                     ;                         DX preserved
=                     ;
=                     WRLIN:
=3259 E4A0            XX10:    IN     AL,GDCSTA
=325B 2402                     AND    AL,FIFULL
=325D 75FA       3259          JNZ    XX10           ;LOOP UNTIL FIFO NOT FULL
=325F B04C                     MOV    AL,FIGS
=3261 E6A1                     OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
=3263 E4A0            XX34:    IN     AL,GDCSTA
=3265 2402                     AND    AL,FIFULL
=3267 75FA       3263          JNZ    XX34           ;LOOP UNTIL FIFO NOT FULL
=3269 B002                     MOV    AL,2
=326B E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=326D E4A0            XX35:    IN     AL,GDCSTA
=326F 2402                     AND    AL,FIFULL
=3271 75FA       326D          JNZ    XX35           ;LOOP UNTIL FIFO NOT FULL
=3273 32C0                     XOR    AL,AL
=3275 E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3277 E4A0            XX36:    IN     AL,GDCSTA
=3279 2402                     AND    AL,FIFULL
=327B 75FA       3277          JNZ    XX36           ;LOOP UNTIL FIFO NOT FULL
=327D 32C0                     XOR    AL,AL
=327F E6A0                     OUT    GDCPAR,AL      ;SEND PARAMETER TO GDC
=3281 E4A0            XX11:    IN     AL,GDCSTA
=3283 2402                     AND    AL,FIFULL
=3285 75FA       3281          JNZ    XX11           ;LOOP UNTIL FIFO NOT FULL
=3287 B020                     MOV    AL,WDAT OR TYWORD OR MOREPL
=3289 E6A1                     OUT    GDCCOM,AL      ;SEND COMMAND TO GDC
```

```
=328B BB0944                   MOV     BX,OFFSET LINBUF
=328E B9A000                   MOV     CX,160          ; FOR WRITE LOOP
=                     WRLIN1:
=3291 E4A0            XX37:    IN      AL,GDCSTA
=3293 2402                     AND     AL,FIFULL
=3295 75FA       3291          JNZ     XX37            ;LOOP UNTIL FIFO NOT FULL
=3297 8A07                     MOV     AL,0[BX]
=3299 E6A0                     OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=329B 43                       INC     BX
=329C E2F3       3291          LOOP    WRLIN1
=329E C3                       RET
=                     ;
=                     ;*** CUROFF      ROUTINE TO TURN CURSOR OFF    (destroys AL)
=                     ;
=                     CUROFF:
=329F E4A0            XX12:    IN      AL,GDCSTA
=32A1 2402                     AND     AL,FIFULL
=32A3 75FA       329F          JNZ     XX12            ;LOOP UNTIL FIFO NOT FULL
=32A5 B04B                     MOV     AL,CCHAR
=32A7 E6A1                     OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
=32A9 E4A0            XX38:    IN      AL,GDCSTA
=32AB 2402                     AND     AL,FIFULL
=32AD 75FA       32A9          JNZ     XX38            ;LOOP UNTIL FIFO NOT FULL
=32AF B00F                     MOV     AL,0FH
=32B1 E6A0                     OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=32B3 C3                       RET
=                     ;
=                     ;*** CURON       ROUTINE TO TURN CURSOR ON     (destoyes AL)
=                     ;
=                     CURON:
=32B4 E4A0            XX13:    IN      AL,GDCSTA
=32B6 2402                     AND     AL,FIFULL
=32B8 75FA       32B4          JNZ     XX13            ;LOOP UNTIL FIFO NOT FULL
=32BA B04B                     MOV     AL,CCHAR
=32BC E6A1                     OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
=32BE E4A0            XX39:    IN      AL,GDCSTA
=32C0 2402                     AND     AL,FIFULL
=32C2 75FA       32BE          JNZ     XX39            ;LOOP UNTIL FIFO NOT FULL
=32C4 B08F                     MOV     AL,08FH
=32C6 E6A0                     OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=32C8 E4A0            XX40:    IN      AL,GDCSTA
=32CA 2402                     AND     AL,FIFULL
=32CC 75FA       32C8          JNZ     XX40            ;LOOP UNTIL FIFO NOT FULL
=32CE 2EA09B25                 MOV     AL,BYTE PTR CURSOR
=32D2 E6A0                     OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=32D4 E4A0            XX41:    IN      AL,GDCSTA
=32D6 2402                     AND     AL,FIFULL
=32D8 75FA       32D4          JNZ     XX41            ;LOOP UNTIL FIFO NOT FULL
=32DA B072                     MOV     AL,072H
=32DC E6A0                     OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=32DE C3                       RET
=                     ;

=                         ;*** INIT10   INITIALIZE SCREEN PAGE VALUES
=                         ;
=                         INIT10:
=32DF 33C0                        XOR     AX,AX
=32E1 A3DE44                      MOV     SP1,AX
=32E4 A3E244                      MOV     SP2,AX          ; START OF PAGES 1 AND 2 = 0
=32E7 A2E544                      MOV     LP22,AL         ; LENGTH OF PAGE 2 = 0
=32EA C606E14419                  MOV     LP12,25         ; LENGTH OF PAGE 1 = 25
=32EF E4A0                XX14:   IN      AL,GDCSTA
=32F1 2402                        AND     AL,FIFULL
=32F3 75FA          32EF          JNZ     XX14            ;LOOP UNTIL FIFO NOT FULL
=32F5 B04C                        MOV     AL,FIGS
=32F7 E6A1                        OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
=32F9 E4A0                XX42:   IN      AL,GDCSTA
=32FB 2402                        AND     AL,FIFULL
=32FD 75FA          32F9          JNZ     XX42            ;LOOP UNTIL FIFO NOT FULL
=32FF B002                        MOV     AL,2
=3301 E6A0                        OUT     GDCPAR,AL       ;SEND PARAMETER TO GDC
=3303 C3                          RET
=                         ;
=                         ;***    SCROLL ROUTINE
=                         ;
=                         SCROLLX:
=3304 33DB                        XOR     BX,BX           ; START OF PAGE 1
=3306 B95000                      MOV     CX,80
=3309 E87301        347F          CALL    SPCLEAR
=330C 8B1EDE44                    MOV     BX,SP1
=3310 83C350                      ADD     BX,80
=3313 891EDE44                    MOV     SP1,BX
=3317 FE0EE144                    DEC     LP12
=331B 7506          3323          JNZ     SCROL2
=331D E8BFFF        32DF          CALL    INIT10
=3320 E90400        3327          JMP     SCROL1
=                         SCROL2:
=3323 FE06E544                    INC     LP22
=                         SCROL1:
=3327 E4A0                XX15:   IN      AL,GDCSTA
=3329 2402                        AND     AL,FIFULL
=332B 75FA          3327          JNZ     XX15            ;LOOP UNTIL FIFO NOT FULL
=332D B070                        MOV     AL,PRAM+0       ;SCROL1 SENDS THE 8 BYTE SCREEN PAGES INFO
=332F E6A1                        OUT     GDCCOM,AL       ;SEND COMMAND TO GDC
=3331 B90800                      MOV     CX,8
=3334 BBDE44                      MOV     BX,OFFSET SP1
=3337 E877FE        31B1          CALL    SENPAR
=333A C3                          RET
=                         ;
=                         ;***  BUMPCUR  -  BUMP CURSOR AND UPDATE CURCOL & CURROW
=                         ;                         CRTPB WILL BE UPDATED WITH THESE VALUES
=                         ;                         BEFOR EXITING THE CRTPIM
=                         ;
=                         BUMPCUR:
=333B FE06DA44                    INC     CURCOL
```

```
2002
2003  =333F 803EDA4450              CMP     CURCOL,SCWID
2004  =3344 7301          3347      JAE     BMPCR1          ; JUMP IF CURCOL+1 IS GREATER THAN 80
2005  =3346 C3                      RET
2006  =                        BMPCR1:
2007  =3347 803EDB4417              CMP     CURROW,ROWS-1
2008  =334C 7501          334F      JNZ     BMPCR2          ; IF WE ARE ON LAST ROW, DO NOTHING (WILL BE
2009  =334E C3                      RET                     ;      CHECKED LATER FOR SCROLLING)
2010  =                        BMPCR2:
2011  =334F C606DA4400              MOV     CURCOL,0
2012  =3354 FE06DB44                INC     CURROW
2013  =3358 E80100        335C      CALL    WRITEPOS
2014  =335B C3                      RET
2015  =                        ;
2016  =                        ;***   WRITEPOS    WRITE CURSOR POSITION ROUTINE
2017  =                        ;                  ENTRY: NONE
2018  =                        ;                  EXIT: AL, BX     -DESTROYED
2019  =                        ;                        AH, CX, DX -PRESERVED
2020  =                        ;
2021  =                        WRITEPOS:
2022  =335C 8B1EDA44                MOV     BX,WORD PTR CURCOL
2023  =3360 E80400        3367      CALL    WRHLPOS         ; COMPUTE ADDRESS IN CRT BUFFER
2024  =3363 E859FE        31BF      CALL    SETCUR
2025  =3366 C3                      RET
2026  =                        ;
2027  =                        ;***   WRHLPOS COMPUTE ADDRESS WITHIN CRT-BUFFER
2028  =                        ;              ENTER - BL = COLUMN
2029  =                        ;                      BH = ROW
2030  =                        ;              EXIT  - BX = ADDRESS IN CRT BUFFER
2031  =                        ;                      AX, CX, DX -PRESERVED
2032  =                        ;
2033  =                        WRHLPOS:
2034  =3367 50                      PUSH    AX
2035  =3368 B050                    MOV     AL,SCWID        ; CHARS/ROW IN AL
2036  =336A F6E7                    MUL     BH              ; MULTIPLY BY ROW NO. - RESULT IN AX
2037  =336C 32FF                    XOR     BH,BH           ; BH = 0
2038  =336E 03D8                    ADD     BX,AX           ; NOW BX IS CORRECT POSITION IN CRT BUFFER
2039  =3370 58                      POP     AX
2040  =3371 C3                      RET
2041  =                        ;
2042  =                        ;***  MUSIC      PLAY MUSIC
2043  =                        ;
2044  =                        MUSIC:
2045  =3372 B106                    MOV     CL,06
2046  =3374 E8A20E        4219      CALL    KBD_OUT         ; CALL KEYBOARD PIM WITH MUSIC FUNCTION CODE
2047  =3377 8A4F04                  MOV     CL,CPB_FREQ[BX]
2048  =337A E89C0E        4219      CALL    KBD_OUT         ; SEND FREQUENCE TO KEYBOARD
2049  =337D 8A4F05                  MOV     CL,CPB_FLEN[BX]
2050  =3380 E8960E        4219      CALL    KBD_OUT         ; SEND LENGTH OF FREQUENCE TO KEYBOARD
2051  =3383 C3                      RET
2052  =                        ;
2053  =                        ;***  INSCHR     INSERT CHARACTER ROUTINE
2054  =                        ;
```

```

=                             INSCHR:
=3384 E86000        33E7              CALL    TEST_POS
=3387 7427          33B0              JZ      BLANK_ONE
=3389 8A3EDB44                        MOV     BH,CURROW
=338D B34E                            MOV     BL,SCWID-2
=338F E8D5FF        3367              CALL    WRHLPOS         ; GET CHARACTER POINTER IN BX
=3392 E80AFF        329F              CALL    CUROFF          ; SWITCH CURSOR OFF
=                             INSCH1:
=3395 53                              PUSH    BX
=3396 E826FE        31BF              CALL    SETCUR          ; SET CURSOR
=3399 E877FD        3113              CALL    RDGCHR          ; GET CHARACTER
=339C 5B                              POP     BX
=339D 43                              INC     BX
=339E 53                              PUSH    BX
=339F E81DFE        31BF              CALL    SETCUR          ; SET CURSOR
=33A2 E84FFD        30F4              CALL    WRGCHR          ; SET CHARACTER
=33A5 5B                              POP     BX
=33A6 4B                              DEC     BX
=33A7 4B                              DEC     BX
=33A8 FEC9                            DEC     CL              ; DECREMENT COUNTER
=33AA 75E9          3395              JNZ     INSCH1          ; LOOP UNTIL ZERO
=33AC E805FF        32B4              CALL    CURON           ; SWITCH CURSOR ON
=33AF 43                              INC     BX
=                             BLANK_ONE:
=33B0 B620                            MOV     DH,' '          ; CHARACTER REQUIRED IN DH
=33B2 E80AFE        31BF              CALL    SETCUR          ; SET CURSOR
=33B5 8A16DC44                        MOV     DL,ATTRIBUTE    ; GET ATTRIBUTE
=33B9 E838FD        30F4              CALL    WRGCHR          ; CLEAR CHARACTER
=33BC E89DFF        335C              CALL    WRITEPOS        ; SET CURSOR
=33BF C3                              RET
=                             ;
=                             ;*** DELCHR     DELETE ONE CHARACTER
=                             ;
=                             DELCHR:
=33C0 E82400        33E7              CALL    TEST_POS        ; RETURNS: CL = NO. OF POSITIONS TO MOVE
=                                                             ;          BX = ROW*80+COL
=                                                             ;          ZF SET IF ZERO POSITIONS TO MOVE
=33C3 74EB          33B0              JZ      BLANK_ONE       ; EXIT IF NONE TO MOVE
=33C5 43                              INC     BX              ; START AT PRES + 1
=33C6 E8D6FE        329F              CALL    CUROFF          ; SWITCH OFF CURSOR
=                             DELCHR1:
=33C9 53                              PUSH    BX
=33CA E8F2FD        31BF              CALL    SETCUR          ; SET CURSOR
=33CD E843FD        3113              CALL    RDGCHR          ; GET CHARACTER
=33D0 5B                              POP     BX
=33D1 4B                              DEC     BX
=33D2 53                              PUSH    BX
=33D3 E8E9FD        31BF              CALL    SETCUR          ; SET CURSOR
=33D6 E81BFD        30F4              CALL    WRGCHR          ; SET CHARACTER
=33D9 5B                              POP     BX
=33DA 43                              INC     BX
=33DB 43                              INC     BX
```

```
=33DC FEC9                      DEC     CL             ; DECREMENT COUNTER OF CHARACTER TO MOVE
=33DE 75E9        33C9          JNZ     DELCHR1        ; LOOP UNTIL ZERO
=33E0 E8D1FE      32B4          CALL    CURON          ; SWITCH ON CURSOR
=33E3 4B                        DEC     BX
=33E4 E9C9FF      33B0          JMP     BLANK_ONE
=                      ;
=                      ;***  TEST_POS  RETURNS CURSOR POSITION AND LENGTH
=                      ;                ENTRY REGS: NONE
=                      ;                EXIT REGS:  BX = CUR POSITION (ROW*80+COL)
=                      ;                           CL = LENGTH TO MOVE
=                      ;                     ZF SET TO ZERO MEANS NO CHARACTERS TO MOVE!
=                      TEST_POS:
=33E7 8B1EDA44                  MOV     BX,WORD PTR CURCOL      ; BL = COLUMN ; BH = ROW
=33EB E879FF      3367          CALL    WRHLPOS        ; COMPUTE ADDRESS WITHIN CRT BUFFER
=33EE B14F                      MOV     CL,SCWID-1     ; TEST IF CURRENT COLUMN = SCWID-1
=33F0 2A0EDA44                  SUB     CL,CURCOL      ; CL = COUNT
=33F4 C3                        RET
=                      ;
=                      ;***     ICLEOL  ERASE TO END OF LINE
=                      ;
=                      ICLEOL:
=33F5 8A1EDA44                  MOV     BL,CURCOL      ; CURRENT COLUMN NUMBER TO CH AND BL
=33F9 8AEB                      MOV     CH,BL
=33FB B050                      MOV     AL,SCWID       ; SUBTRACT COLUMN NUMBER FROM SCREEN WIDTH TO
=33FD 2AC5                      SUB     AL,CH          ;      GET NUMBER OF BYTES TO CLEAR
=33FF 7414        3415          JZ      ICLEOL_RET
=3401 8AC8                      MOV     CL,AL          ; CX = NUMBER OF BYTES TO CLEAR
=3403 32ED                      XOR     CH,CH
=3405 51                        PUSH    CX
=3406 A0DB44                    MOV     AL,CURROW
=3409 8AF8                      MOV     BH,AL
=340B E859FF      3367          CALL    WRHLPOS        ; BX = ADDRESS OF CHARACTER IN CRT RAM
=340E 59                        POP     CX             ; CX = NUMBER OF BYTES TO CLEAR
=340F E86D00      347F          CALL    SPCLEAR        ; CLEAR
=3412 E847FF      335C          CALL    WRITEPOS
=3415 C3               ICLEOL_RET:      RET
=                      ;
=                      ;***     CLEOS   CLEAR FROM CURRENT ROW TO END OF SCREEN
=                      ;
=                      CLEOS:
=3416 B017                      MOV     AL,ROWS-1      ; CALCULATE NUMBER OF ROWS TO BE CLEARED
=3418 2A06DB44                  SUB     AL,CURROW
=341C 741A        3438          JZ      CLEOS1         ; IF ZERO, JUST CLEAR CURRENT ROW
=341E 8A3EDB44                  MOV     BH,CURROW
=3422 FEC7                      INC     BH             ; BH = CURRENT ROW + 1
=3424 32DB                      XOR     BL,BL          ; BL = 0  (COLLUMN 0)
=3426 E83EFF      3367          CALL    WRHLPOS
=3429 B250                      MOV     DL,SCWID
=342B F6E2                      MUL     DL             ; AX = NUMBER OF BYTES TO CLEAR
=342D 8BC8                      MOV     CX,AX
=342F E86DFE      329F          CALL    CUROFF         ; SWITCH OFF CURSOR
=3432 E84A00      347F          CALL    SPCLEAR        ; CLEAR TO SPACES
```

```

=3435 E87CFE      32B4         CALL    CURON            ; SWITCH ON CURSOR
=                       CLEOS1:
=3438 E88AFF      33F5         CALL    ICLEOL
=343B C3                       RET
=                       ;
=                       ;***  IHOME   PHYSICAL HOME CURSOR
=                       ;
=                       IHOME:
=343C C7060A440000             MOV     WORD PTR CURCOL,0       ; ZERO OUT CURCOL AND CURROW
=3442 E817FF      335C         CALL    WRITEPOS
=3445 C3                       RET
=                       ;
=                       ;***    ILF     INTERNAL LINE FEED
=                       ;
=                       ILF:
=3446 A00B44                   MOV     AL,CURROW
=3449 FEC0                     INC     AL
=344B 3C18                     CMP     AL,ROWS
=344D 7307        3456         JAE     ILF1
=344F A20B44                   MOV     CURROW,AL
=3452 E807FF      335C         CALL    WRITEPOS
=3455 C3                       RET
=                       ILF1:
=3456 E96600      34BF         JMP     SCLUP3
=                       ;
=                       ;***  VCLEAR  CLEAR SCREEN; HOME CURSOR
=                       ;
=                       VCLEAR:
=3459 E843FE      329F         CALL    CUROFF  ; CURSOR OFF
=345C E880FE      320F         CALL    INIT10
=345F E8C5FE      3327         CALL    SCROL1  ; INITIALIZE PAGES
=3462 BB0000                   MOV     BX,0
=3465 B90007                   MOV     CX,ROWS*SCWID+SCWID
=3468 E81400      347F         CALL    SPCLEAR ; DO IT TO THE SCREEN
=346B E8CEFF      343C         CALL    IHOME
=346E E843FE      32B4         CALL    CURON   ; TURN CURSOR BACK ON
=3471 C3                       RET
=                       ;
=                       ;*** CLRLIN  CLEAR ROW (AL) TO SPACES
=                       ;
=                       CLRLIN:
=3472 B350                     MOV     BL,SCWID
=3474 F6E3                     MUL     BL      ; CALCULATE ABSOLUTE CURSOR POSITION
=3476 8BD8                     MOV     BX,AX   ; AND MOVE IT TO BX
=3478 B95000                   MOV     CX,SCWID
=347B E80100      347F         CALL    SPCLEAR
=347E C3                       RET
=                       ;
=                       ;*** SPCLEAR   ENTRY: BX - START ADDRESS IN CRT RAM
=                       ;                     CX - NO. OF BYTES TO CLEAR
=                       ;              EXIT: ALL REGISTERS DESTROYED!
=                       ;

=                        SPCLEAR:
=347F 53                         PUSH    BX
=3480 E83CF0       31BF          CALL    SETCUR  ; SET CURSOR
=3483 5B                         POP     BX
=3484 E8BFFC       3146          CALL    SPCLEAR1
=3487 E8D2FE       335C          CALL    WRITEPOS
=348A C3                         RET
=                        ;
=                        ;*** SCROLLUP
=                        ;
=                        ;       ENTRY REGISTERS: NONE
=                        ;       EXIT REGISTERS:  ALL REGISTERS DESTROYED!
=                        SCROLLUP:
=348B A0DB44                     MOV     AL,CURROW
=348E 0AC0                       OR      AL,AL
=3490 742D         34BF          JZ      SCLUP3
=3492 8AE8                       MOV     CH,AL          ; CH = ROW NO.
=3494 B017                       MOV     AL,ROWS-1
=3496 2AC5                       SUB     AL,CH
=3498 7411         34AB          JZ      SCLUP2
=349A 8AC8                       MOV     CL,AL          ; CL = NO. OF ROWS TO MOVE
=349C E800FE       329F          CALL    CUROFF         ; TURN OFF CURSOR
=                        SCLUP1:
=349F E86D00       350F          CALL    MUROW   ; ROW NO. IN CH
=34A2 FEC5                       INC     CH             ; INCREMENT ROW NO.
=34A4 FEC9                       DEC     CL             ; DECREMENT NO. OF ROWS TO MOVE
=34A6 75F7         349F          JNZ     SCLUP1
=34A8 E809FE       32B4          CALL    CURON          ; TURN CURSOR BACK ON
=                        ;
=                        SCLUP2:
=34AB B017                       MOV     AL,ROWS-1
=34AD E8C2FF       3472          CALL    CLRLIN         ; CLEAR LINE
=34B0 C6060A4400                 MOV     CURCOL,0
=34B5 E8A4FE       335C          CALL    WRITEPOS
=34B8 C3                         RET
=                        SCLUP4:
=34B9 C7060A440017               MOV     WORD PTR CURCOL,1700H   ; LOAD COL/ROW WITH 0/23
=                        SCLUP3:
=34BF E8DDFD       329F          CALL    CUROFF
=34C2 BB8007                     MOV     BX,24*80
=34C5 E8F7FC       31BF          CALL    SETCUR
=34C8 E84DFD       3218          CALL    RDLIN
=34CB BB8007                     MOV     BX,24*80
=34CE B95000                     MOV     CX,80
=34D1 E8ABFF       347F          CALL    SPCLEAR        ; CLEAR STATUS LINE
=34D4 E82DFE       3304          CALL    SCROLLX
=34D7 BB8007                     MOV     BX,24*80
=34DA E8E2FC       31BF          CALL    SETCUR
=34DD E879FD       3259          CALL    WRLIN
=34E0 E8D1FD       32B4          CALL    CURON
=34E3 E876FE       335C          CALL    WRITEPOS
=34E6 C3                         RET
```

```

=                          ;
=                          ;*** SCOLLDN  - SCROLL DOWN  - ENTRY REGISTERS: NONE
=                          ;                              EXIT REGISTERS: ALL DESTROYED!
=                          ;
=                          SCROLLDN:
=34E7 A0DB44                       MOV     AL,CURROW
=34EA 50                           PUSH    AX
=34EB B117                         MOV     CL,ROWS-1
=34ED 2AC8                         SUB     CL,AL            ; CL = ROWS TO MOVE
=34EF 7411          3502           JZ      SCLDN2
=34F1 B516                         MOV     CH,ROWS-2        ; CH = ROW TO START
=34F3 E8A9FD        329F           CALL    CUROFF
=                          SCLDN1:
=34F6 E83400        352D           CALL    MDROW
=34F9 FECD                         DEC     CH
=34FB FEC9                         DEC     CL
=34FD 75F7          34F6           JNZ     SCLDN1
=34FF E8B2FD        32B4           CALL    CURON
=                          SCLDN2:
=3502 58                           POP     AX
=3503 E86CFF        3472           CALL    CLRLIN           ; CLEAR CURRENT LINE
=3506 C606DA4400                   MOV     CURCOL,0
=350B E84EFE        335C           CALL    WRITEPOS
=350E C3                           RET
=                          ;
=                          ;*** MUROW   MOVE ROW UP  - MOVE ROW [CH+1] TO ROW CH
=                          ;
=                          ;       Entry Register: CH = Row
=                          ;       Exit:           CX - Preserved  (Both CH and CL must be preserved!)
=                          ;                       AX, BX, DX  Destroyed
=                          ;
=                          MUROW:
=350F 51                           PUSH    CX
=3510 8AC5                         MOV     AL,CH
=3512 B150                         MOV     CL,SCWID
=3514 F6E1                         MUL     CL               ; AX = ROW * CHR/ROW
=3516 8BD0                         MOV     DX,AX
=3518 055000                       ADD     AX,SCWID         ; AX = (ROW+1)*(CHR/ROW)
=351B 8BD8                         MOV     BX,AX            ; DX = ROW B; BX = ROW B+1
=351D E89FFC        31BF           CALL    SETCUR           ; CURSOR TO THE START OF ROW B+1
=3520 E8F5FC        3218           CALL    RDLIN            ; READ IN A ROW (CHAR AND ATTRIBUTE)
=3523 8BDA                         MOV     BX,DX            ; NOW SET CURSOR TO START OF ROW B
=3525 E897FC        31BF           CALL    SETCUR
=3528 E82EFD        3259           CALL    WRLIN            ; WRITE OUT A ROW
=352B 59                           POP     CX
=352C C3                           RET
```

```
=                        ;
=                        ;*** MDROW   MOVE A ROW DOWN
=                        ;
=                        ;       Entry:  CH = row number
=                        ;       Exit:   AX, BX, DX destroyed
=                        ;               CX preserved

=                        ;
=                        MDROW:
=352D 51                         PUSH    CX
=352E 8AC5                       MOV     AL,CH
=3530 B150                       MOV     CL,SCWID
=3532 F6E1                       MUL     CL              ; MULTIPLY ROW NO. TIMES CHAR/ROW
=3534 8BD8                       MOV     BX,AX
=3536 8BD0                       MOV     DX,AX
=3538 E884FC        31BF         CALL    SETCUR          ; SET CURSOR TO START OF ROW B
=353B E8DAFC        3218         CALL    RDLIN           ; READ IN A ROW TO LINBUF
=353E 8BDA                       MOV     BX,DX
=3540 83C350                     ADD     BX,SCWID
=3543 E879FC        31BF         CALL    SETCUR          ; SET CURSOR TO START OF ROW B+1
=3546 E810FD        3259         CALL    WRLIN           ; WRITE ROW IN LINBUF
=3549 59                         POP     CX
=354A C3                         RET
```

```
=                                       INCLUDE C:DISKMGRC.SEG
=
=                               ;******************************************************************************
=                               ;
=                               ;
=                               ;   DISKINIT  -  INITIALIZE DISK SYSTEM
=                               ;
=                               ;
=                               ;   ENTRY VIA CALL
=                               ;
=                               ;
=                               ;   EXIT VIA RETURN
=                               ;
=                               ;
=                               ;******************************************************************************
=                               ;
=                               ;
=                               DISKINIT:
=
=                                       IF NOT LOADER_BIOS
=
=354B E82400        3572                CALL    INITTYP
=354E 2EA08D25                          MOV     AL,RSTC
=3552 FEC0                              INC     AL
=3554 A20748                            MOV     RETRIES,AL      ;SET RESTORE COUNTER FOR FLEX PIM
=3557 E82700        3581                CALL    CLOSE
=355A E8DB09        3F38                CALL    FIXREADY        ;IF THE WINCHESTER DRIVE IS READY,
=355D 7503          3562                JNZ     INITEND
=355F E8E909        3F4B                CALL    FIXDR           ;   THEN  RESTORE IT
=
=                                       ENDIF
=
=                               INITEND:
=3562 B100                              MOV     CL,0            ;SET DEFAULT TO DRIVE A
=3564 C3                                RET
=                               ;
=                               ;
=                               ;
=                               ;******************************************************************************
=                               ;
=                               ;
=                               ;   DISKWBOOT  -  WARM BOOT DISK SYSTEM
=                               ;
=                               ;
=                               ;   ENTRY VIA CALL
=                               ;
=                               ;
=                               ;   EXIT VIA RETURN
=                               ;
=                               ;
=                               ;******************************************************************************
=                               ;
=                               ;

=                               DISKWBOOT:
=3565 2EA08D25                          MOV     AL,RSTC
=3569 FEC0                              INC     AL
=356B A20748                            MOV     RETRIES,AL      ;SET RESTORE COUNTER FOR FLEX PIN
=356E E81000          3581              CALL    CLOSE
=3571 C3                                RET
=                               ;
=                               ;
=                               INITTYP:
=                                       ;
=                                       ;INITIALIZE DISK TYPE TABLE
=3572 B90400                            MOV     CX,4
=3575 BB0000                            MOV     BX,0
=                               ITLOOP:
=3578 C6873247FF                        MOV     DSKTYP[BX],0FFH
=357D 43                                INC     BX
=357E E2F8            3578              LOOP    ITLOOP
=3580 C3                                RET
=                               ;
=                               ;
=                               CLOSE:
=                                       ;
=                                       ;RESET READ/WRITE VARIABLES
=3581 B000                              MOV     AL,0
=3583 A26D47                            MOV     UNACNT,AL
=3586 A26B47                            MOV     HSTACT,AL
=3589 A26C47                            MOV     HSTWRT,AL
=358C C3                                RET

=
=                               ;****************************************************************************
=                               ;
=                               ;
=                               ;   HOME  -  MOVE TO TRACK 0
=                               ;
=                               ;
=                               ;   ENTRY VIA JMP
=                               ;
=                               ;
=                               ;   EXIT VIA RETURN
=                               ;
=                               ;
=                               ;****************************************************************************
=                               ;
=                               ;
=                               HOME:
=358D A06C47                            MOV     AL,HSTWRT       ;PENDING WRITE?
=3590 84C0                              TEST    AL,AL
=3592 7505            3599              JNZ     HOMED
=3594 C6066B4700                        MOV     HSTACT,0        ;NO, CLEAR HOST ACTIVE FLAG
=                               HOMED:
=3599 C70663470000                      MOV     SEKTRK,0        ;SET TRACK TO ZERO
=359F C3                                RET
```

```

=
=                       ;*****************************************************************************
=                       ;
=                       ;
=                       ;   SELDSK  - SELECT DISK DRIVE
=                       ;
=                       ;
=                       ;   ENTRY VIA JMP
=                       ;       CL - DISK DRIVE NUMBER
=                       ;       DL - BIT 0 = 0 IF FIRST SELECT
=                       ;            BIT 0 = 1 IF NOT FIRST SELECT
=                       ;
=                       ;
=                       ;   EXIT VIA RETURN
=                       ;       BX - DPH ADDRESS
=                       ;            0 IF INVALID DRIVE
=                       ;
=                       ;
=                       ;*****************************************************************************
=                       ;
=                       ;
=                       SELDSK:
=35A0 2E3A0E9425                CMP     CL,NUMHDSK     ;CHECK DRIVE NUMBER
=35A5 733E         35E5         JAE     SELERR
=                               ;
=                               ;VALID DRIVE NUMBER
=35A7 880E6247                  MOV     SEKDSK,CL      ;SEKDSK = DISK DRIVE NUMBER
=35AB F6C201                    TEST    DL,1           ;FIRST SELECT?
=35AE 7526         35D6         JNZ     GETDPH         ;NO, JUST NEED TO GET DPH ADDR
=                               ;
=                               ;THIS IS THE FIRST SELECT ON THIS DRIVE
=35B0 2E3A0E8A25                CMP     CL,NBRFLEX     ;CHECK DISK TYPE
=35B5 7313         35CA         JAE     HARDDISK
=                               ;
=                               ;FLEX DISK SELECT
=35B7 E82F00       35E9         CALL    FLUSH
=35BA E8C4FF       3581         CALL    CLOSE
=35BD E83400       35F4         CALL    GETTYP
=35C0 8A1E6247                  MOV     BL,SEKDSK
=35C4 E8A900       3670         CALL    INITDPB
=35C7 E90C00       35D6         JMP     GETDPH
=                       HARDDISK:
=
=                               IF NOT LOADER_BIOS
=
=35CA B003                      MOV     AL,3           ;ONLY ONE TYPE OF HARD DISK
=35CC 8A1E6247                  MOV     BL,SEKDSK
=35D0 E89D00       3670         CALL    INITDPB
=35D3 E8C900       369F         CALL    DHOME
=
=                               ENDIF
=

=                              GETDPH:
=35D6 B104                             MOV     CL,4
=35D8 8A1E6247                         MOV     BL,SEKDSK
=35DC B700                             MOV     BH,0
=35DE D3E3                             SHL     BX,CL           ;DRIVE NUMBER * 16
=35E0 81C3E644                         ADD     BX,OFFSET DPBASE;BX = DPH ADDRESS
=35E4 C3                               RET
=                              SELERR:
=35E5 BB0000                           MOV     BX,0000H
=35E8 C3                               RET
=                              ;
=                              ;
=                              FLUSH:
=35E9 A06C47                           MOV     AL,HSTWRT
=35EC 84C0                             TEST    AL,AL
=35EE 7403           35F3              JZ      ENDFLUSH
=35F0 E9FC02         38EF              JMP     WRITEHST
=                              ENDFLUSH:
=35F3 C3                               RET
```

```

=
=                          ;
=                          ;
=                          GETTYP:
=35F4 A06247                         MOV   AL,SEKDSK
=35F7 A26647                         MOV   HSTDSK,AL
=35FA A2E447                         MOV   DRV,AL
=35FD C606E54700                     MOV   HEAD,0
=3602 E87607        3D7B             CALL  DREST           ;FIRST RESTORE, THEN
=3605 E80B04        3A13             CALL  FLEXERR
=3608 3C52                           CMP   AL,'R'
=360A 74E8          35F4             JZ    GETTYP
=360C E8C307        3D02             CALL  DREADID         ;READ SECTOR LENGTH FROM DISK
=360F E80104        3A13             CALL  FLEXERR
=3612 3C52                           CMP   AL,'R'
=3614 74DE          35F4             JZ    GETTYP
=                          READSEC1:
=3616 C706FB47004C                   MOV   DMAADDR,OFFSET HSTBUF
=361C 8C1EFD47                       MOV   DMAADDR+2,DS
=3620 C606E34701                     MOV   CYLMODE,1
=3625 C606E54700                     MOV   HEAD,0
=362A C606E64700                     MOV   TRACK,0
=362F C606E74701                     MOV   SECTOR,1
=3634 C706E8470100                   MOV   SECCNT,1
=363A A0FA47                         MOV   AL,ERRBUF+6
=363D A20448                         MOV   BYTSEC,AL       ;BYTSEC = SECTOR LENGTH
=3640 E86205        3BA5             CALL  DREAD           ;READ FIRST SECTOR
=3643 E8CD03        3A13             CALL  FLEXERR
=3646 3C52                           CMP   AL,'R'
=3648 74CC          3616             JZ    READSEC1
=364A FC                             CLD
=364B BE0A4C                         MOV   SI,OFFSET HSTBUF+10
=364E BF6B36                         MOV   DI,OFFSET NCRTYP
=3651 B90500                         MOV   CX,5
=3654 F3A6                           REP CMPS AL,AL        ;CHECK FOR NCR TYPE DISK
=3656 7510          3668             JNZ   NOTNCR
=3658 B031                           MOV   AL,'1'
=365A 3804                           CMP   [SI],AL
=365C 7405          3663             JZ    DDSS
=365E B002                           MOV   AL,2            ;DDDS - TYPE 2
=3660 E90700        366A             JMP   RETTYP
=3663 B001                 DDSS:     MOV   AL,1            ;DDSS - TYPE 1
=3665 E90200        366A             JMP   RETTYP
=3668 B000                 NOTNCR:   MOV   AL,0            ;NON-NCR - TYPE 0
=366A C3                   RETTYP:   RET
=                          ;
=366B 4E43522046           NCRTYP    DB    'NCR F'
```

```

=
=                       ;
=                       ;
=                       INITDPB:
=3670 881E6647                  MOV     HSTDSK,BL
=3674 B700                      MOV     BH,0
=3676 88873247                  MOV     DSKTYP[BX],AL   ;DSKTYP[DRIVE] = TYPE
=367A 8AD8                      MOV     BL,AL
=367C BEE246                    MOV     SI,OFFSET DSKSPT
=367F 03F3                      ADD     SI,BX           ;SI = BEGIN OF DPB IN TYPE TABLE[TYPE]
=3681 BFE645                    MOV     DI,OFFSET DPB0
=3684 B80F00                    MOV     AX,LENGTH DPB0  ;LENGTH OF DPB
=3687 8A0E6247                  MOV     CL,SEKDSK
=368B B500                      MOV     CH,0
=368D F7E1                      MUL     CX              ;MULTIPLY BY DRIVE NUMBER
=368F 03F8                      ADD     DI,AX           ;DI = DPB[DRIVE NUMBER]
=3691 B90F00                    MOV     CX,LENGTH DPB0  ;MOVE LEN = DPB LEN
=3694 B80400                    MOV     AX,LENGTH DSKSPT;INCREMENT FOR SI
=3697 48                        DEC     AX
=                       DPBMOV:
=3698 FC                        CLD
=3699 A4                        MOVS    AL,AL
=369A 03F0                      ADD     SI,AX
=369C E2FA         3698         LOOP    DPBMOV
=369E C3                        RET
=                       ;
=                       ;
=                       DHOME:
=
=                               IF NOT LOADER_BIOS
=
=369F E89608       3F38         CALL    FIXREADY        ;IS WINCHESTER CONTROLLER READY?
=36A2 740D         36B1         JZ      READY
=36A4 BB7F47                    MOV     BX,OFFSET NOTRDY
=36A7 E8F403       3A9E         CALL    DISPERR
=36AA 3C52                      CMP     AL,'R'
=36AC 74F1         369F         JZ      DHOME
=36AE E92E00       36DF         JMP     ENDHOME
=                       READY:
=36B1 A06647                    MOV     AL,HSTDSK               ;SET UP PARM BLOCK FOR WINCHESTER PIM
=36B4 2E2A068A25                SUB     AL,NBRFLEX
=36B9 A20A48                    MOV     WIPAR+0,AL
=36BC C6060B4810                MOV     WIPAR+1,10H
=36C1 C6060C4800                MOV     WIPAR+2,0
=36C6 C6060D4800                MOV     WIPAR+3,0
=36CB C6060E4800                MOV     WIPAR+4,0
=36D0 C6060F4800                MOV     WIPAR+5,0
=36D5 E87308       3F48         CALL    FIXDR           ;RESTORE
=36D8 E8A903       3A84         CALL    FIXERR
=36DB 3C52                      CMP     AL,'R'
=36DD 74C0         369F         JZ      DHOME
=

=                               ENDIF
=
=                       ENDHOME:
=36DF C3                        RET
```

```

=
=                         ;***************************************************************************
=                         ;
=                         ;
=                         ;   SETTRK  -  SET TRACK NUMBER
=                         ;
=                         ;
=                         ;   ENTRY VIA JMP
=                         ;       CX - TRACK NUMBER
=                         ;
=                         ;
=                         ;   EXIT VIA RETURN
=                         ;       ALL PRESERVED
=                         ;
=                         ;
=                         ;***************************************************************************
=                         ;
=                         ;
=                         SETTRK:
=36E0 890E6347                    MOV     SEKTRK,CX
=36E4 C3                          RET
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;***************************************************************************
=                         ;
=                         ;
=                         ;   SETSEC  - SET SECTOR NUMBER
=                         ;
=                         ;
=                         ;   ENTRY VIA JMP
=                         ;       CX - SECTOR NUMBER
=                         ;
=                         ;
=                         ;   EXIT VIA RETURN
=                         ;       ALL PRESERVED
=                         ;
=                         ;
=                         ;***************************************************************************
=                         ;
=                         ;
=                         SETSEC:
=36E5 880E6547                    MOV     SEKSEC,CL       ;WE ONLY USE 1 BYTE OF SECTOR
=36E9 C3                          RET
```

```

=                         ;*********************************************************************************
=                         ;
=                         ;
=                         ;   SETDMA  - SET DMA OFFSET ADDRESS
=                         ;
=                         ;
=                         ;   ENTRY VIA JMP
=                         ;       CX - DMA OFFSET
=                         ;
=                         ;
=                         ;   EXIT VIA RETURN
=                         ;       ALL PRESERVED
=                         ;
=                         ;
=                         ;*********************************************************************************
=                         ;
=                         ;
=                         SETDMA:
=36EA 890E7847                    MOV     DMAOFF,CX
=36EE C3                          RET
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;*********************************************************************************
=                         ;
=                         ;
=                         ;   SETDMAB  - SET DMA SEGMENT ADDRESS
=                         ;
=                         ;
=                         ;   ENTRY VIA JMP
=                         ;       CX - DMA SEGMENT
=                         ;
=                         ;
=                         ;   EXIT VIA RETURN
=                         ;       ALL PRESERVED
=                         ;
=                         ;
=                         ;*********************************************************************************
=                         ;
=                         ;
=                         SETDMAB:
=36EF 890E7647                    MOV     DMASEG,CX
=36F3 C3                          RET
```

```

 =                          ;*****************************************************************************
 =                          ;
 =                          ;
 =                          ;   SECTRAN  -  SECTOR TRANSLATE
 =                          ;
 =                          ;
 =                          ;   ENTRY VIA JMP
 =                          ;       CX - SECTOR NUMBER
 =                          ;       DX - TRANSLATE TABLE OFFSET
 =                          ;
 =                          ;
 =                          ;   EXIT VIA RETURN
 =                          ;       BX - TRANSLATED SECTOR NUMBER
 =                          ;       ALL OTHERS PRESERVED
 =                          ;
 =                          ;
 =                          ;*****************************************************************************
 =                          ;
 =                          ;
 =                          SECTRAN:
 =                                  ;TRANSLATE SECTOR NUMBER CX WITH TABLE AT [DX]
 =36F4 85D2                         TEST    DX,DX   ;TEST FOR HARD SKEWED
 =36F6 7407          36FF           JZ      NOTRAN  ;BLOCKED MUST BE HARD SKEWED
 =36F8 8BD9                         MOV     BX,CX
 =36FA 03DA                         ADD     BX,DX
 =36FC 8A1F                         MOV     BL,[BX]
 =36FE C3                           RET
 =                          NOTRAN:
 =                                  ;HARD SKEWED DISK, PHYSICAL = LOGICAL SECTOR
 =36FF 8BD9                         MOV     BX,CX
 =3701 C3                           RET


 =
 =                          ;*****************************************************************************
 =                          ;
 =                          ;
 =                          ;   READ  - READ ONE SECTOR FROM DISK
 =                          ;
 =                          ;
 =                          ;   ENTRY VIA JMP
 =                          ;
 =                          ;
 =                          ;   EXIT VIA RETURN
 =                          ;       AL - 0 = NO ERROR
 =                          ;            1 = NON-RECOVERABLE ERROR
 =                          ;
 =                          ;
 =                          ;*****************************************************************************
 =                          ;
 =                          ;
 =                          READ:
 =3702 C606D04700                   MOV     UNACNT,0
 =3707 C606744701                   MOV     READOP,1
 =370C C606734701                   MOV     RSFLAG,1
 =3711 C606754702                   MOV     WRTYPE,WRUAL
 =3716 E98900        37A2           JMP     RWOPER
 
 =
 =                                 ;*****************************************************************************
 =                                 ;
 =                                 ;
 =                                 ;   WRITE  -  WRITE ONE SECTOR TO DISK
 =                                 ;
 =                                 ;
 =                                 ;   ENTRY VIA JMP
 =                                 ;       CL - 0 = NORMAL SECTOR WRITE
 =                                 ;            1 = WRITE TO DIRECTORY SECTOR
 =                                 ;            2 = WRITE TO FIRST SECTOR OF A NEW ALLOCATION BLOCK
 =                                 ;
 =                                 ;
 =                                 ;   EXIT VIA RETURN
 =                                 ;       AL - 0 = NO ERROR
 =                                 ;            1 = NON-RECOVERABLE ERROR
 =                                 ;
 =                                 ;
 =                                 ;*****************************************************************************
 =                                 ;
 =                                 ;
 =                                 WRITE:
 =                                         ;WRITE THE SELECTED CP/M SECTOR
 =3719 C606744700                          MOV     READOP,0        ;WRITE OPERATION
 =371E 880E7547                            MOV     WRTYPE,CL
 =3722 80F902                              CMP     CL,WRUAL        ;WRITE UNALLOCATED?
 =3725 7523            374A                JNZ     CHKUNA          ;CHECK FOR UNALLOC
 =                                         ;
 =                                         ;FIRST WRITE TO NEW ALLOC BLOCK, SET PARAMETERS
 =3727 8A1E6247                            MOV     BL,SEKDSK
 =372B B700                                MOV     BH,0
 =372D 8A9F3247                            MOV     BL,DSKTYPE[BX]
 =3731 8A872247                            MOV     AL,DSKCNT[BX]
 =3735 A26D47                              MOV     UNACNT,AL       ;UNACNT = CP/M SECTORS/ALLOC BLOCK
 =3738 A06247                              MOV     AL,SEKDSK
 =373B A26E47                              MOV     UNADSK,AL       ;UNADSK = SEKDSK
 =373E A16347                              MOV     AX,SEKTRK
 =3741 A36F47                              MOV     UNATRK,AX       ;UNATRK = SEKTRK
 =3744 A06547                              MOV     AL,SEKSEC
 =3747 A27147                              MOV     UNASEC,AL       ;UNASEC = SEKSEC
 =                                 CHKUNA:
 =                                         ;CHECK FOR WRITE TO UNALLOCATED SECTOR
 =374A BB6D47                              MOV     BX,OFFSET UNACNT;POINT "UNA" AT UNACNT
 =374D 8A07                                MOV     AL,UNA
 =374F 84C0                                TEST    AL,AL           ;ANY UNALLOC REMAIN?
 =3751 7445            3798                JZ      ALLOC           ;SKIP IF NOT
 =                                         ;
 =                                         ;MORE UNALLOCATED RECORDS REMAIN
 =3753 FEC8                                DEC     AL
 =3755 8807                                MOV     UNA,AL          ;UNACNT = UNACNT-1
 =3757 A06247                              MOV     AL,SEKDSK       ;SAME DISK?
 =375A BB6E47                              MOV     BX,OFFSET UNADSK
```

```
=375D 3A07                      CMP     AL,UNA          ;SEKDSK = UNADSK?
=375F 7537         3798         JNZ     ALLOC           ;SKIP IF NOT
=                               ;
=                               ;DISKS ARE THE SAME
=3761 A16F47                    MOV     AX,UNATRK
=3764 3B066347                  CMP     AX,SEKTRK
=3768 752E         3798         JNZ     ALLOC           ;SKIP IF NOT
=                               ;
=                               ;TRACKS ARE THE SAME
=376A A06547                    MOV     AL,SEKSEC       ;SAME SECTOR?
=376D BB7147                    MOV     BX,OFFSET UNASEC;POINT UNA AT UNASEC
=3770 3A07                      CMP     AL,UNA          ;SEKSEC =UNASEC?
=3772 7524         3798         JNZ     ALLOC           ;SKIP IF NOT
=                               ;
=                               ;MATCH, MOVE TO NEXT SECTOR FOR FUTURE REF
=3774 FE07                      INC     UNA             ;UNASEC = UNASEC+1
=3776 8A17                      MOV     DL,UNA
=3778 53                        PUSH    BX
=3779 8A1E6247                  MOV     BL,SEKDSK
=377D B700                      MOV     BH,0
=377F 8A9F3247                  MOV     BL,DSKTYP[BX]
=3783 3A97E246                  CMP     DL,DSKSPT[BX]   ;END OF TRACK?
=3787 5B                        POP     BX
=3788 7207         3791         JB      NOOVF           ;SKIP IF BELOW
=                               ;
=                               ;OVERFLOW TO NEXT TRACK
=378A C60700                    MOV     UNA,0           ;UNASEC = 0
=378D FF066F47                  INC     UNATRK          ;UNATRK = UNATRK+1
=                       NOOVF:
=                               ;MATCH FOUND, MARK AS UNNECESSARY READ
=3791 C606734700                MOV     RSFLAG,0        ;RSFLAG = 0
=3796 EB0A         37A2         JMPS    RWOPER          ;TO PERFORM THE WRITE
=                       ALLOC:
=                               ;NOT AN UNALLOCATED RECORD, REQUIRES PRE-READ
=3798 C6066D4700                MOV     UNACNT,0        ;UNACNT = 0
=379D C606734701                MOV     RSFLAG,1        ;RSFLAG = 1
=                               ;DROP THROUGH TO RWOPER
```

```

=
=                             ;************************************************************************
=                             ;
=                             ;   READ/WRITE OPERATION
=                             ;   COMMON CODE FOR READ AND WRITE
=                             ;
=                             ;************************************************************************
=                             ;
=                             ;
=                             RWOPER:
=                                     ;ENTER HERE TO PERFORM THE READ/WRITE
=37A2 C606724700                      MOV     ERFLAG,0        ;NO ERRORS (YET)
=37A7 8A1E6247                        MOV     BL,SEKDSK
=37AB B700                            MOV     BH,0
=37AD 8A9F3247                        MOV     BL,DSKTYP[BX]
=37B1 8A8FDA46                        MOV     CL,DSKSLC[BX]
=37B5 A06547                          MOV     AL,SEKSEC
=37B8 D2E8                            SHR     AL,CL
=37BA A26A47                          MOV     SEKHST,AL       ;PHYSICAL SECTOR
=                                     ;
=                                     ;ACTIVE HOST SECTOR?
=37BD B001                            MOV     AL,1
=37BF 86066B47                        XCHG    AL,HSTACT       ;ALWAYS BECOMES 1
=37C3 84C0                            TEST    AL,AL           ;WAS IT ALREADY?
=37C5 7425              37EC          JZ      FILHST          ;FILL HOST IF NOT
=                                     ;
=                                     ;HOST BUFFER ACTIVE, SAME AS SEEK BUFFER?
=37C7 A06247                          MOV     AL,SEKDSK
=37CA 3A066647                        CMP     AL,HSTDSK               ;SEKDSK = PHYSICAL DRIVE?
=37CE 7512              37E2          JNZ     NOMATCH
=                                     ;
=                                     ;SAME DISK, SAME TRACK?
=37D0 A16747                          MOV     AX,HSTTRK
=37D3 3B066347                        CMP     AX,SEKTRK       ;PHYSICAL TRACK SAME AS SEEK TRACK
=37D7 7509              37E2          JNZ     NOMATCH
=                                     ;
=                                     ;SAME DISK, SAME TRACK, SAME BUFFER?
=37D9 A06A47                          MOV     AL,SEKHST
=37DC 3A066947                        CMP     AL,HSTSEC       ;SEKHST = PHYSICAL SECTOR?
=37E0 742B              380D          JZ      MATCH           ;SKIP IF MATCH
=                             NOMATCH:
=                                     ;PROPER DISK, BUT NOT CORRECT SECTOR
=37E2 A06C47                          MOV     AL,HSTWRT
=37E5 84C0                            TEST    AL,AL           ;"DIRTY" BUFFER?
=37E7 7403              37EC          JZ      FILHST          ;NO, DON'T NEED TO WRITE
=37E9 E80301            38EF          CALL    WRITEHST        ;YES, CLEAR HOST BUFFER
=                             FILHST:
=                                     ;MAY HAVE TO FILL THE HOST BUFFER
=37EC A06247                          MOV     AL,SEKDSK
=37EF A26647                          MOV     HSTDSK,AL
=37F2 A16347                          MOV     AX,SEKTRK
=37F5 A36747                          MOV     HSTTRK,AX
```

```

=37F8 A06A47                    MOV     AL,SEKHST
=37FB A26947                    MOV     HSTSEC,AL
=37FE A07347                    MOV     AL,RSFLAG
=3801 84C0                      TEST    AL,AL          ;NEED TO READ?
=3803 7403         3808         JZ      FILHSTL
=3805 E88300       3888         CALL    READHST        ;YES, IF 1
=                       FILHSTL:
=3808 C6066C4700                MOV     HSTWRT,0       ;NO PENDING WRITE
=                       MATCH:
=                               ;COPY DATA TO OR FROM BUFFER DEPENDING ON "READOP'
=380D 8A1E6647                  MOV     BL,HSTDSK
=3811 B700                      MOV     BH,0
=3813 8A9F3247                  MOV     BL,DSKTYP[BX]
=3817 8A870E46                  MOV     AL,DSKSMA[BX]
=381B 22066547                  AND     AL,SEKSEC
=381F 98                        CBW
=3820 B107                      MOV     CL,7
=3822 D3E0                      SHL     AX,CL          ;SHIFT LEFT 7 (* 128 = 2**7)
=                               ;
=                               ;AX HAS RELATIVE HOST BUFFER OFFSET
=3824 05004C                    ADD     AX,OFFSET HSTBUF;AX HAS BUFFER ADDRESS
=3827 8BF0                      MOV     SI,AX          ;PUT IN SOURCE INDEX REGISTER
=3829 8B3E7847                  MOV     DI,DMAOFF      ;USER BUFFER IS DEST IF READOP
=382D 06                        PUSH    ES
=382E 1E                        PUSH    DS             ;SAVE SEGMENT REGISTERS
=382F 8E067647                  MOV     ES,DMASEG      ;SET DESTSEG TO THE USERS SEG
=3833 B94000                    MOV     CX,128/2       ;LENGTH OF MOVE IN WORDS
=3836 A07447                    MOV     AL,READOP
=3839 84C0                      TEST    AL,AL          ;WHICH WAY?
=383B 750F         384C         JNZ     RWMOVE         ;SKIP IF READ
=                               ;
=                               ;WRITE OPERATION, MARK AND SWITCH DIRECTION
=383D C6066C4701                MOV     HSTWRT,1       ;HSTWRT = 1 (DIRTY BUFFER NOW)
=3842 87F7                      XCHG    SI,DI          ;SOURCE/DEST INDEX SWAP
=3844 8CD8                      MOV     AX,DS
=3846 8EC0                      MOV     ES,AX
=3848 8E1E7647                  MOV     DS,DMASEG      ;SETUP DS,ES FOR WRITE
=                       RWMOVE:
=384C FC                        CLD
=384D F3A5                      REP MOVS AX,AX         ;MOVE AS 16 BIT WORDS
=384F 1F                        POP     DS
=3850 1E                        PUSH    DS
=3851 8A1E6647                  MOV     BL,HSTDSK
=3855 B700                      MOV     BH,0
=3857 8A9F3247                  MOV     BL,DSKTYP[BX]
=385B 8A870646                  MOV     AL,DSKSID[BX]
=385F A880                      TEST    AL,10000000B
=3861 740C         386F         JZ      NOCOMP         ;COMPLEMENT BIT ON?
=3863 8CC0                      MOV     AX,ES         ';YES, SET UP TO COMPLEMENT DATA
=3865 8ED8                      MOV     DS,AX
=3867 B98000                    MOV     CX,128
=                       COMPLOOP:

 =386A 4F                         DEC    DI              ;GO BACKWARDS THROUGH BUFFER
 =386B F615                       NOT    BYTE PTR [DI]   ;COMPLEMENT EACH BYTE
 =386D E2FB          386A         LOOP   COMPLOOP
 =                         NOCOMP:
 =336F 1F                         POP    DS
 =3870 07                         POP    ES              ;RESTORE SEGMENT REGISTERS
 =                                ;
 =                                ;DATA HAS BEEN MOVED TO/FROM HOST BUFFER
 =3871 803E754701                 CMP    WRTYPE,WRDIR    ;WRITE TYPE TO DIRECTORY?
 =3876 A07247                     MOV    AL,ERFLAG       ;IN CASE OF ERRORS
 =3879 750F          388A         JNZ    RETURNRW        ;NO FURTHER PROCESSING
 =                                ;
 =                                ;CLEAR HOST BUFFER FOR DIRECTORY WRITE
 =387B 84C0                       TEST   AL,AL           ;ERRORS?
 =387D 750B          388A         JNZ    RETURNRW        ;SKIP IF SO
 =387F C6066C4700                 MOV    HSTWRT,0        ;BUFFER WRITTEN
 =3884 E86800        38EF         CALL   WRITEHST
 =3887 A07247                     MOV    AL,ERFLAG
 =                         RETURNRW:
 =388A C3                         RET

 =                              ;***************************************************************************
 =                              ;
 =                              ;
 =                              ;   HOST DISK OPERATIONS
 =                              ;
 =                              ;
 =                              ;***************************************************************************
 =                              ;
 =                              READHST:
 =388B 8C1E7A47                         MOV     PMAADDR,DS
 =388F C7067C47004C                     MOV     PMAADDR+2,OFFSET HSTBUF
 =3895 E8BB00          3953             CALL    LOGLAC
 =                              SREAD:
 =3898 A06647                           MOV     AL,HSTDSK
 =389B 2E3A068A25                       CMP     AL,NBRFLEX
 =38A0 730C            38AE             JAE     FIXREAD
 =38A2 E8D600          397B             CALL    SETFLXVAR
 =38A5 E8FD02          3BA5             CALL    DREAD
 =38A8 E86801          3A13             CALL    FLEXERR
 =38AB E93100          38DF             JMP     READEND
 =                              FIXREAD:
 =
 =                                      IF NOT LOADER_BIOS
 =
 =38AE 2E8A0E8D25                       MOV     CL,RSTC         ;WINCHESTER PIM DOES NO RETRIES,
 =38B3 B500                             MOV     CH,0            ;    SO WE BETTER
 =38B5 FEC1                             INC     CL
 =                              FIXRETRY:
 =38B7 51                               PUSH    CX
 =38B8 E88301          3A3E             CALL    SETFIXVAR
 =38BB C6060B4820                       MOV     WIPAR+1,20H
 =38C0 E88806          3F4B             CALL    FIXDR           ;READ
 =38C3 A00E48                           MOV     AL,WIPAR+4
 =38C6 A801                             TEST    AL,00000001B
 =38C8 59                               POP     CX
 =38C9 7411            38DC             JZ      FIXCONT         ;GO OUT OF RETRY LOOP IF NO ERROR
 =38CB 83F901                           CMP     CX,1
 =38CE 740C            38DC             JZ      FIXCONT         ;    OR END OF RETRIES
 =38D0 51                               PUSH    CX
 =38D1 C6060B4810                       MOV     WIPAR+1,10H
 =38D6 E87206          3F4B             CALL    FIXDR           ;RESTORE
 =38D9 59                               POP     CX
 =38DA E2DB            38B7             LOOP    FIXRETRY
 =                              FIXCONT:
 =38DC E8A501          3A84             CALL    FIXERR
 =
 =                                      ENDIF
 =
 =                              READEND:
 =38DF 3C52                             CMP     AL,'R'          ;CHECK FOR USER REQUEST TO RETRY
 =38E1 74B5            3898             JZ      SREAD

 =38E3 3C00                             CMP     AL,0
 =38E5 7407            38EE             JZ      READRET
 =38E7 B0FF                             MOV     AL,0FFH
 =38E9 C606724701                       MOV     ERFLAG,1
 =                              READRET:
 =38EE C3                               RET
```

```

=
=                       ;
=                       ;
=                       WRITEHST:
=38EF 8C1E7A47                  MOV     PMAADDR,DS
=38F3 C7067C47004C              MOV     PMAADDR+2,OFFSET HSTBUF
=38F9 E85700       3953         CALL    LOGLAC
=                       SWRITE:
=38FC A06647                    MOV     AL,HSTDSK
=38FF 2E3A068A25                CMP     AL,NBRFLEX
=3904 730C         3912         JAE     FIXWRITE
=3906 E87200       397B         CALL    SETFLXVAR
=3909 E8A302       3BAF         CALL    DWRITE
=390C E80401       3A13         CALL    FLEXERR
=390F E93100       3943         JMP     WRITEEND
=                       FIXWRITE:
=
=                               IF NOT LOADER_BIOS
=
=3912 2E8A0E8D25                MOV     CL,RSTC         ;WINCHESTER PIM DOES NOT DO RETRIES,
=3917 B500                      MOV     CH,0            ;    SO WE BETTER
=3919 FEC1                      INC     CL
=                       FIXRTRY:
=391B 51                        PUSH    CX
=391C E81F01       3A3E         CALL    SETFIXVAR
=391F C6060B4830                MOV     WIPAR+1,30H
=3924 E82406       3F4B         CALL    FIXDR           ;WRITE
=3927 A00E48                    MOV     AL,WIPAR+4
=392A A801                      TEST    AL,00000001B
=392C 59                        POP     CX
=392D 7411         3940         JZ      FIXCON          ;GO OUT OF RETRY LOOP IF NO ERROR
=392F 83F901                    CMP     CX,1
=3932 740C         3940         JZ      FIXCON          ;    OR END OF RETRIES
=3934 51                        PUSH    CX
=3935 C6060B4810                MOV     WIPAR+1,10H
=393A E80E06       3F4B         CALL    FIXDR           ;RESTORE
=393D 59                        POP     CX
=393E E208         391B         LOOP    FIXRTRY
=                       FIXCON:
=3940 E84101       3A84         CALL    FIXERR
=
=                               ENDIF
=
=                       WRITEEND:
=3943 3C52                      CMP     AL,'R'          ;CHECK FOR USER REQUEST TO RETRY
=3945 74B5         38FC         JZ      SWRITE
=3947 3C00                      CMP     AL,0
=3949 7407         3952         JZ      WRITERET
=394B B0FF                      MOV     AL,0FFH
=394D C606724701                MOV     ERFLAG,1
=                       WRITERET:
=3952 C3                        RET
```

```

=
=                       ;
=                       ;
=                       LOGLAC:
=                               ;
=                               ;NEEDED FOR DISKS THAT HAVE LOGICAL SECTOR LACING (NON-DM5)
=3953 A06947                    MOV     AL,HSTSEC
=3956 8A1E6647                  MOV     BL,HSTDSK
=395A B700                      MOV     BH,0
=395C 8A9F3247                  MOV     BL,DSKTYP[BX]
=3960 8A871E47                  MOV     AL,DSKDBL[BX]
=3964 84C0                      TEST    AL,AL
=3966 A06947                    MOV     AL,HSTSEC
=3969 740A         3975         JZ      NOLAC
=396B 98                        CBW
=396C 8BD8                      MOV     BX,AX
=396E 8AB74247                  MOV     AL,XLT[BX]      ;TRANSLATED SECTOR
=3972 E90200       3977         JMP     CONLAC
=                       NOLAC:
=3975 FEC0                      INC     AL
=                       CONLAC:
=3977 A2E747                    MOV     SECTOR,AL
=397A C3                        RET
```

```

 =                          ;
 =                          ;
 =                          SETFLXVAR:
 =                                 ;SET VARIABLES FOR FLEX DISK PIM
 =                                 ;
 =397B A17C47                      MOV     AX,PMAADDR+2
 =397E A3FB47                      MOV     DMAADDR,AX
 =3981 A17A47                      MOV     AX,PMAADDR
 =3984 A3FD47                      MOV     DMAADDR+2,AX
 =3987 C606E34701                  MOV     CYLMODE,1        ;START WITH NOT CYL MODE, MAYBE WILL CHANGE
 =398C C706E8470100                MOV     SECCNT,1         ;READ/WRITE ONE SECTOR AT A TIME
 =3992 C606E54700                  MOV     HEAD.0           ;START WITH 0, WILL CHANGE IF NEEDED
 =3997 A06647                      MOV     AL,HSTDSK
 =399A A2E447                      MOV     DRV,AL
 =399D A16747                      MOV     AX,HSTTRK
 =39A0 A2E647                      MOV     TRACK,AL         ;FLEX DISK ONLY NEEDS 1 BYTE OF TRACK
 =39A3 8A1EE447                    MOV     BL,DRV
 =39A7 B700                        MOV     BH,0
 =39A9 8A9F3247                    MOV     BL,DSKTYP[BX]
 =39AD 8A870A46                    MOV     AL,DSKSLC[BX]
 =39B1 A20448                      MOV     BYTSEC,AL
 =39B4 8A872E47                    MOV     AL,DSKMSC[BX]
 =39B8 A20248                      MOV     SECTRK,AL
 =39BB 8A870646                    MOV     AL,DSKSID[BX]
 =39BF A807                        TEST    AL,00000111B
 =39C1 744F          3A12          JZ      SETRET           ;DOSS, NOTHING CHANGES
 =39C3 A806                        TEST    AL,00000110B
 =39C5 7535          39FC          JNZ     CYLMOD           ;CYLINDER MODE RECORDING
 =39C7 8A1EE447                    MOV     BL,DRV  ;DDDS
 =39CB B700                        MOV     BH,0
 =39CD 8A9F3247                    MOV     BL,DSKTYP[BX]
 =39D1 8A872647                    MOV     AL,DSKTRK[BX]
 =39D5 D0C8                        ROR     AL,1             ;AL = NO OF TRACKS PER SIDE
 =39D7 3806E647                    CMP     TRACK,AL         ;TRACK > NO OF TRACKS PER SIDE?
 =39DB 7235          3A12          JB      SETRET
 =39DD 2806E647                    SUB     TRACK,AL         ;YES, SUBTRACT NO PER SIDE
 =39E1 C606E54701                  MOV     HEAD,1           ; AND GO TO SIDE 2
 =39E6 50                          PUSH    AX
 =39E7 8A870646                    MOV     AL,DSKSID[BX]
 =39EB A840                        TEST    AL,01000000B     ;DOES THIS DISK HAVE 'REVERSE RECORDING'
 =39ED 58                          POP     AX               ; ON SIDE 2?
 =39EE 7422          3A12          JZ      SETRET
 =39F0 2C01                        SUB     AL,1             ;YES, TRANSLATE THE TRACK NO
 =39F2 2A06E647                    SUB     AL,TRACK
 =39F6 A2E647                      MOV     TRACK,AL
 =39F9 E91600        3A12          JMP     SETRET
 =                          CYLMOD:
 =39FC C606E34700                  MOV     CYLMODE,0
 =3A01 A0E747                      MOV     AL,SECTOR
 =3A04 A880                        TEST    AL,10000000B     ;IF SECTOR HIGH BIT IS 1, SIDE IS 1
 =3A06 740A          3A12          JZ      SETRET
```

```

=3A08 C606E54701                 MOV     HEAD,1
=3A0D 247F                       AND     AL,01111111B
=3A0F A2E747                     MOV     SECTOR,AL       ;STRIP HIGH BIT OF SECTOR
=                        SETRET:
=3A12 C3                         RET
=                        ;
=                        ;
=                        FLEXERR:
=3A13 A0F447                     MOV     AL,ERRBUF
=3A16 A8C0                       TEST    AL,11000000B    ;CHECK FOR SUCCESSFUL FUNCTION
=3A18 B000                       MOV     AL,0
=3A1A 7421          3A3D         JZ      FLEXEND
=3A1C A0F447                     MOV     AL,ERRBUF
=3A1F A808                       TEST    AL,00001000B    ;CHECK FOR NOT READY
=3A21 BB7F47                     MOV     BX,OFFSET NOTRDY
=3A24 7514          3A3A         JNZ     FLEXDISP
=3A26 A0F547                     MOV     AL,ERRBUF+1
=3A29 A802                       TEST    AL,00000010B    ;CHECK FOR WRITE PROTECT
=3A2B BB9247                     MOV     BX,OFFSET PROTECT
=3A2E 750A          3A3A         JNZ     FLEXDISP
=3A30 A895                       TEST    AL,10010101B    ;CHECK FOR FATAL ERROR
=3A32 BBAB47                     MOV     BX,OFFSET FATAL
=3A35 7503          3A3A         JNZ     FLEXDISP
=3A37 BBC247                     MOV     BX,OFFSET IOERR ;ELSE, I/O ERROR
=                        FLEXDISP:
=3A3A E86100        3A9E         CALL    DISPERR
=                        FLEXEND:
=3A3D C3                         RET
```

```

=
=                         ;
=                         ;
=                         SETFIXVAR:
=
=                                 IF NOT LOADER_BIOS
=
=3A3E E8F704        3F38          CALL    FIXREADY        ;IS WINCHESTER CONTROLLER READY?
=3A41 740D          3A50          JZ      SET
=3A43 BB7F47                      MOV     BX,OFFSET NOTRDY
=3A46 E85500        3A9E          CALL    DISPERR
=3A49 3C52                        CMP     AL,'R'
=3A4B 74F1          3A3E          JZ      SETFIXVAR
=3A4D E93300        3A83          JMP     SETEND
=                         SET:
=3A50 A17A47                      MOV     AX,PMAADDR
=3A53 A31048                      MOV     WORD PTR WIPAR+6,AX
=3A56 A17C47                      MOV     AX,PMAADDR+2
=3A59 A31248                      MOV     WORD PTR WIPAR+8,AX
=3A5C A06647                      MOV     AL,HSTDSK               ;YES, READY
=3A5F 2E2A068A25                  SUB     AL,NBRFLEX
=3A64 A20A48                      MOV     WIPAR+0,AL
=3A67 B81100                      MOV     AX,11H          ;SPECIFIC TO WINCHESTER DISK,
=3A6A F7266747                    MUL     HSTTRK          ;  MUST CHANGE IF ANOTHER FIXED DISK
=3A6E 8A1E6947                    MOV     BL,HSTSEC       ;  IS ADDED
=3A72 B700                        MOV     BH,0
=3A74 03C3                        ADD     AX,BX
=3A76 A30C48                      MOV     WORD PTR WIPAR+2,AX
=3A79 C6060E4800                  MOV     WIPAR+4,0
=3A7E C6060F4800                  MOV     WIPAR+5,0
=                         SETEND:
=
=                                 ENDIF
=
=3A83 C3                          RET
=                         ;
=                         ;
=                         ;
=                         FIXERR:
=
=                                 IF NOT LOADER_BIOS
=
=3A84 A00E48                      MOV     AL,WIPAR+4
=3A87 A801                        TEST    AL,00000001B
=3A89 B000                        MOV     AL,0
=3A8B 7410          3A9D          JZ      FIXEND
=3A8D A00F48                      MOV     AL,WIPAR+5
=3A90 A860                        TEST    AL,01100000B
=3A92 BBC247                      MOV     BX,OFFSET IOERR
=3A95 7503          3A9A          JNZ     FIXDISP
=3A97 BBAB47                      MOV     BX,OFFSET FATAL
=                         FIXDISP:
```

```

=3A9A E80100        3A9E           CALL    DISPERR
=                          FIXEND:
=
=                                  ENDIF
=
=3A9D C3                           RET

=
=                          ;
=                          ;
=                          ;
=                          DISPERR:
=3A9E 803E7E4700                   CMP     DISPFLAG,0
=3AA3 7520          3AC5           JNZ     ERR_RET         ;IF NO MESSAGES TO BE DISPLAYED, JUST RET
=3AA5 A06647                       MOV     AL,HSTDSK
=3AA8 FEC0                         INC     AL
=3AAA 0C40                         OR      AL,40H
=3AAC 8807                         MOV     [BX],AL
=                          DISP:
=3AAE E884F4        2F35           CALL    ERR_DISP        ; CALL ROUTINE TO DISPLAY THE ERROR MESSAGE
=3AB1 3C52                         CMP     AL,'R'
=3AB3 7408          3AC0           JZ      ERREND
=3AB5 3C4F                         CMP     AL,'O'
=3AB7 7407          3AC0           JZ      ERREND
=3AB9 3C58                         CMP     AL,'X'
=3ABB 75F1          3AAE           JNZ     DISP            ; INVALID RESPONSE, TRY AGAIN
=3ABD E902F0        2B92           JMP     WBOOT           ; ABORT, DO A WARM BOOT
=
=                          ERREND:
=3AC0 50                           PUSH    AX
=3AC1 E809F5        2FCD           CALL    ERR_DISP1
=3AC4 58                           POP     AX
=                          ERR_RET:
=3AC5 C3                           RET
```

```

=
=                       ;*********************************************************************
=                       ;                                                                    *
=                       ;                                                                    *
=                       ;   SPECFUN  -  SPECIAL BIOS FUNCTIONS FOR UTILITIES                 *
=                       ;                                                                    *
=                       ;                                                                    *
=                       ;   ENTRY VIA JMP                                                    *
=                       ;       CL - FUNCTION NUMBER                                         *
=                       ;                                                                    *
=                       ;                                                                    *
=                       ;   EXIT VIA RETURN                                                  *
=                       ;                                                                    *
=                       ;                                                                    *
=                       ;*********************************************************************
=                       ;
=                       ;
=                       SPECFUN:
=
=                               IF NOT LOADER_BIOS
=
=3AC6 8AC1                      MOV     AL,CL
=3AC8 D0E0                      SHL     AL,1
=3ACA 98                        CBW
=3ACB BED23A                    MOV     SI,OFFSET SFUNCTAB
=3ACE 03F0                      ADD     SI,AX
=3AD0 FF24                      JMP     WORD PTR [SI]
=                       ;
=                       SFUNCTAB:
=3AD2 EE3A                      DW      NOTIMPL         ; 0 - NOT USED
=3AD4 EF3A                      DW      SWRTRK          ; 1 - WRITE TRACK
=3AD6 223B                      DW      SRDTRK          ; 2 - READ TRACK (FLEX DISK ONLY)
=3AD8 3C3B                      DW      SSETDMA         ; 3 - SET DMA OFFSET
=3ADA 413B                      DW      SSELDSK         ; 4 - SELECT DISK
=3ADC 5D3B                      DW      SSETTRK         ; 5 - SET TRACK
=3ADE 643B                      DW      SSETSEC         ; 6 - SET SECTOR
=3AE0 983B                      DW      SREAD           ; 7 - READ
=3AE2 FC3B                      DW      SWRITE          ; 8 - WRITE
=3AE4 6D3B                      DW      SHOME           ; 9 - HOME
=3AE6 5F3B                      DW      SSETTRK2        ; A - SET TRACK (TWO BYTES)
=3AE8 773B                      DW      SEREAD          ; B - READ WITH ERROR RETURNED
=3AEA 853B                      DW      SEWRITE         ; C - WRITE WITH ERROR RETURNED
=3AEC 933B                      DW      SSETDMAB        ; D - SET DMA SEGMENT
=                       ;
=                       ;
=                       NOTIMPL:
=3AEE C3                        RET
=                       ;
=                       ;
=                       SWRTRK:
=3AEF A06647                    MOV     AL,HSTDSK
=3AF2 2E3A068A25                CMP     AL,NBRFLEX
```

```
=3AF7 7310            3B09                JAE     FIXWRTRK
=3AF9 88160648                            MOV     PATTERN,DL
=3AFD E87BFE          397B                CALL    SETFLXVAR
=3B00 E80303          3E06                CALL    DFORMAT
=3B03 E80DFF          3A13                CALL    FLEXERR
=3B06 E90E00          3B17                JMP     WRTRKEND
=                           FIXWRTRK:
=3B09 E832FF          3A3E                CALL    SETFIXVAR
=3B0C C6060B4850                          MOV     WIPAR+1,50H
=3B11 E83704          3F4B                CALL    FIXDR
=3B14 E86DFF          3A84                CALL    FIXERR
=                           WRTRKEND:
=3B17 3C52                                CMP     AL,'R'
=3B19 74D4            3AEF                JZ      SWRTRK
=3B1B 3C00                                CMP     AL,0
=3B1D 7402            3B21                JZ      WRTRKRET
=3B1F B0FF                                MOV     AL,0FFH
=                           WRTRKRET:
=3B21 C3                                  RET
=                           ;
=                           ;
=                           SRDTRK:
=3B22 E856FE          397B                CALL    SETFLXVAR
=3B25 C706E8470800                        MOV     SECCNT,8
=3B2B E87700          3BA5                CALL    DREAD
=3B2E E8E2FE          3A13                CALL    FLEXERR
=3B31 3C52                                CMP     AL,'R'
=3B33 74ED            3B22                JZ      SRDTRK
=3B35 3C00                                CMP     AL,0
=3B37 7402            3B3B                JZ      SRDRET
=3B39 B0FF                                MOV     AL,0FFH
=                           SRDRET:
=3B3B C3                                  RET
=                           ;
=                           ;
=                           SSETDMA:
=3B3C 89167C47                            MOV     PMAADDR+2,DX     ;OFFSET
=3B40 C3                                  RET
=                           ;
=                           ;
=                           SSELDSK:
=3B41 52                                  PUSH    DX
=3B42 E8A4FA          35E9                CALL    FLUSH
=3B45 E839FA          3581                CALL    CLOSE
=3B48 5A                                  POP     DX
=3B49 2E3A168A25                          CMP     DL,NBRFLEX
=3B4E 7305            3B55                JAE     S1
=3B50 B002                                MOV     AL,2
=3B52 E90200          3B57                JMP     S2
=                           S1:
=3B55 B003                                MOV     AL,3
=                           S2:
```

```

=3B57 8ADA                      MOV     BL,DL
=3B59 E814FB       3670         CALL    INITDPB
=3B5C C3                        RET
=                         ;
=                         ;
=                         SSETTRK:
=3B50 B600                      MOV     DH,0
=                         SSETTRK2:
=3B5F 89166747                  MOV     HSTTRK,DX
=3B63 C3                        RET
=                         ;
=                         ;
=                         SSETSEC:
=3B64 88166947                  MOV     HSTSEC,DL
=3B68 8816E747                  MOV     SECTOR,DL
=3B6C C3                        RET
=                         ;
=                         ;
=                         SHOME:
=3B6D A06647                    MOV     AL,HSTDSK
=3B70 A2E447                    MOV     DRV,AL
=3B73 E80502       3D7B         CALL    DREST
=3B76 C3                        RET
=                         ;
=                         ;
=                         SEREAD:
=3B77 C6067E47FF                MOV     DISPFLAG,0FFH
=3B7C E819FD       3898         CALL    SREAD
=3B7F C6067E4700                MOV     DISPFLAG,0
=3B84 C3                        RET
=                         ;
=                         ;
=                         SEWRITE:
=3B85 C6067E47FF                MOV     DISPFLAG,0FFH
=3B8A E86FFD       38FC         CALL    SWRITE
=3B8D C6067E4700                MOV     DISPFLAG,0
=3B92 C3                        RET
=                         ;
=                         ;
=                         SSETDMAB:
=3B93 89167A47                  MOV     PMAADDR,DX
=3B97 C3                        RET
=
=                               ENDIF
=                         ;*************************************************************************
=                         ;                                                                       *
=                         ;                                                                       *
=                         ;   SELTYP  -  RETURNS PARAMETERS FOR THE EXCHANGE UTILITY              *
=                         ;                                                                       *
=                         ;                                                                       *
=                         ;   ENTRY VIA JMP                                                       *
=                         ;                                                                       *
=                         ;                                                                       *
=                         ;   EXIT VIA RETURN                                                     *
=                         ;                                                                       *
=                         ;                                                                       *
=                         ;*************************************************************************
=                         ;
=                         ;
=                         SELTYP:
=3B98 B80400                    MOV     AX,LENGTH DSKBLM        ;WIDTH OF TYPE DEFINITION TABLE
=3B9B BB2000                    MOV     BX,VERLEN               ;VERSION NUMBER AND XLT LENGTH
=3B9E BA4247                    MOV     DX,OFFSET XLT           ;ADDRESS OF XLT TABLE
=3BA1 B9D646                    MOV     CX,OFFSET DSKSID        ;ADDRESS OF TYPE DEFINITION TABLE
=3BA4 C3                        RET

```

```


 =                      INCLUDE C:FLEXPIMC.SEG
 =                ;     TITLE   FLEX DISK DRIVER PIM (CODE SEGMENT)
 =                ;
 =                ;
 =                ;  ROUTINE NAME:       DREAD
 =                ;                      DWRITE
 =                ;
 =                ;
 =                ;
 =                ;
 =                ;
 =                ;  FUNCTION:           DREAD - low level READ DATA
 =                ;                      DWRITE - low level WRITE DATA
 =                ;
 =                ;
 =                ;
 =                ;

 =                ;  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)
 =                ;
 =                ;
 =                ;
 =                ;**********************************************************************
 =                ;**********************************************************************
 =                ;**********************************************************************
 =                ;
 =                ;
 =                ;
 =
=                      DREAD:                              ;
=3BA5 B106                     MOV    CL,READDAT           ; CL <-- READ DATA COMMAND
=3BA7 C606014847               MOV    DMAFUNC,DMAWRT       ; DMAFUNC <-- WRITE DMA COMMAND
=3BAC E90700       3BB6        JMP    I01                  ;
=                      DWRITE:                             ;
=3BAF B105                     MOV    CL,WRITDAT           ; CL <-- WRITE DATA COMMAND
=3BB1 C60601484B               MOV    DMAFUNC,DMAREAD      ; DMAFUNC <-- READ DMA COMMAND
=                      I01:                                ;
=3BB6 833EE84700               CMP    SECCNT,0             ; Check if an I/O is necessary
=3BBB 7501         3BBE        JNZ    I02                  ; Jump if necessary
=3BBD C3                       RET                         ; Return if not necessary
=                      I02:                                ;
=                                                          ; Check TRACK conflict
=3BBE B700                     MOV    BH,00                ; --------------------
=3BC0 8A1E0248                 MOV    BL,SECTRK            ; BX <-- SECTORS PER TRACK
=3BC4 FEC3                     INC    BL                   ;
=3BC6 2A1EE747                 SUB    BL,SECTOR            ; BX - remainding sectors in track
=                                                          ;
=3BCA A0E347                   MOV    AL,CYLMODE           ; If CYLINDER MODE
=3BCD 0A06E547                 OR     AL,HEAD              ; and HEAD 0
=3BD1 7504         3BD7        JNZ    I03                  ;
=3BD3 021E0248                 ADD    BL,SECTRK            ; then add sectors of corresponding track
=                      I03:                                ;
=3BD7 3B1EE847                 CMP    BX,SECCNT            ; Compare remainding sectors with SECCNT
=3BDB 7204         3BE1        JB     I04                  ; Jump if more than one I/O
=3BDD 8B1EE847                 MOV    BX,SECCNT            ;
=                      I04:                                ; BX - number of sectors fitting in TRACK
=                                                          ;
=                                                          ; Check BANK conflict
=                                                          ; -------------------
=3BE1 A1FD47                         MOV     AX,DMAADDR+2  ; AX <-- DMA SEGMENT
=3BE4 D1E0                           SHL     AX,1          ;
=3BE6 D1E0                           SHL     AX,1          ;
=3BE8 D1E0                           SHL     AX,1          ;
=3BEA D1E0                           SHL     AX,1          ;
=3BEC 0306FB47                       ADD     AX,DMAADDR    ; AX <-- absolute addr within BANK
=3BF0 F7D8                           NEG     AX            ; AX <-- remainding bytes within BANK
=3BF2 8A360448                       MOV     DH,BYTSEC     ;
=3BF6 B200                           MOV     DL,00         ; DX <-- sector size
=3BF8 80FE00                         CMP     DH,00         ;
=3BFB 7502              3BFF         JNZ     I05           ;
=                                                          ;
=3BFD B280                           MOV     DL,128        ;
=                            I05:                          ;
=3BFF 8BF2                           MOV     SI,DX         ; SI <-- sector size
=3C01 BA0000                         MOV     DX,0000       ; DX <-- 0000
=3C04 F7F6                           DIV     SI            ; AX <-- number of sectors fitting in BANK
=                                                          ;
=3C06 3BC3                           CMP     AX,BX         ; Check if we must do Special Sector Handling
=3C08 7203              3C0D         JB      I06           ; Jump if we must
=                                                          ;
=3C0A E98600            3C93         JMP     I015          ; Jump around if not
=                            I06:                          ;
=3C0D 93                             XCHG    BX,AX         ; BX <-- number of sectors fitting in BANK
=3C0E 83FB00                         CMP     BX,00         ; Check if we must do now Special Sector Handling
=3C11 7403              3C16         JZ      I07           ; Jump if we must     ---
=                                                          ;
=3C13 E97D00            3C93         JMP     I015          ; Jump around if not
=
=
=
=                            I07:                          ;** Special Sector Handling
=                                                          ;** ----------------------
=3C16 832EE84701                     SUB     SECCNT,01     ;** SECCNT <-- remainding sectors for next I/O
=                                                          ;**
=3C1B 8A260448                       MOV     AH,BYTSEC     ;**
=3C1F B000                           MOV     AL,00         ;** AX <-- sector size
=3C21 80FC00                         CMP     AH,00         ;**
=3C24 7502              3C28         JNZ     I08           ;**
=                                                          ;**
=3C26 B080                           MOV     AL,128        ;**
=                            I08:                          ;**
=3C28 A3FF47                         MOV     DMALENG,AX    ;** DMALENG <-- sector size
=                                                          ;**
=3C2B 80E10F                         AND     CL,0FH        ;** Clear upper bits
=3C2E 80F905                         CMP     CL,WRITDAT    ;** Check if WRITE DATA COMMAND
=3C31 751B              3C4E         JNZ     I09           ;** Jump around if not
=                                                          ;*
=                                                          ;*
=                                                          ;*
```

```

=3C33 51                        PUSH   CX             ;* Save CX
=3C34 8B36FB47                  MOV    SI,DMAADDR     ;* SI <-- source offset
=3C38 BF0848                    MOV    DI,OFFSET SSB  ;* DI <-- destination offset
=3C3B 8B0EFF47                  MOV    CX,DMALENG     ;* CX <-- sector size
=3C3F D1E9                      SHR    CX,1           ;* We move WORDS
=3C41 FC                        CLD                   ;* incrementing
=3C42 1E                        PUSH   DS             ;* Save DS
=3C43 A1FD47                    MOV    AX,DMAADDR+2   ;*
=3C46 8ED8                      MOV    DS,AX          ;* DS <-- SEGMENT of TRANSFER ADDR
=3C48 07                        POP    ES             ;*
=3C49 06                        PUSH   ES             ;* ES <-- our SEGMENT of Special Sector Buffer
=                                                     ;*
=                                                     ;* W R I T E  D A T A  C O M M A N D:
=3C4A F3A5                REP   MOVSW                 ;* Move BANK into Special Sector Buffer
=                                                     ;* -----------------------------------
=3C4C 1F                        POP    DS             ;* Restore DS
=3C4D 59                        POP    CX             ;* Restore CX
=                                                     ;*
=                                                     ;*
=                                                     ;*
=                         I09:                        ;**
=3C4E A1FB47                    MOV    AX,DMAADDR     ;**
=3C51 50                        PUSH   AX             ;** Save DMA OFFSET
=3C52 A1FD47                    MOV    AX,DMAADDR+2   ;**
=3C55 50                        PUSH   AX             ;** Save DMA SEGMENT
=                                                     ;**
=3C56 B80848                    MOV    AX,OFFSET SSB  ;**
=3C59 A3FB47                    MOV    DMAADDR,AX     ;** new OFFSET <-- Special Sector Buffer
=3C5C 8CD8                      MOV    AX,DS          ;**
=3C5E A3FD47                    MOV    DMAADDR+2,AX   ;** new SEGMENT <-- our SEGMENT
=                                                     ;**
=3C61 E85100        3C85        CALL   IO             ;** Do I/O
=                                                     ;** ------
=3C64 7203          3C69        JC     IO10           ;** Jump if normal termination
=3C66 58                        POP    AX             ;** else
=3C67 58                        POP    AX             ;** flush STACK
=3C68 C3                        RET                   ;** and return with bad status in ERRBUF
=                         IO10:                       ;**
=3C69 58                        POP    AX             ;**
=3C6A A3FD47                    MOV    DMAADDR+2,AX   ;** Restore DMA SEGMENT
=3C6D 8EC0                      MOV    ES,AX          ;**
=3C6F 58                        POP    AX             ;**
=3C70 A3FB47                    MOV    DMAADDR,AX     ;** Restore DMA OFFSET
=                                                     ;**
=                                                     ;**
=                                                     ;**
=3C73 80E10F                    AND    CL,0FH         ;** Clear upper bits
=3C76 80F906                    CMP    CL,READDAT     ;** Check if READ DATA COMMAND
=3C79 7512          3C8D        JNZ    IO11           ;** Jump around if not
=                                                     ;*
=3C7B 51                        PUSH   CX             ;* Save CX
=3C7C BE0848                    MOV    SI,OFFSET SSB  ;* SI <-- source offset
```

```

=3C7F 8B3EF847                MOV     DI,DMAADOR      ;* DI <-- destination offset
=3C83 8B0EFF47                MOV     CX,DMALENG      ;* CX <-- sector size
=3C87 D1E9                    SHR     CX,1            ;* We move WORDS
=3C89 FC                      CLD                     ;* incrementing
=                                                     ;* R E A D  0 A T A  C 0 M M A N D:
=3C8A F3A5              REP   MOVSW                   ;* Move Special Sector Buffer into BANK
=                                                     ;* ----------------------------------
=3C8C 59                      POP     CX              ;* Restore CX
=                                                     ;*
=                                                     ;*
=                       I011:                         ;**
=3C8D BB0100                  MOV     BX,0001         ;** BX - number of sectors of previous I/O
=3C90 E96200      3CF5        JMP     I030            ;** Jump to update variables for next I/O
=
=
=
=                       I015:                         ; BX - number of sectors for I/O
=3C93 53                      PUSH    BX              ; ------------------------------
=3C94 291EE847                SUB     SECCNT,BX       ; SECCNT <-- remainding sectors for next I/O
=                                                     ;
=3C98 8A260448                MOV     AH,BYTSEC       ;
=3C9C B000                    MOV     AL,00           ; AX <-- sector size
=3C9E 80FC00                  CMP     AH,00           ;
=3CA1 7502        3CA5        JNZ     I016            ;
=                                                     ;
=3CA3 B080                    MOV     AL,128          ;
=                       I016:                         ;
=3CA5 F7E3                    MUL     BX              ; * sectors for I/O gives DMA LENGTH
=3CA7 A3FF47                  MOV     DMALENG,AX      ; DMALENG <-- DMA LENGTH
=                                                     ;
=3CAA E80800      3CB5        CALL    IO              ; Do I/O
=                                                     ; ------
=3CAD 7202        3CB1        JC      I017            ; Jump if normal termination
=3CAF 58                      POP     AX              ; else flush STACK
=3CB0 C3                      RET                     ; and return with bad status in ERRBUF
=                       I017:                         ;
=3CB1 5B                      POP     BX              ; BX - number of sectors of previous I/O
=3CB2 E94000      3CF5        JMP     I030            ; Jump to update variables for next I/O
=
=
=
=                       IO:                           ; Disk I/O
=                                                     ; --------
=3CB5 A00748                  MOV     AL,RETRIES      ; AL <--  retry counter
=                       I020:                         ;
=3CB8 50                      PUSH    AX              ; Save retry counter
=3CB9 E86901      3E25        CALL    SETUP9          ; Set up COMMAND STRING and DMA
=3CBC E8F401      3EB3        CALL    XWAIT           ; Send COMMAND STRING to FDC
=3CBF E80F02      3ED1        CALL    GETBYT          ; Get STATUS BYTES
=3CC2 58                      POP     AX              ; Restore retry counter
=                                                     ;
=3CC3 F606F447C0              TEST`   ERRBUF,0C0H     ; Test for normal termination
```

```

=3CC8 7502         3CCC            JNZ     I021            ; Jump on error
=3CCA F9                           STC                     ; Set status flag
=3CCB C3                           RET                     ; Return with good status
=                                                          ;
=                                                          ;
=                                                          ;
=                          I021:                           ;
=3CCC F606F44708                   TEST    ERRBUF,08H      ; Test for 'NOT READY'
=3CD1 7402         3CD5            JZ      I022            ;
=3CD3 F8                           CLC                     ; Set status flag
=3CD4 C3                           RET                     ; Return immediately if disk 'NOT READY'
=                          I022:                           ;
=3CD5 F606F54702                   TEST    ERRBUF+1,02H    ; Test for 'WRITE PROTECTED'
=3CDA 7402         3CDE            JZ      I023            ;
=3CDC F8                           CLC                     ; Set status flag
=3CDD C3                           RET                     ; Return immediately if 'WRITE PROTECTED'
=                          I023:                           ;
=3CDE F606F44780                   TEST    ERRBUF,80H      ; Test for 'INVALID COMMAND'
=3CE3 7402         3CE7            JZ      I024            ;
=3CE5 F8                           CLC                     ; Set status flag
=3CE6 C3                           RET                     ; Return immediately if 'INVALID COMMAND'
=                          I024:                           ;
=3CE7 FEC8                         DEC     AL              ; Decrement retry counter
=3CE9 7408         3CF3            JZ      I025            ; Jump to exit with bad status
=                                                          ;
=3CEB 50                           PUSH    AX              ; Save retry counter
=3CEC E88C00       3D7B            CALL    DREST           ; Do a low level RESTORE
=3CEF 58                           POP     AX              ; Restore retry counter
=3CF0 E9C5FF       3CB8            JMP     I020            ; Do retries
=                                                          ;
=                                                          ;
=                                                          ;
=                          I025:                           ;
=3CF3 F8                           CLC                     ; Set status flag
=3CF4 C3                           RET                     ; Return with bad status
=
=
=                          I030:                           ; Update variables for next I/O
=                                                          ; ----------------------------
=                                                          ; BX - number of sectors of previous I/O
=                                                          ;
=3CF5 833EE84700                   CMP     SECCNT,0        ; Check if another I/O is necessary
=3CFA 7501         3CFD            JNZ     I031            ; Jump if necessary
=3CFC C3                           RET                     ; Return if not necessary
=                          I031:                           ;
=3CFD 8B16FF47                     MOV     DX,DMALENG      ; DX <-- previous DMA LENGTH
=3D01 D1EA                         SHR     DX,1            ;
=3D03 D1EA                         SHR     DX,1            ;
=3D05 D1EA                         SHR     DX,1            ;
=3D07 D1EA                         SHR     DX,1            ; DX - previous DMA LENGTH in paragraphs
=3D09 0116FD47                     ADD     WORD PTR DMAADDR+2,DX ; Update DMAADDR (SEGMENT)
=                                                          ;
```

```

=3D0D 001EE747               ADD    SECTOR,BL        ; Update SECTOR variable
=3D11 A00248                 MOV    AL,SECTRK        ; AL <-- sectors per track
=                                                    ;
=3D14 803EE34700             CMP    CYLMODE,00       ; Check if CYLINDER MODE
=3D19 7429       3D44        JZ     I034             ; Jump if CYLINDER MODE
=                                                    ;
=                                                    ; Not CYLINDER MODE
=                                                    ; ---------------
=3D1B 3A06E747               CMP    AL,SECTOR        ; Check for legal SECTOR variable
=3D1F 7203       3D24        JB     I032             ; Jump if not legal
=                                                    ;
=3D21 E992FE     3B86        JMP    I01              ; Do next I/O
=                      I032:                         ;
=3D24 C606E74701             MOV    SECTOR,1         ; Set SECTOR to begin of track
=3D29 803EE64727             CMP    TRACK,39         ; Check if side 1 is full
=3D2E 7407       3D37        JZ     I033             ; Jump if full
=                                                    ;
=3D30 FE06E647               INC    TRACK            ; Increment TRACK
=3D34 E97FFE     3B86        JMP    I01              ; Do next I/O
=                      I033:                         ;
=3D37 C606E54701             MOV    HEAD,1           ; If side 1 is full
=3D3C C606E64700             MOV    TRACK,0          ; then initialize for side 2
=3D41 E972FE     3B86        JMP    I01              ; Do next I/O
=                      I034:                         ;
=                                                    ; CYLINDER MODE
=                                                    ; ------------
=3D44 3A06E747               CMP    AL,SECTOR        ; Check for legal SECTOR variable
=3D48 7203       3D4D        JB     I035             ; Jump if not legal
=                                                    ;
=3D4A E969FE     3B86        JMP    I01              ; Do next I/O
=                      I035:                         ;
=3D4D 803EE54701             CMP    HEAD,1           ; Check if cylinder is full
=3D52 7416       3D6A        JZ     I036             ; Jump if full
=                                                    ;
=3D54 D0E0                   SHL    AL,1             ; AL <-- sectors per cylinder
=3D56 3A06E747               CMP    AL,SECTOR        ; Check if cylinder is full
=3D5A 720E       3D6A        JB     I036             ; Jump if full
=                                                    ;
=3D5C D0E8                   SHR    AL,1             ; AL <-- sectors per track
=3D5E 2806E747               SUB    SECTOR,AL        ; Set SECTOR variable within
=3D62 C606E54701             MOV    HEAD,1           ; corresponding track with HEAD 1
=3D67 E94CFE     3B86        JMP    I01              ; Do next I/O
=                      I036:                         ;
=3D6A FE06E647               INC    TRACK            ; Increment TRACK
=3D6E C606E54700             MOV    HEAD,0           ; Set HEAD 0
=3D73 C606E74701             MOV    SECTOR,1         ; Set SECTOR to begin of cylinder
=3D78 E93BFE     3B86        JMP    I01              ; Do next I/O
=
=
=                     ;***************************************************************************
=                     ;***************************************************************************
=                     ;***************************************************************************

=                     ;    ROUTINE NAME:       DREST
=                     ;
=                     ;
=                     ;
=                     ;
=                     ;
=                     ;    FUNCTION:           Low level RESTORE
=                     ;
=                     ;
=                     ;
=                     ;
=                     ;    ENTRY VIA:          CALL
=                     ;
=                     ;
=                     ;    ENTRY CONDITIONS:   DRV variable is set
=                     ;
=                     ;
=                     ;
=                     ;
=                     ;    EXIT VIA:           RETURN
=                     ;
=                     ;
=                     ;    EXIT CONDITIONS:    CL - preserved
=                     ;                        STATUS (returned in ERRBUF)
=                     ;
=                     ;
=                     ;
=                     ;***************************************************************************
=                     ;***************************************************************************
=                     ;***************************************************************************
=                     ;
=                     ;
=                     ;
=
=                     DREST:                          ;
=3D7B B402                     MOV    AH,02           ; Special retry for CP/M
=                                                     ;
=                     DREST1:                         ; Set up COMMAND STRING
=                                                     ; --------------------
=3D7D C606EA4702               MOV    COMSTR,2        ; COMMAND STRING <-- LENGTH 2
=3D82 C606EB4707               MOV    COMSTR+1,RESTORE;                <-- RESTORE COMMAND
=3D87 A0E447                   MOV    AL,DRV          ;
=3D8A A2EC47                   MOV    COMSTR+2,AL     ;                <-- DRIVE NUMBER
=                                                     ;
=3D8D 50                       PUSH   AX              ; Save retry counter


=3D8E E82201        3EB3       CALL   XWAIT           ; Send COMMAND STRING to FDC
=                     DREST2:                         ;
=3D91 E413                     IN     AL,SYSSTA       ; Wait on interrupt
=3D93 2408                     AND    AL,08           ; Test DISK INTERRUPT BIT
=3D95 74FA          3D91       JZ     DREST2          ; Jump if no interrupt
=                                                     ;
=3D97 E85B00        3DF5       CALL   DSIS            ; Reset interrupt via low level SENSE
=                                                     ; INTERRUPT STATUS
=                                                     ;
=3D9A 58                       POP    AX              ; Restore retry counter
=3D9B F606F447C0               TEST   ERRBUF,0C0H     ; Test for normal termination
=3DA0 7404          3DA6       JZ     DREST3          ; Jump if normal termination
=                                                     ;
=3DA2 FECC                     DEC    AH              ; Decrement retry counter
=3DA4 7507          3D7D       JNZ    DREST1          ; Do special retry !
=                     DREST3:                         ; Reason: MOTOR OFF & RESTORE in CP/M
=3DA6 C3                       RET                    ;
=
3951  =                             ;**********************************************************************
3952  =                             ;**********************************************************************
3953  =                             ;**********************************************************************
3954  =                             ;
3955  =                             ;
3956  =                             ;
3957  =                             ;
3958  =                             ;
3959  =                             ;
3960  =                             ;
3961  =                             ;   ROUTINE NAME:       DSEEK
3962  =                             ;
3963  =                             ;
3964  =                             ;
3965  =                             ;
3966  =                             ;
3967  =                             ;   FUNCTION:           Low level SEEK A TRACK
3968  =                             ;
3969  =                             ;
3970  =                             ;
3971  =                             ;
3972  =                             ;   ENTRY VIA:          CALL
3973  =                             ;
3974  =                             ;
3975  =                             ;   ENTRY CONDITIONS:   Following variables are set:
3976  =                             ;                       DRV, HEAD, and TRACK
3977  =                             ;
3978  =                             ;
3979  =                             ;
3980  =                             ;
3981  =                             ;   EXIT VIA:           RETURN
3982  =                             ;
3983  =                             ;
3984  =                             ;   EXIT CONDITIONS:    CL - preserved
3985
3986  =                             ;                       STATUS (returned in ERRBUF)
3987  =                             ;
3988  =                             ;
3989  =                             ;
3990  =                             ;**********************************************************************
3991  =                             ;**********************************************************************
3992  =                             ;**********************************************************************
3993  =                             ;
3994  =                             ;
3995  =                             ;
3996  =
3997  =                             DSEEK:                          ; Set up COMMAND STRING
3998  =                                                             ; --------------------
3999  =3DA7 C606EA4703                      MOV     COMSTR,3        ; COMMAND STRING <-- LENGTH 3
4000  =3DAC C606EB470F                      MOV     COMSTR+1,SEEKTRK;                <-- SEEK COMMAND
4001  =3DB1 A0E547                          MOV     AL,HEAD         ;
4002  =3DB4 D0E0                            SHL     AL,1            ;
4003  =3DB6 D0E0                            SHL     AL,1            ;
4004  =3DB8 0A06E447                        OR      AL,DRV          ;
4005  =3DBC A2EC47                          MOV     COMSTR+2,AL     ;                <-- DRIVE & HEAD
4006  =3DBF A0E647                          MOV     AL,TRACK        ;
4007  =3DC2 A2ED47                          MOV     COMSTR+3,AL     ;                <-- TRACK
4008  =                                                             ;
4009  =3DC5 E8EB00          3EB3            CALL    XWAIT           ; Send COMMAND STRING to FDC
4010  =                             DSEEK1:                         ;
4011  =3DC8 E413                            IN      AL,SYSSTA       ; Wait on interrupt
4012  =3DCA 2408                            AND     AL,08           ; Test DISK INTERRUPT BIT
4013  =3DCC 74FA            3DC8            JZ      DSEEK1          ; jump if no interrupt
4014  =                                                             ;
4015  =3DCE E82400          3DF5            CALL    DSIS            ; Reset interrupt via low level SENSE
4016  =                                                             ; INTERRUPT STATUS
4017  =3DD1 C3                              RET                     ;
4018  =
=                       ;**********************************************************************
=                       ;**********************************************************************
=                       ;**********************************************************************
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;    ROUTINE NAME:       DREADID
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;    FUNCTION:           Low level READ ID
=                       ;                        (Used to get SECTOR SIZE)
=                       ;

=                       ;
=                       ;
=                       ;
=                       ;    ENTRY VIA:          CALL
=                       ;
=                       ;
=                       ;    ENTRY CONDITIONS:   Following variables are set:
=                       ;                        DRV and HEAD
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;    EXIT VIA:           RETURN
=                       ;
=                       ;
=                       ;    EXIT CONDITIONS:    STATUS and BYTES PER SECTOR (returned in ERRBUF)
=                       ;
=                       ;
=                       ;
=                       ;**********************************************************************
=                       ;**********************************************************************
=                       ;**********************************************************************
=                       ;
=                       ;
=                       ;
=
=                       DREADID:                        ; Set up COMMAND STRING
=                                                       ; -------------------
=3D02 C606EA4702                MOV     COMSTR,2        ; COMMAND STRING <-- LENGTH 2
=3D07 B00A                      MOV     AL,IDREAD       ;
=3D09 0A060348                  OR      AL,DENSITY      ;
=3DDD A2EB47                    MOV     COMSTR+1,AL     ;                <-- READ ID COMMAND & DENSITY
=3DE0 A0E547                    MOV     AL,HEAD         ;
=3DE3 D0E0                      SHL     AL,1            ;
=3DE5 D0E0                      SHL     AL,1            ;
=3DE7 0A06E447                  OR      AL,DRV          ;
=3DEB A2EC47                    MOV     COMSTR+2,AL     ;                <-- DRIVE & HEAD
=                                                       ;
=3DEE E8C200       3EB3         CALL    XWAIT           ; Send COMMAND STRING to FCB
=3DF1 E8DD00       3ED1         CALL    GETBYT          ; Get STATUS BYTES (sector size)
=3DF4 C3                        RET                     ;
=
=                      ;**********************************************************************
=                      ;**********************************************************************
=                      ;**********************************************************************
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;

=                      ;   ROUTINE NAME:      DSIS
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;   FUNCTION:          Low level SENSE INTERRUPT STATUS
=                      ;                      (used to reset interrupt)
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;   ENTRY VIA:         CALL
=                      ;
=                      ;
=                      ;   ENTRY CONDITIONS:  NONE
=                      ;
=                      ;
=                      ;
=                      ;
=                      ;   EXIT VIA:          RETURN
=                      ;
=                      ;
=                      ;   EXIT CONDITIONS:   STATUS (returned in ERRBUF)
=                      ;
=                      ;
=                      ;
=                      ;**********************************************************************
=                      ;**********************************************************************
=                      ;**********************************************************************
=                      ;
=                      ;
=                      ;
=
=                      DSIS:                         ; Set up COMMAND STRING
=                                                    ; ------------------
=3DF5 C606EA4701              MOV     COMSTR,1       ; COMMAND STRING (-- LENGTH 1
=3DFA C606EB4708              MOV     COMSTR+1,FDCSIS ;                (-- FDCSIS COMMAND
=                                                    ;
=3DFF E88100       3EB3       CALL    XWAIT          ; Send COMMAND STRING to FDC
=3E02 E8CC00       3ED1       CALL    GETBYT         ; Get STATUS BYTES
=3E05 C3                      RET                    ;
=
=                         ;*****************************************************************************
=                         ;*****************************************************************************
=                         ;*****************************************************************************
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;

=                         ;
=                         ;    ROUTINE NAME:      DFORMAT
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;    FUNCTION:          Low level FORMAT A TRACK
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;    ENTRY VIA:         CALL
=                         ;
=                         ;
=                         ;    ENTRY CONDITIONS:  Following variables are set:
=                         ;                       DRV, HEAD, TRACK, PATTERN
=                         ;                       and DMAADDR (SEGMENT and OFFSET)
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;    EXIT VIA:          RETURN
=                         ;
=                         ;
=                         ;    EXIT CONDITIONS:   STATUS (returned in ERRBUF)
=                         ;
=                         ;
=                         ;
=                         ;*****************************************************************************
=                         ;*****************************************************************************
=                         ;*****************************************************************************
=                         ;
=                         ;
=                         ;
=
=                         DFORMAT:                        ;
=3E06 B100                        MOV     CL,WRITFMT      ; CL (-- FORMAT COMMAND
=3E08 C606014848                  MOV     DMAFUNC,DMAREAD ; DMAFUNC (-- READ DMA COMMAND
=3E0D B700                        MOV     BH,00           ;
=3E0F 8A1E0248                    MOV     BL,SECTRK       ;
=3E13 D1E3                        SHL     BX,1            ;
=3E15 D1E3                        SHL     BX,1            ;
=3E17 891EFF47                    MOV     DMALENG,BX      ; DMALENG (-- DMA LENGTH (SECTRK*4)
=                                                         ;
=3E18 E85C00       3E7A           CALL    SETUP6          ; Set up COMMAND STRING and DMA
=3E1E E89200       3EB3           CALL    XWAIT           ; Send COMMAND STRING to FDC
=3E21 E8AD00       3ED1           CALL    GETBYT          ; Get STATUS BYTES
=3E24 C3                          RET                     ;
=
=            ;****************************************************************************
=            ;****************************************************************************

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

```
 =                        ;
 =                        ;
 =
 =                        SETUP9:                             ;
 =3E25 E87FFF      3DA7            CALL     DSEEK             ; First do low level SEEK A TRACK
 =                                                            ;
 =3E28 C606EA4709                  MOV      COMSTR,9          ; COMMAND STRING <-- LENGTH 9
 =3E2D 0A0E0348                    OR       CL,DENSITY        ;
 =3E31 803EE34700                  CMP      CYLMODE,00        ;
 =3E36 7503        3E3B            JNZ      SET1              ;
 =                                                            ;

 =3E38 80C980                      OR       CL,80H            ;
 =                        SET1:                               ;
 =3E3B 880EEB47                    MOV      COMSTR+1,CL       ;             <-- FUNCTION & DENSITY & MT
 =3E3F A0E547                      MOV      AL,HEAD           ;
 =3E42 D0E0                        SHL      AL,1              ;
 =3E44 D0E0                        SHL      AL,1              ;
 =3E46 0A06E447                    OR       AL,DRV            ;
 =3E4A A2EC47                      MOV      COMSTR+2,AL       ;             <-- DRIVE & HEAD
 =3E4D A0E647                      MOV      AL,TRACK          ;
 =3E50 A2ED47                      MOV      COMSTR+3,AL       ;             <-- TRACK
 =3E53 A0E547                      MOV      AL,HEAD           ;
 =3E56 A2EE47                      MOV      COMSTR+4,AL       ;             <-- HEAD
 =3E59 A0E747                      MOV      AL,SECTOR         ;
 =3E5C A2EF47                      MOV      COMSTR+5,AL       ;             <-- SECTOR
 =3E5F A00448                      MOV      AL,BYTSEC         ;
 =3E62 A2F047                      MOV      COMSTR+6,AL       ;             <-- BYTES PER SECTOR
 =3E65 A00248                      MOV      AL,SECTRK         ;
 =3E68 A2F147                      MOV      COMSTR+7,AL       ;             <-- SECTORS PER TRACK
 =3E6B A00548                      MOV      AL,GPL            ;
 =3E6E A2F247                      MOV      COMSTR+8,AL       ;             <-- GAP LENGTH
 =3E71 C606F347FF                  MOV      COMSTR+9,0FFH     ;             <-- DTL
 =                                                            ;
 =3E76 E88100      3EFA            CALL     DMA               ; Initialize DMA
 =3E79 C3                          RET                        ;
 =
;************************************************************************
;************************************************************************
;************************************************************************
;
;
;
;
;
;
;
;    ROUTINE NAME:       SETUP6
;
;
;
;
;
;    FUNCTION:           Set up (6 byte) COMMAND STRING and DMA
;
;
;
;
;    ENTRY VIA:          CALL
;
;
;    ENTRY CONDITIONS:   CL - (FORMAT) COMMAND
;                        Following variables are set:
;                        DMAADDR (SEGMENT and OFFSET)

;                        DMALENG and DMAFUNC
;
;
;
;
;    EXIT VIA:           RETURN
;
;
;    EXIT CONDITIONS:    NONE
;
;
;
;************************************************************************
;************************************************************************
;************************************************************************
;
;
;
```

```
4322  =
4323  =                        SETUP6:                          ;
4324  =3E7A E82AFF       3DA7         CALL    DSEEK            ; First do low level SEEK A TRACK
4325  =                                                         ;
4326  =3E7D C606EA4706                MOV     COMSTR,6         ; COMMAND STRING (-- LENGTH 6
4327  =3E82 0A0E0348                  OR      CL,DENSITY       ;
4328  =3E86 880EEB47                  MOV     COMSTR+1,CL      ;                (-- FUNCTION & DENSITY
4329  =3E8A A0E547                    MOV     AL,HEAD          ;
4330  =3E8D D0E0                      SHL     AL,1             ;
4331  =3E8F D0E0                      SHL     AL,1             ;
4332  =3E91 0A06E447                  OR      AL,DRV           ;
4333  =3E95 A2EC47                    MOV     COMSTR+2,AL      ;                (-- DRIVE & HEAD
4334  =3E98 A00448                    MOV     AL,BYTSEC        ;
4335  =3E9B A2ED47                    MOV     COMSTR+3,AL      ;                (-- BYTES PER SECTOR
4336  =3E9E A00248                    MOV     AL,SECTRK        ;
4337  =3EA1 A2EE47                    MOV     COMSTR+4,AL      ;                (-- SECTORS PER TRACK
4338  =3EA4 C606EF4750                MOV     COMSTR+5,50H     ;                (-- GAP LENGTH
4339  =3EA9 A00648                    MOV     AL,PATTERN       ;
4340  =3EAC A2F047                    MOV     COMSTR+6,AL      ;                (-- PATTERN
4341  =                                                         ;
4342  =3EAF E84800       3EFA         CALL    DMA              ; Initialize DMA
4343  =3EB2 C3                        RET                      ;
4344  =
=                           ;***********************************************************************
=                           ;***********************************************************************
=                           ;***********************************************************************
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;    ROUTINE NAME:      XWAIT

=                           ;
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;    FUNCTION:          Send COMMAND STRING to FDC
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;    ENTRY VIA:         CALL
=                           ;
=                           ;
=                           ;    ENTRY CONDITIONS:  NONE
=                           ;
=                           ;
=                           ;
=                           ;
=                           ;    EXIT VIA:          RETURN
=                           ;
=                           ;
=                           ;    EXIT CONDITIONS:   CL - preserved
=                           ;
=                           ;
=                           ;
=                           ;***********************************************************************
=                           ;***********************************************************************
=                           ;***********************************************************************
=                           ;
=                           ;
=                           ;
=
=                           XWAIT:                                  ;
=3EB3 E83200        3EE8            CALL    MOTORCK                 ; SWITCH MOTOR ON
=                                                                   ;
=3EB6 8A2EEA47                      MOV     CH,COMSTR               ; CH <-- COMMAND STRING LENGTH
=3EBA BBEA47                        MOV     BX,OFFSET COMSTR; BX <-- Addr of COMMAND STRING
=                           XWAIT1:                                 ;
=3EBD 43                            INC     BX                      ;
=3EBE E82000        3EE1            CALL    FDCRDY                  ; Wait until FDC is ready
=3EC1 8A07                          MOV     AL,BYTE PTR [BX]; AL <-- next COMMAND STRING byte
=3EC3 E651                          OUT     DCOMD,AL                ; Send byte to FDC
=3EC5 FECD                          DEC     CH                      ; Decrement counter
=3EC7 75F4          3EBD            JNZ     XWAIT1                  ; Loop until last byte
=                                                                   ;
=3EC9 E81500        3EE1            CALL    FDCRDY                  ; Wait until FDC is ready
=                                                                   ;
=3ECC B007                          MOV     AL,07                   ;
=3ECE E62A                          OUT     DMAMB,AL                ; Disable DMA CHANNEL
=3ED0 C3                            RET                             ;
=
=

=                              ;*********************************************************************
=                              ;*********************************************************************
=                              ;*********************************************************************
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;    ROUTINE NAME:      GETBYT
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;    FUNCTION:          Get STATUS BYTES into ERRBUF
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;    ENTRY VIA:         CALL
=                              ;
=                              ;
=                              ;    ENTRY CONDITIONS:  NONE
=                              ;
=                              ;
=                              ;
=                              ;
=                              ;    EXIT VIA:          RETURN
=                              ;
=                              ;
=                              ;    EXIT CONDITIONS:   NONE
=                              ;
=                              ;
=                              ;
=                              ;*********************************************************************
=                              ;*********************************************************************
=                              ;*********************************************************************
=                              ;
=                              ;
=                              ;
=
=                              GETBYT:                           ;
=3ED1 BBF447                           MOV     BX,OFFSET ERRBUF; BX <-- Addr of ERROR BUFFER
=                              GETBYT1:                          ;
=3ED4 E451                             IN      AL,FDCRA        ; Read STATUS BYTE from FDC
=3ED6 8807                             MOV     BYTE PTR [BX],AL; into ERROR BUFFER
=3ED8 43                               INC     BX              ;
=3ED9 E80500         3EE1              CALL    FDCRDY          ; Wait until FDC is ready
=3EDC A840                             TEST    AL,40H          ; Check if FDC has another byte
=3EDE 75F4           3ED4              JNZ     GETBYT1         ; Jump to fetch next byte

=3EE0 C3                               RET                     ;
=
=                       ;*************************************************************************
=                       ;*************************************************************************
=                       ;*************************************************************************
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;   ROUTINE NAME:      FDCRDY
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;   FUNCTION:          Wait until FDC is ready
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;   ENTRY VIA:         CALL
=                       ;
=                       ;
=                       ;   ENTRY CONDITIONS:  NONE
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;   EXIT VIA:          RETURN
=                       ;
=                       ;
=                       ;   EXIT CONDITIONS:   NONE
=                       ;
=                       ;
=                       ;
=                       ;*************************************************************************
=                       ;*************************************************************************
=                       ;*************************************************************************
=                       ;
=                       ;
=                       ;
=
=                       FDCRDY:                         ;
=3EE1 E450                      IN      AL,DSTAT        ; AL <-- DISK STATUS
=3EE3 A880                      TEST    AL,80H          ; Test MASTER REQUEST BIT
=3EE5 74FA         3EE1         JZ      FDCRDY          ; Jump if no MASTER REQUEST (means: in execution)
=                                                       ;
=3EE7 C3                        RET                     ; Return if FDC is ready
=

=                            ;**********************************************************************
=                            ;**********************************************************************
=                            ;**********************************************************************
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;    ROUTINE NAME:       MOTORCK
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;    FUNCTION:           Check if motor is on
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;    ENTRY VIA:          CALL
=                            ;
=                            ;
=                            ;    ENTRY CONDITIONS:   NONE
=                            ;
=                            ;
=                            ;
=                            ;
=                            ;    EXIT VIA:           RETURN
=                            ;
=                            ;
=                            ;    EXIT CONDITIONS:    Motor is on
=                            ;
=                            ;
=                            ;
=                            ;**********************************************************************
=                            ;**********************************************************************
=                            ;**********************************************************************
=                            ;
=                            ;
=                            ;
=
=                            MOTORCK:                                 ;
=3EE8 E413                           IN      AL,SYSSTA        ; AL <-- SYSTEM STATUS
=3EEA 2401                           AND     AL,01            ; Test DISK MOTOR ON BIT
=3EEC E614                           OUT     MOTORON,AL       ; Switch motor on
=3EEE 7501          3EF1             JNZ     MOTORCK1         ;
=3EF0 C3                             RET                      ; Return if motor was on
=                            MOTORCK1:                                ;
=3EF1 BBFFFF                         MOV     BX,0FFFFH        ; Wait some time if motor was off
=                            MOTORCK2:                                ;

=3EF4 D40A                           AAM                      ; (83)
=3EF6 4B                             DEC     BX               ; ( 2)
=3EF7 75FB          3EF4             JNZ     MOTORCK2         ; ( 8)  = 93 CLOCKS * FFFF = 1 sec
=                                                             ;
=3EF9 C3                             RET                      ;
=
=                    ;******************************************************************************
=                    ;******************************************************************************
=                    ;******************************************************************************
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;    ROUTINE NAME:      DMA
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;    FUNCTION:          DMA routines
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;    ENTRY VIA:         CALL
=                    ;
=                    ;
=                    ;    ENTRY CONDITIONS:  Following variables are set:
=                    ;                       DMAADDR (SEGMENT and OFFSET)
=                    ;                       DMALENG and DMAFUNC
=                    ;
=                    ;
=                    ;
=                    ;
=                    ;    EXIT VIA:          RETURN
=                    ;
=                    ;
=                    ;    EXIT CONDITIONS:   NONE
=                    ;
=                    ;
=                    ;
=                    ;******************************************************************************
=                    ;******************************************************************************
=                    ;******************************************************************************
=                    ;
=                    ;
=                    ;
=
=                     DMA:                               ;

=3EFA A00148                 MOV     AL,DMAFUNC          ; DMAFUNC <-- DMA FUNCTION
=3EFD E62B                   OUT     DMAMO,AL            ; OUT MODE
=                                                        ;
=3EFF A1FD47                 MOV     AX,DMAADDR+2        ; AX <-- DMA SEGMENT
=3F02 D1E0                   SHL     AX,1                ;
=3F04 D1E0                   SHL     AX,1                ;
=3F06 D1E0                   SHL     AX,1                ;
=3F08 D1E0                   SHL     AX,1                ;
=3F0A 0306FB47               ADD     AX,DMAADDR          ; AX <-- absolute addr within BANK
=3F0E E626                   OUT     C0AD,AL             ; OUT DMA ADDR low
=3F10 8AC4                   MOV     AL,AH               ;
=3F12 E626                   OUT     C0AD,AL             ; OUT DMA ADDR high
=                                                        ;
=3F14 A1FF47                 MOV     AX,DMALENG          ; AX <-- DMA LENGTH
=3F17 48                     DEC     AX                  ;
=3F18 E627                   OUT     C0TC,AL             ; OUT DMA LENGTH low
=3F1A 8AC4                   MOV     AL,AH               ;
=3F1C E627                   OUT     C0TC,AL             ; OUT DMA LENGTH high
=                                                        ;
=3F1E B600                   MOV     DH,00               ;
=3F20 B2E0                   MOV     DL,BANK             ; DX - BANK 0 initialisation
=3F22 800200                 ADC     DL,00               ; DX - next BANK if SEGMENT + OFFSET > 64K
=                                                        ;
=3F25 A1FD47                 MOV     AX,DMAADDR+2        ; AX <-- DMA SEGMENT
=3F28 D0EC                   SHR     AH,1                ;
=3F2A D0EC                   SHR     AH,1                ;
=3F2C D0EC                   SHR     AH,1                ;
=3F2E D0EC                   SHR     AH,1                ;
=3F30 02D4                   ADD     DL,AH               ; DX <-- BANK SELECT PORT
=                                                        ;
=3F32 EE                     OUT     DX,AL               ; SELECT BANK
=                                                        ; -----------
=3F33 B003                   MOV     AL,03               ;
=3F35 E62A                   OUT     DMAMB,AL            ; Enable FDC CHANNEL
=3F37 C3                     RET                         ;



```

```
                                    IF NOT LOADER_BIOS


=                                   INCLUDE C:WIPIMC.SEG
=                           ;
=                           ;****************************************
=                           ;*                                      *
=                           ;*   CHECK IF WINCHESTER DRIVE IS       *
=                           ;*    CONNECTED AND POWERED ON.         *
=                           ;*                                      *
=                           ;*    EXIT: ZERO FLAG ON = DRIVE READY  *
=                           ;*                                      *
=                           ;****************************************
=                           ;
=                           FIXREADY:
=3F38 B055                          MOV     AL,55H
=3F3A E6C4                          OUT     CYLLO,AL                ;OUTPUT PATTERN TO R/W PORT
=3F3C B0AA                          MOV     AL,0AAH
=3F3E E6C3                          OUT     SECNO,AL
=3F40 E4C4                          IN      AL,CYLLO                ;READ PATTERN BACK AND COMPARE
=3F42 3C55                          CMP     AL,55H
=3F44 7504              3F4A        JNZ     FIXREADY1
=3F46 E4C3                          IN      AL,SECNO
=3F48 3CAA                          CMP     AL,0AAH
=                           FIXREADY1:
=3F4A C3                            RET
=                           ;
=                           ;
=                           ;****************************************
=                           ;*                                      *
=                           ;*    WINCHESTER DISK DRIVER            *
=                           ;*                                      *
=                           ;*     ENTRY: PARAMETER BLOCK FILLED UP *
=                           ;*     EXIT:  STATUS BYTES IN PARAM.    *
=                           ;*            BLOCK UPDATED AND ALL     *
=                           ;*            REGISTERS SAVED.          *
=                           ;****************************************
=                           ;
=                           ;
=3F4B 50                    FIXDR:  PUSH    AX
=3F4C 53                            PUSH    BX
=3F4D 51                            PUSH    CX
=3F4E 52                            PUSH    DX
=3F4F A10C48                        MOV     AX,WORD PTR WIPAR+2     ;GET LOGIC SECTOR NUMBER
=3F52 B91100                        MOV     CX,17
=3F55 BA0000                        MOV     DX,0
=3F58 F7F1                          DIV     CX                      ;CALCULATE CYL/HEAD
=3F5A 50                            PUSH    AX
=3F5B 8AC2                          MOV     AL,DL
=3F5D E6C3                          OUT     SECNO,AL                ;SET SECTOR NUMBER
=3F5F 8A1E0A48                      MOV     BL,BYTE PTR WIPAR       ;GET DISK UNIT #
=3F63 8AFB                          MOV     BH,BL
=3F65 81E30106                      AND     BX,0601H
=3F69 D0C7                          ROL     BH,1                    ;SET DRIVE
```

```

=3F6B 0ADF                  OR      BL,BH               ;SET UNIT
=3F6D D0C3                  ROL     BL,1
=3F6F 58                    POP     AX
=3F70 50                    PUSH    AX
=3F71 2401                  AND     AL,01H              ;GET HEAD BIT
=3F73 0AC3                  OR      AL,BL
=3F75 0CA0                  OR      AL,SDHREG           ;ECC/CRC AND BYTES PER SECTOR
=3F77 E6C6                  OUT     SDH,AL              ;SET ECC/CRC-BYTES/SECT-DRIVE-HEAD
=3F79 58                    POP     AX
=3F7A D1C8                  ROR     AX,1
=3F7C E6C4                  OUT     CYLLO,AL            ;SET CYLINDER LOW
=3F7E 80E403                AND     AH,03H
=3F81 8AC4                  MOV     AL,AH
=3F83 E6C5                  OUT     CYLHI,AL            ;SET CYLINDER HIGH
=3F85 E4C7                  IN      AL,STAT             ;GET DISK STATUS
=3F87 A20E48                MOV     BYTE PTR WIPAR+4,AL
=3F8A 2480                  AND     AL,CBUSY            ;CHECK IF CONTROLLER BUSY
=3F8C 7516      3FA4        JNZ     FIXD3
=3F8E A00B48                MOV     AL,BYTE PTR WIPAR+1
=3F91 E6C7                  OUT     COMND,AL            ;SET FUNCTION
=3F93 24F0                  AND     AL,0F0H
=3F95 3C20                  CMP     AL,WIREAD
=3F97 7416      3FAF        JZ      RD                  ;GO READ DATA
=3F99 3C30                  CMP     AL,WIWRITE
=3F9B 744E      3FEB        JZ      WR                  ;GO WRITE DATA
=3F9D 3C50                  CMP     AL,FORMAT
=3F9F 7446      3FE7        JZ      WR0                 ;GO FORMAT ONE TRACK
=3FA1 E95B00    3FFF        JMP     WR2                 ;SEEK OR RESTORE
=3FA4 E4C6           FIXD3: IN      AL,SDH
=3FA6 0C18                  OR      AL,18H
=3FA8 E6C6                  OUT     SDH,AL              ;CLEAR DISK LAMP
=3FAA 5A                    POP     DX
=3FAB 59                    POP     CX
=3FAC 5B                    POP     BX
=3FAD 58                    POP     AX
=3FAE C3                    RET
=                    ;
=                    ;
=                    ;
=                    ;        ****************************
=                    ;        *     READ ROUTINE         *
=                    ;        ****************************
=                    ;
=3FAF E81F00    3FD1 RD:    CALL    WAIT                ;WAIT UNTIL READ COMPLETE
=3FB2 1E                    PUSH    DS
=3FB3 8B1E1248              MOV     BX,WORD PTR WIPAR+8 ;GET OFFSET
=3FB7 8E1E1048              MOV     DS,WORD PTR WIPAR+6 ;GET SEGMENT ADDR.
=3FBB B90002                MOV     CX,512              ;INPUT COUNT
=3FBE E4C0           RD2:   IN      AL,DATA             ;INPUT DATA
=3FC0 8807                  MOV     BYTE PTR[BX],AL     ;SAVE INPUT
=3FC2 43                    INC     BX
=3FC3 E0F9      3FBE        LOOPNZ  RD2         ;CONTINUE UNTIL ALL BYTES IN BUFFER
```

```

 =                                                          ;BUT STOP BEFORE BUFFER ADDR. WRAP AROUND
 =3FC5 83F900                    CMP    CX,0
 =3FC8 7404        3FCE          JZ     RD4
 =3FCA E4C0              RD3:    IN     AL,DATA             ;CLEAR CONTROLLER BUFFER
 =3FCC E2FC        3FCA          LOOP   RD3
 =3FCE 1F                RD4:    POP    DS
 =3FCF EB03        3FA4          JMPS   FIXD3
 =                       ;
 =                       ;
 =                       ;        ****************************
 =                       ;        *     WAIT ROUTINE         *
 =                       ;        ****************************
 =                       ;
 =3FD1 E4C7              WAIT:   IN     AL,STAT             ;GET STATUS
 =3FD3 2480                      AND    AL,CBUSY
 =3FD5 75FA        3FD1          JNZ    WAIT                ;LOOP UNTIL DISK READY
 =3FD7 E4C7                      IN     AL,STAT
 =3FD9 A20E48                    MOV    BYTE PTR WIPAR+4,AL ;SAVE STATUS
 =3FDC D0D8                      RCR    AL,1
 =3FDE 7201        3FE1          JC     ER1                 ;JUMP IF ERROR CONDITION
 =3FE0 C3                        RET
 =                       ;
 =3FE1 E4C1              ER1:    IN     AL,WIERROR          ;GET ERROR STATUS
 =3FE3 A20F48                    MOV    BYTE PTR WIPAR+5,AL ;SAVE STATUS
 =3FE6 C3                        RET
 =                       ;
 =                       ;        ****************************
 =                       ;        *     WRITE ROUTINE        *
 =                       ;        ****************************
 =                       ;
 =3FE7 B011              WR0:    MOV    AL,17
 =3FE9 E6C2                      OUT    SECNT,AL            ;SET SECT COUNT FOR FORMAT
 =                       ;
 =3FEB 1E                WR:     PUSH   DS
 =3FEC 8B1E1248                  MOV    BX,WORD PTR WIPAR+8 ;BUFFER ADDR.(OFFSET)
 =3FF0 8E1E1048                  MOV    DS,WORD PTR WIPAR+6 ;BUFFER ADDR.(SEGMENT)
 =3FF4 B90002                    MOV    CX,512              ;INPUT COUNT
 =3FF7 8A07              WR1:    MOV    AL,BYTE PTR[BX]     ;GET BYTE FROM BUFFER
 =3FF9 E6C0                      OUT    DATA,AL             ;OUTPUT DATA
 =3FFB 43                        INC    BX
 =3FFC E2F9        3FF7          LOOP   WR1
 =3FFE 1F                        POP    DS
 =3FFF E8CFFF      3FD1  WR2:    CALL   WAIT                ;WAIT UNTIL FUNCT. COMPLETE
 =4002 EBA0        3FA4          JMPS   FIXD3
 =                       ;
 =                       ;
 =                       ;


                                 ENDIF


                                  
                                  
 =                                    INCLUDE C:KBDMGRC.SEG
 =
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          ;
 =                          .

 =
 =                          ;*****************************************************
 =                          ;*                                                   *
 =                          ;*                                                   *
 =                          ;*                                                   *
 =                          ;*                                                   *
 =                          ;*  ROUTINE NAME: KEYST                              *
 =                          ;*  FUNCTION: GET KBD STATUS                         *
 =                          ;*                                                   *
 =                          ;*  ENTRY VIA: JUMP                                  *
 =                          ;*  ENTRY CONDITIONS: NONE                           *
 =                          ;*                                                   *
 =                          ;*  EXIT VIA: RETURN (TO BDOS)                       *
 =                          ;*  EXIT CONDITIONS: AL = 00 -> NO CHARACTER READY   *
 =                          ;*                   AL = FF -> CHARACTER READY      *
 =                          ;*                                                   *
 =                          ;*****************************************************
 =
 =
 =
 =                          KEYST:
 =4004 803E1448FF                   CMP    FUNACT,0FFH           ; CHECK IF FUNCTION ACTIVE
 =4009 7407           4012          JE     CHAR_READY            ; IF SO RETURN
 =400B E441                         IN     AL,BYTE PTR RSKEY     ; FOR PERFORMANCE REASONS, THE "IN" IS
                                                                                      DONE HERE
 =                                                               ; (NOT IN THE PIM)
 =400D 2401                         AND    AL,KBDAT86            ; CHECK FOR CHARACTER READY
 =400F 7501           4012          JNZ    CHAR_READY
 =4011 C3                           RET                          ; AL = 00 -> NO CHAR. READY
 =
 =
 =                          CHAR_READY:
 =4012 B0FF                         MOV    AL,0FFH               ; AL = FF -> CHAR. READY
 =
 =                          KEYST_END:
 =4014 C3                           RET
 =
 =
 =

 =
 =                        ;*******************************************************
 =                        ;*                                                     *
 =                        ;*                                                     *
 =                        ;*                                                     *
 =                        ;*                                                     *
 =                        ;*  ROUTINE NAME: KEYIN                                *
 =                        ;*  FUNCTION: GET CHARACTER FROM KBD                   *
 =                        ;*                                                     *
 =                        ;*  ENTRY VIA: JUMP                                    *
 =                        ;*  ENTRY CONDITIONS: NONE                             *
 =                        ;*                                                     *
 =                        ;*  EXIT VIA: RETURN (TO BDOS)                         *
 =                        ;*  EXIT CONDITIONS: AL = CHARACTER                    *
 =                        ;*                                                     *
 =                        ;*******************************************************
 =
 =
 =
 =                        KEYIN:
 =4015 803E1448FF                 CMP     FUNACT,0FFH             ; CHECK FOR FUNCTION ACTIVE
 =401A 743A          4056         JE      KEYIN2                  ; IF SO JUMP
 =                        KEYIN1:
 =401C E8E201        4201         CALL    KBD_IN                  ; GET CHAR. FROM KBD PIM
 =401F 3C9E                       CMP     AL,9EH                  ; CHECK FOR HEBREW ON
 =4021 7449          406C         JZ      HEBREW_ON
 =4023 3C9F                       CMP     AL,9FH                  ; HEBREW OFF?
 =4025 744C          4073         JZ      HEBREW_OFF
 =4027 3CA0                       CMP     AL,0A0H
 =4029 7240          406B         JB      KEYIN_END               ; RETURN VALUES < A0
 =402B 3CB3                       CMP     AL,0B3H
 =402D 7610          403F         JBE     FUN_CHECK               ; A0 - B3 -> FUNCTION KEY VALUE
 =402F 3CC0                       CMP     AL,0C0H
 =4031 72E9          401C         JB      KEYIN1                  ; B4 - BF -> INVALID ENTRY
 =4033 3CD3                       CMP     AL,0D3H
 =4035 7608          403F         JBE     FUN_CHECK               ; C0 - D3 -> FUNCTION KEY VALUE
 =4037 3CE0                       CMP     AL,0E0H
 =4039 72E1          401C         JB      KEYIN1                  ; D4 - DF -> INVALID ENTRY
 =403B 3CF3                       CMP     AL,0F3H
 =403D 77DD          401C         JA      KEYIN1                  ; E0 - F3 -> FUNCTION KEY VALUE
 =
 =                        FUN_CHECK:
 =403F 2E803E8F2500               CMP     CONFIGFL,00H            ; CHECK FOR CONFIG-FLAG SET
 =4045 7724          406B         JA      KEYIN_END               ; IF SO RETURN FUNCTION CHAR.
 =4047 E83000        407A         CALL    FUNSET                  ; SET POINTER TO START ADDR. OF FUNCT.
 =404A 3D0000                     CMP     AX,00H                  ; IF FUNCTION LENGTH = 0 -> INVALID
 =404D 7507          4056         JNZ     KEYIN2
 =404F C606144800                 MOV     FUNACT,0                ; RESET FUNCTION ACTIVE FLAG
 =4054 EBC6          401C         JMPS    KEYIN1
 =
 =                        KEYIN2:
 =4056 8B1E1548                   MOV     BX, FPOINTER
```

```

=405A 8A07                    MOV     AL, [BX]                ; GET FUNCTION CHARACTER
=405C FF061548                INC     FPOINTER                ; POINT TO NEXT CHARACTER OF FUNCTION
=4060 FF0E1748                DEC     FCHARCNT                ; DECREMENT FUNCTION LENGTH
=4064 7505          406B      JNZ     KEYIN_END               ; WAS IT THE LAST CHARACTER?
=4066 C606144800              MOV     FUNACT,00H              ; IF SO, RESET FUNCTION ACTIVE FLAG
=
=                       KEYIN_END:
=406B C3                      RET
=
=                       HEBREW_ON:
=406C C6063B48FF              MOV     HEBREW,0FFH
=4071 EBA9          401C      JMPS    KEYIN1
=
=                       HEBREW_OFF:
=4073 C6063B4800              MOV     HEBREW,00H
=4078 EBA2          401C      JMPS    KEYIN1

=
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;*                                                   *
=                       ;*                                                   *
=                       ;*                                                   *
=                       ;*  ROUTINE NAME: FUNSET                             *
=                       ;*  FUNCTION: GET START ADDRESS OF FUNCTION          *
=                       ;*                                                   *
=                       ;*  ENTRY VIA: CALL                                  *
=                       ;*  ENTRY CONDITIONS: AL = FUNCTION NUMBER           *
=                       ;*                                                   *
=                       ;*  EXIT VIA: RETURN                                 *
=                       ;*  EXIT CONDITIONS: FPOINTER = START ADDR. OF FUNCTION  *
=                       ;*                   FCHARCNT = LENGTH OF FUNCTION   *
=                       ;*                   FUNACT = FF -> FUNCTION ACTIVE  *
=                       ;*                                                   *
=                       ;*****************************************************
=
=
=
=                       FUNSET:
=407A 241F                    AND     AL,01FH                 ; CLEAR BITS 8...6
=407C 3C14                    CMP     AL,20
=407E 7713          4093      JA      FUNSET_END              ; FUNCTION NR. > 20 -> INVALID FUNCTION
=4080 8AC8                    MOV     CL,AL
=4082 FEC1                    INC     CL
=4084 E80D00        4094      CALL    GETFPOS                 ; GET POSITION OF FUNCTION IN FUNTBL.
=4087 A31748                  MOV     FCHARCNT,AX             ; LENGTH OF FUNCTION -> FCHARCNT
=408A 891E1548                MOV     FPOINTER,BX             ; SAVE START ADDRESS OF FUNCTION
=408E C6061448FF              MOV     FUNACT,0FFH             ; SET FUNCTION ACTIVE FLAG
=
=                       FUNSET_END:
=4093 C3                      RET
```

```

 =                          ;**************************************************************
 =                          ;*                                                            *
 =                          ;*                                                            *
 =                          ;*                                                            *
 =                          ;*                                                            *
 =                          ;*  ROUTINE NAME: GETFPOS                                     *
 =                          ;*  FUNCTION: GET POSITION OF FUNCTION IN FUNCTION TABLE      *
 =                          ;*                                                            *
 =                          ;*  ENTRY VIA: CALL                                           *
 =                          ;*  ENTRY CONDITIONS: CL = FUNCTION NUMBER                    *
 =                          ;*                                                            *
 =                          ;*  EXIT VIA: RETURN                                          *
 =                          ;*  EXIT CONDITIONS: AX = FUNCTION LENGTH                     *
 =                          ;*                   BX = START ADDRESS  OF FUNCTION          *
 =                          ;*                                                            *
 =                          ;**************************************************************
 =
 =
 =
 =                          GETFPOS:
 =4094 BD8D25                       MOV     BP,OFFSET FUNC_TABLE    ; GET START ADDRESS OF TABLE
 =4097 BE0000                       MOV     SI,0000H
 =409A B500                         MOV     CH,00H                  ; CX = COUNTER
 =
 =                          GETFUN:
 =409C 8B02                         MOV     AX,[BP+SI]              ; GET LENGTH OF TABLE ENTRY
 =409E 03F0                         ADD     SI,AX                   ; ADD LENGTH OF ENTRY TO OFFSET POINTER
 =40A0 E2FA          409C           LOOP    GETFUN
 =
 =40A2 48                           DEC     AX
 =40A3 48                           DEC     AX                      ; DECREMENT LENGTH
 =40A4 2BF0                         SUB     SI,AX                   ; WE NOW POINT TO THE END OF THE
 =40A6 8BDE                         MOV     BX,SI                   ; FUNCTION, SO SUBTRACT THE LENGTH
 =40A8 03DD                         ADD     BX,BP                   ; TO GET THE START ADDRESS
 =
 =                          GFP_END:
 =40AA C3                           RET
```

```

=
=                       ;*************************************************************
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*  ROUTINE NAME: GETFCHAR                                   *
=                       ;*  FUNCTION: ERASE THE FUNCTION TO BE CHANGED               *
=                       ;*                                                           *
=                       ;*  ENTRY VIA: JUMP                                          *
=                       ;*  ENTRY CONDITIONS: CL = FUNCTION NUMBER                   *
=                       ;*                                                           *
=                       ;*  EXIT VIA: RETURN                                         *
=                       ;*  EXIT CONDITIONS: NONE                                    *
=                       ;*                                                           *
=                       ;*************************************************************
=
=                       GETFCHAR:
=40AB 880E1948                  MOV     FNCCHAR,CL                      ; SAVE FUNCTION NUMBER
=40AF 80E11F                    AND     CL,1FH                          ; CLEAR BITS 8..6
=40B2 FEC1                      INC     CL
=40B4 E8D0FF       4094         CALL    GETFPOS                         ; GET POS. OF FUNCTION
=40B7 891E1A48                  MOV     FNSTR,BX                        ; SAVE START ADDRESS
=40BB 891E1C48                  MOV     FNACT,BX                        ; OF FUNCTION
=40BF 03D8                      ADD     BX,AX                           ; ADD LENGTH OF FUNCTION
=40C1 8BD3                      MOV     DX,BX                           ; DX = END ADDR. OF FUNCTION TO
                                                                                              BE CHANGED
=40C3 B114                      MOV     CL,20                           ; GET POSITION OF LAST FUNCTION
=40C5 E8CCFF       4094         CALL    GETFPOS                         ; (#20) IN FUNCTION TABLE
=40C8 03D8                      ADD     BX,AX                           ; CALCULATE LENGTH OF FUNCTIONS
=40CA 2BDA                      SUB     BX,DX                           ; FROM ACTUAL FUNCTION TO END
=40CC 8BCB                      MOV     CX,BX
=40CE 891E1E48                  MOV     RSTLEN,BX                       ; OF FUNCTION TABLE AND SAVE IT
=40D2 8BF2                      MOV     SI,DX                           ; START ADDRESS OF ACTUAL FUNCTION
=40D4 8B3E1A48                  MOV     DI,FNSTR                        ; GET LENGTH OF FUNCTION
=40D8 FC                        CLD
=40D9 F3A4                      REP MOVS AL,AL
=40DB 893E2048                  MOV     FNEND,DI
=40DF C70622480200              MOV     FNLEN,2                         ; SET FUNCTION LENGTH = 0
=40E5 C7060744F140              MOV     DRQ_ADRS,OFFSET CHAN_CHAR       ; SET ADDR. OF "CHANGE FUNCT. CHAR."
                                                                                                 ROUTINE
=40EB 800E044401                OR      STATUS_FLAG,DRQFLG              ; SET DATA REQUEST FLAG
=40F0 C3                        RET
```

```
=
=                       ;*************************************************************
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*                                                           *
=                       ;*  ROUTINE NAME: CHANCHAR                                   *
=                       ;*  FUNCTION: INSERT ONE CHARACTER IN FUNCTION TABLE         *
=                       ;*                                                           *
=                       ;*  ENTRY VIA: JUMP                                          *
=                       ;*  ENTRY CONDITIONS: CL = CHARACTER                         *
=                       ;*                                                           *
=                       ;*  EXIT VIA: RETURN                                         *
=                       ;*  EXIT CONDITIONS: NONE                                    *
=                       ;*                                                           *
=                       ;*************************************************************
=
=                       CHAN_CHAR:
=40F1 380E1948                  CMP     FNCCHAR,CL              ; IS CHAR. = FUNCTION # ?
=40F5 743A         4131         JE      CHAN_END                ; IF YES, IT'S END OF FUNCTION
=40F7 803E2448FF                CMP     FNERR,0FFH              ; HAVE WE GOT AN ERROR?
=40FC 722D         412B         JB      CHAN_CHAR_END           ; IF SO JUMP TO THE END
=40FE 8AD1                      MOV     DL,CL
=4100 FF062248                  INC     FNLEN                   ; INCREMENT FUNCTION LENGTH
=4104 FF062048                  INC     FNEND                   ; END OF FUNCTIONS WILL MOVE 1 BYTE
=4108 B8CD29                    MOV     AX,OFFSET FUN_END
=410B 39062048                  CMP     FNEND,AX                ; DID WE REACH END OF FUNCT. TABLE?
=410F 7733         4144         JA      FUN_ERR                 ; IF SO GO TO ERROR ROUTINE
=4111 FD                        STD                             ; SET REVERSE DIRECTION
=4112 A12048                    MOV     AX,FNEND
=4115 48                        DEC     AX
=4116 8BF8                      MOV     DI,AX
=4118 48                        DEC     AX
=4119 8BF0                      MOV     SI,AX
=411B 8B0E1E48                  MOV     CX,RSTLEN
=
=411F F3A4                      REP     MOVS    AL,AL           ; MOVE REST OF FUNCTIONS ONE BYTE
=4121 8B3E1C48                  MOV     DI,FNACT
=4125 8815                      MOV     [DI],DL                 ; INSERT CHARACTER AT CURRENT LOCATION
=4127 FF061C48                  INC     FNACT                   ; POINT TO NEXT LOCATION
=
=                       CHAN_CHAR_END:
=412B 800E044401                OR      STATUS_FLAG,DRQFLG      ; SET DATA REQUEST BYTE
=4130 C3                        RET
=

 =
 =
 =                          CHAN_END:
 =4131 803E2448FF                   CMP     FNERR,0FFH              ; DID WE GET AN ERROR
 =4136 7232            416A         JB      FUN_ERR_DISP            ; IF YES, GO AND DISPLAY IT
 =4138 A12248                       MOV     AX,FNLEN                ; LENGTH OF FUNCTION
 =413B 8B3E1A48                     MOV     DI,FNSTR                ; IS FIRST WORD OF
 =413F 4F                           DEC     DI
 =4140 4F                           DEC     DI                      ; FUNCTION ENTRY
 =4141 8905                         MOV     [DI],AX
 =4143 C3                           RET
 =
 =
 =                          ;*** THIS ROUTINE IS ENTERED IF THE END OF FUNCTION TABLE WAS REACHED
 =                          FUN_ERR:
 =4144 C606244800                   MOV     FNERR,00H               ; SET FUNCTION ERROR FLAG
 =4149 8B0E1E48                     MOV     CX,RSTLEN
 =414D 8B3E1A48                     MOV     DI,FNSTR
 =4151 8B361C48                     MOV     SI,FNACT
 =4155 FC                           CLD
 =4156 F3A4                         REP     MOVS    AL,AL           ; ERASE ALREADY ENTERED CHAR.
 =4158 8B3E1A48                     MOV     DI,FNSTR
 =415C 4F                           DEC     DI
 =415D C60500                       MOV     BYTE PTR [DI],0
 =4160 4F                           DEC     DI
 =4161 C60502                       MOV     BYTE PTR [DI],2         ; SET LENGTH OF FUNCTION = 0
 =4164 800E044401                   OR      STATUS_FLAG,DRQFLG      ; SET DATA REQUEST FLAG
 =4169 C3                           RET
 =
 =
 =                          ;*** DISPLAY ERROR MESSAGE IF END OF FUNCTION TABLE HAS BEEN REACHED
 =
 =                          FUN_ERR_DISP:
 =416A BB2548                       MOV     BX,OFFSET FN_ERR_MESS
 =416D E8C5ED          2F35         CALL    ERR_DISP
 =4170 E85AEE          2FCD         CALL    ERR_DISP1
 =4173 C3                           RET
 =
 =



 =                        INCLUDE C:KBDPINC.SEG
 =                  ;
 =                  ;
 =                  ;     ##############################################
 =                  ;     ##                                          ##
 =                  ;     ##             K E Y B O A R D              ##
 =                  ;     ##                                          ##
 =                  ;     ##                 P I N                    ##
 =                  ;     ##                                          ##
 =                  ;     ##############################################
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;   ROUTINE NAME:       KBD_INIT
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;   FUNCTION:           INITIALIZE THE KEYBOARD AND GET ITS LANGUAGE CODE

 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;   ENTRY VIA:          CALL
 =                  ;
 =                  ;
 =                  ;   ENTRY CONDITIONS:   MUST BE FIRST ROUTINE ON KEYBOARD AFTER THE POWER UP
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;   EXIT VIA:           RETURN
 =                  ;
 =                  ;
 =                  ;   EXIT CONDITIONS:    AL = LANGUAGE CODE  (00H - 07H)
 =                  ;
 =                  ;
 =                  ;
 =                  ;##################################################################
 =                  ;##################################################################
 =                  ;##################################################################
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;
```

```
=                             kbd_init:
=4174 B001                                mov     al,country               ; load command to get language code
=4176 E641                                out     byte ptr kcount,al       ; send this command
=                             kbd_init_1:
=4178 E441                                in      al,byte ptr rskey        ; get keyboard status
=417A A801                                test    al,kbdat86               ; when data not ready
=417C 74FA              4178              jz      kbd_init_1               ; try again (loop)
=417E E441                                in      al,byte ptr rskey        ;
=4180 A880                                test    al,lgdat86               ; when language code ready
=4182 7505              4189              jnz     kbd_init_2               ; get it
=4184 E440                                in      al,byte ptr rdkey        ; dummy read neede for 8741 controller
=4186 E9EFFF            4178              jmp     kbd_init_1               ; try again
=                             kbd_init_2:
=4189 E440                                in      al,byte ptr rdkey        ; get language code
=418B C6063C4807                          mov     language,07h
=4190 20063C48                            and     language,al              ; clear bits:7,...,3
=4194 24F8                                and     al,not 07h               ; clear lower bits
=4196 B90300                              mov     cx,03h                   ; look for the 3 variantes
=                             kbd_init_4:
=4199 3A063D48                            cmp     al,kbd_var               ; get # of
=419D 740C              41AB              jz      kbd_init_5               ; keyboard variante
=419F 80063C4810                          add     language,10h             ; and change
=41A4 802E3D4810                          sub     kbd_var,10h              ; language code
=41A9 E2EE              4199              loop    kbd_init_4               ; accordingly
=                             kbd_init_5:


=41AB 803E3C4801                          cmp     language,01h                                   ; if language is
=41B0 763F              41F1              jbe     kbd_init_6
=41B2 803E3C4810                          cmp     language,10h
=41B7 7438              41F1              jz      kbd_init_6
=41B9 803E3C4811                          cmp     language,11h
=41BE 7431              41F1              jz      kbd_init_6
=41C0 803E3C4823                          cmp     language,23h                                   ; CANADA
=41C5 742A              41F1              jz      kbd_init_6
=41C7 803E3C4832                          cmp     language,32h                                   ; HEBREW
=41CC 7419              41E7              jz      kbd_init_7
=41CE 2EC6064C42A2C                       mov     byte ptr dec_sign_1,2ch                        ;; SPAR 02332
=41D4 2EC606D42A2C                        mov     byte ptr dec_sign_2,2ch                        ;;     "
=41DA BFD82A                              mov     di,offset kbd_tt +1eh
=41DD C6051E                              mov     byte ptr [di],1eh                              ; for Hebrew the codes
=41E0 47                                  inc     di                                             ; 9Eh and 9Fh switch on
=41E1 C6051F                              mov     byte ptr [di],1fh                              ; and off display of
=41E4 E91600            41FD              jmp     kbd_init_3
=                             kbd_init_7:
=41E7 BFD82A                              mov     di,offset kbd_tt +1eh
=41EA C6059E.                             mov     byte ptr [di],9eh                              ; for Hebrew the codes
=41ED 47                                  inc     di                                             ; 9Eh and 9Fh switch on
=41EE C6059F                              mov     byte ptr [di],9fh                              ; and off display of
=                                                                                                ; hebrew characters
=                             kbd_init_6:
=41F1 2EC606C42A2E                        mov     byte ptr dec_sign_1,2eh                        ; 00 = us or  01 = uk
=41F7 2EC606D42A2E                        mov     byte ptr dec_sign_2,2eh                        ; use decimal point
=                                                                                                ; instead of comma
=                             kbd_init_3:
=41FD C3                                  ret
=                             ;
```

```
 =                    ;************************************************************************
 =                    ;************************************************************************
 =                    ;************************************************************************
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;   ROUTINE NAME:      KBD_ST
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;   FUNCTION:          GET STATUS OF KEYBOARD CONTROLLER
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;   ENTRY VIA:         CALL

 =                    ;
 =                    ;
 =                    ;   ENTRY CONDITIONS:  NON
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;   EXIT VIA:          RETURN
 =                    ;
 =                    ;
 =                    ;   EXIT CONDITIONS:   AL = STATUS OF KEYBOARD CONTROLLER
 =                    ;
 =                    ;
 =                    ;
 =                    ;************************************************************************
 =                    ;************************************************************************
 =                    ;************************************************************************
 =                    ;
 =                    ;
 =                    ;
 =                    kbd_st:
 =41FE E441                  in      al,byte ptr rskey          ; get status of keyboard controller
 =4200 C3                    ret
 =                    ;
 =                    ;
 =                    ;
 =                    ;
 =                    ;
```

```
=                         ;*********************************************************************
=                         ;*********************************************************************
=                         ;*********************************************************************
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;   ROUTINE NAME:      KBD_IN
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;   FUNCTION:          GET AN INPUT FROM KEYBOARD
=                         ;                      (AND WAIT UNTIL ONE IS COMING
=                         ;
=                         ;
=                         ;
=                         ;   ENTRY VIA:         CALL
=                         ;
=                         ;

=                         ;   ENTRY CONDITIONS:  NON
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;   EXIT VIA:          RETURN
=                         ;
=                         ;
=                         ;   EXIT CONDITIONS:   AL = CHARACTER FROM KEYBOARD INPUT
=                         ;
=                         ;
=                         ;
=                         ;*********************************************************************
=                         ;*********************************************************************
=                         ;*********************************************************************
=                         ;
=                         ;
=                         ;
=                         kbd_in:
=4201 E441                        in      al,byte ptr rskey          ; wait for character ready
=4203 A801                        test    al,kbdat86
=4205 74FA          4201          jz      kbd_in                     ;  (loop)
=4207 E440                        in      al,byte ptr rdkey          ; get character for keyboard
=4209 3C80                        cmp     al,80h                     ; if char is a ASCII one
=420B 720B          4218          jb      kbd_in_2                   ; okay  return
=420D 3CA0                        cmp     al,0a0h                    ; also function keys are returned
=420F 7307          4218          jae     kbd_in_2
=4211 241F                        and     al,1fh                     ; all char. ) 80h and ( a0h
=4213 BBBA2A                      mov     bx,offset kbd_tt           ; are translated
=4216 2ED7                        xlat    CS:KBD_TT                  ; by the keyboard translation table
=                                                                    ; the character ) 80h
=                         kbd_in_2:
=4218 C3                          ret
=                         ;
=                         ;
=                         ;
=                         ;
=                         ;
=                ;************************************************************************
=                ;************************************************************************
=                ;************************************************************************
=                ;
=                ;
=                ;
=                ;
=                ;
=                ;
=                ;
=                ;   ROUTINE NAME:      KBD_OUT
=                ;
=                ;
=                ;

=                ;
=                ;
=                ;   FUNCTION:          OUTPUT TO KEYBOARD
=                ;
=                ;
=                ;
=                ;
=                ;   ENTRY VIA:         CALL
=                ;
=                ;
=                ;   ENTRY CONDITIONS:  CL = CHARACTER FOR RETREIVE ON KEYBOARD
=                ;                      (WAITING UNTIL KEYBOARD CAN TAKE IT)
=                ;
=                ;
=                ;
=                ;   EXIT VIA:          RETURN
=                ;
=                ;
=                ;   EXIT CONDITIONS:   NON
=                ;
=                ;
=                ;
=                ;************************************************************************
=                ;************************************************************************
=                ;************************************************************************
=                       ;
=                       ;
=                       ;
=                       kbd_out:
=                       kbd_out_2:
=                                                               ; output character in CL
=4219 E441                      in      al,byte ptr rskey       ; get keyboard status
=421B A801                      test    al,kbdat86              ; when a character is ready
=421D 7402          4221        jz      kbd_out_1               ;
=421F E440                      in      al,byte ptr rdkey       ; do a dummy read (needed for 8741 con
                        troller)
=                       kbd_out_1:
=4221 E441                      in      al,byte ptr rskey       ; get keyboard status
=4223 A802                      test    al,inpbuff86            ; and check whether output to kbd can
                        be done
=4225 75F2          4219        jnz     kbd_out_2               ; if not, try again
=4227 8AC1                      mov     al,cl                   ; get character for output
=4229 E641                      out     byte ptr kbell,al       ; and send it
=422B C3                        ret
=                       ;
=                       ;
=                       ;
=                       ;
```

```
                           IF NOT LOADER_BIOS

=                               INCLUDE C:SERPIMC.SEG
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;*****************************************************************************
=                       ;*                                                                           *
=                       ;*           SERIAL INTERFACE PERIPHERAL INTERFACE MODULE                    *
=                       ;*                                                                           *
=                       ;*****************************************************************************
=                       ;
=                       ;
=                       ; SERIAL OUTPUT ENTRY POINT
=                       ;
=422C BB3A42            SRLOUT:         MOV     BX,OFFSET SO_DISP_TBL
=422F 2EA09325          SIF_DISP:       MOV     AL,PVRS232      ;GET PROTOCOL VECTOR
=4233 D0E0                              SHL     AL,1            ;AL*2...TABLE TYPE WORD
=4235 98                                CBW                     ;EXPAND BYTE IN AL TO WORD IN AX
=4236 03D8                              ADD     BX,AX           ;BX = POINTER TO ROUTINE ADDRESS
=4238 FF27                              JMP     WORD PTR [BX]   ;JUMP TO ROUTINE FOR DEFINED PROTOCOL
=
=                       SO_DISP_TBL:
=423A A542                              DW      SPAOUT
=423C A542                              DW      SPAOUT
=423E A542                              DW      SPAOUT
=4240 A542                              DW      SPAOUT
=
=                       SST_DISP_TBL:
=4242 7542                              DW      SPAOST
=4244 7542                              DW      SPAOST
=4246 7542                              DW      SPAOST
=4248 7542                              DW      SPAOST
=                       ;

=                      ; SERIAL OUTPUT STATUS
=                      ;
=424A BB4242           SRLSTAT:     MOV   BX,OFFSET SST_DISP_TBL
=424D E9DFFF     422F               JMP   SIF_DISP      ;JUMP TO ROUTINE ACCORDING TO PROTOCOL
=                      ;
=                      ; GET INPUT STATUS
=                      ;
=4250 F6063E48FF       SPAIST:      TEST  SACTIVE,-1    ;TEST FOR SERIAL I/F ACTIVE
=4255 7503       425A               JNZ   SPAI1         ; JUMP IF TRUE
=4257 E85500     42AF               CALL  SIOINIT       ;INITIALIZE SERIAL I/F IF REQUIRED
=425A E461             SPAI1:       IN    AL,SPRSTAT
=425C 2438                          AND   AL,OVERRUN OR PARITY OR FRAMING
=425E 7403       4263               JZ    SPAI2         ;JUMP IF NONE OF CHECKED ERRORS OCCURED
=4260 E80900     426C               CALL  TRERR         ;CALL ERROR ROUTINE, ERROR ENCOUNTERED
=                                                       ; IN RECEIVER
=4263 E461             SPAI2:       IN    AL,SPRSTAT
=4265 2402                          AND   AL,RXRDY      ;TEST FOR CHARACTER RECEIVED
=4267 7402       426B               JZ    SPAI3         ; JUMP IF NOT
=4269 0CFF                          OR    AL,-1         ;FLAG CHARACTER RECEIVED
=426B C3               SPAI3:       RET
=
=426C E460             TRERR:       IN    AL,SPRDATA    ;DUMMY READ
=426E E463                          IN    AL,SPRCOM     ;READ COMMAND BYTE
=4270 0C10                          OR    AL,10H        ;RESET ERROR
=4272 E667                          OUT   SPWCOM,AL
=4274 C3                            RET
=                      ;
=                      ; GET PRINTER STATUS
=                      ;
=4275 F6063E48FF       SPAOST:      TEST  SACTIVE,-1    ;TEST FOR SERIAL I/F ACTIVE
=427A 7503       427F               JNZ   SPA1          ; SKIP INITIALIZATION IF TRUE
=427C E83000     42AF               CALL  SIOINIT       ; INITIALIZE THE SERIAL I/F
=427F E8CEFF     4250  SPA1:        CALL  SPAIST        ;CHECK INPUT STATUS
=4282 7406       428A               JZ    SPA2          ;JUMP IF NO INPUT
=4284 E81600     429D               CALL  SPAIN         ;GET INPUT CHARACTER
=4287 A24048                        MOV   XOFFFLG,AL
=428A 803E404813       SPA2:        CMP   XOFFFLG,XOFF  ;TEST FOR PRINTER NOT READY
=428F 7409       429A               JZ    SPA3          ;JUMP IF XOFF .. PRINTER NOT READY
=4291 E461                          IN    AL,SPRSTAT
=4293 2401                          AND   AL,TXRDY      ;TEST FOR TRANSMITTER READY
=4295 7402       4299               JZ    SPA4          ; JUMP IF NOT
=4297 0CFF                          OR    AL,-1         ;FLAG TRANSMITTER READY
=4299 C3               SPA4:        RET
=429A 32C0             SPA3:        XOR   AL,AL         ;FLAG PRINTER NOT READY
=429C C3                            RET
=                      ;
=                      ; GET CHARACTER FROM INTERFACE
=                      ;
=429D E8B0FF     4250  SPAIN:       CALL  SPAIST        ;CHECK INPUT STATUS
=42A0 74FB       429D               JZ    SPAIN         ;WAIT IF ZERO
=42A2 E460                          IN    AL,SPRDATA    ;GET CHARACTER
=42A4 C3                            RET
```

```

=                             ;
=                             ; OUTPUT CHARACTER
=                             ;
=42A5 E8CDFF             4275 SPAOUT:      CALL    SPAOST          ;CHECK OUTPUT STATUS
=42A8 74FB               42A5              JZ      SPAOUT          ;WAIT IF ZERO
=42AA 86C1                                 XCHG    AL,CL           ;CHARACTER TO AL
=42AC E664                                 OUT     SPWDATA,AL      ;OUTPUT THE CHARACTER
=42AE C3                                   RET
=                             ;
=                             ; INITIALIZE THE SERIAL I/O
=                             ;
=42AF 2EA09025                SIOINIT:     MOV     AL,M1RS232      ;GET FRAMING AND MODE
=42B3 E666                                 OUT     SPWMODE,AL      ;OUT MODE 1 BYTE
=42B5 2EA09125                             MOV     AL,M2RS232      ;CLOCK AND SPEED
=42B9 E666                                 OUT     SPWMODE,AL      ;OUT MODE 2 BYTE
=42BB B037                                 MOV     AL,37H          ;ENABLE TRANSMITTER AND RECEIVER
=42BD E667                                 OUT     SPWCOM,AL       ; SET DTR AND RTS, RESET ERROR
=42BF C6063E48FF                           MOV     SACTIVE,-1      ;FLAG  SERIAL INTERFACE AS ENABLED
=42C4 C6063F4800                           MOV     PACTIVE,0       ;FLAG PARALLEL INTERFACE DISABLED
=42C9 C3                                   RET

```

```
5614
5615
5616  =                                   INCLUDE C:PARPINC.SEG
5617  =                           ;
5618  =                           ;
5619  =                           ;
5620  =                           ;
5621  =                           ;
5622  =                           ;
5623  =                           ;
5624  =                           ;
5625  =                           ;
5626  =                           ;
5627  =                           ;
5628  =                           ;
5629  =                           ;
5630  =                           ;
5631  =                           ;
5632  =                           ;
5633  =                           ;
5634  =                           ;
5635  =                           ;
5636  =                           ;
5637  =                           ;
5638  =
5639  =                           ;*************************************************************
5640  =                           ;*************************************************************
5641  =                           ;
5642  =                           ;               PARALLEL INTERFACE (CENTRONICS)
5643  =                           ;
5644  =                           ;*************************************************************
5645  =                           ;*************************************************************
5646  =                           ;
5647  =                           ; INITIALIZE PARALLEL INTERFACE
5648  =                           ;
5649  =42CA B0AA                  PINIT:        MOV     AL,0AAH
5650  =42CC E663                                OUT     PBCOM,AL        ;INITIALIZE INTERFACE
5651  =42CE C6063E4800                          MOV     SACTIVE,0       ;DISABLE SERIAL INTERFACE
5652  =42D3 C6063F48FF                          MOV     PACTIVE,-1      ;FLAG PARALLEL I/F AS ACTIVE
5653  =42D8 C3                                  RET
5654  =
5655  =                           ;
5656  =                           ; OUTPUT CHARACTER IN CL
5657  =                           ;
5658  =42D9 E80700           42E3 P1CHROUT:     CALL    P1STATUS        ;CHECK INTERFACE STATUS
5659  =42DC 74FB             42D9               JZ      P1CHROUT        ; WAIT
5660  =42DE 86C1                                XCHG    AL,CL           ;CHARACTER TO AL
5661  =42E0 E660                                OUT     PBDA,AL         ;OUTPUT THE CHARACTER IN AL
5662  =42E2 C3                                  RET
5663  =                           ;
5664  =                           ; GET PRINTER STATUS
5665  =                           ;
5666  =42E3 F6063F48FF            P1STATUS:     TEST    PACTIVE,-1      ;TEST FOR PARALLEL I/F ACTIVE
```

```

=42E8 7503      42ED                  JNZ    P1STA1          ;JUMP IF ACTIVE
=42EA E8DDFF    42CA                  CALL   PINIT           ;INITIALIZE PARALLEL I/F
=42ED E461           P1STA1:          IN     AL,PBSTA        ;GET PRINTER STATUS
=42EF 2422                            AND    AL,BUSY OR POBF
=42F1 7403      42F6                  JZ     P1STATX         ;JUMP IF PRINTER ACCEPTS A BYTE
=42F3 32C0                            XOR    AL,AL           ;ZERO INDICATES PRINTER NOT READY
=42F5 C3                              RET
=
=42F6 0CFF           P1STATX:         OR     AL,-1           ;NOT ZERO INDICATES PRINTER READY
=42F8 C3                              RET
=
=                    ;
=                    ;

                             ENDIF


```

```


  42F9              DATASEG EQU     OFFSET $
                            DSEG
                            ORG     DATASEG
                    ;
                    ; *** BIOS GLOBAL DATA
                    ;
  0680              TPA_START EQU   680H            ; SEGMENT START ADDRESS OF TPA (PHYSICAL)
                                                    ; RELATIVE TO SEGMENT 40H, THIS IS ADDR 5C00H
                    ;
                    ; ATTENTION!!!! IF THIS VALUE CHANGES SOME OTHER VALUES HAVE TO BE CHECKED TOO:
                    ; START ADDRESS OF MOVCPM  (BY CHANGING SIZE OF PATCH AREA)
                    ; 900H AS SIZE OF 2. OS + DDT IN MOVCPM
                    ; 500H AS TPA START WITH DDT IN MOVCPM
                    ; 2800H AS A COUNTER FOR THE MOVS IN MOVCPM
                    ; START ADDRESS OF DISK BUFFERS IN DISKMANAGER CODE SEGMENT
                    ;
                    ;
                    ;
                    ;
  0980              TPA_LENGTH EQU  1000H-TPA_START         ; SEGMENT LENGTH OF TPA (ASSUMING 64K)
  FE06              MEMSIZ  EQU     0FE06H
                    ;
                    ; *** SEGMENT - OFFSET FOR JMPF TO INIT40 (SEGMENT 40H)
                    ;
 42F9 002A          PARA40  DW      OFFSET(INIT40)  ;ENTRY POINT INTO 400 HEX BIOS
 42FB 4000                  DW      40H
  42F9              BIOS40  EQU     DWORD PTR PARA40
                    ;
                    ;       ***     MEMORY REGION TABLE
                    ;
 42FD 01            MRT     DB      1       ; ONE MEMORY REGION  ( END OF O.S. TO END OF MEMORY
 42FE 8006                  DW      OFFSET TPA_START
 4300 8009          MRTLEN  DW      OFFSET TPA_LENGTH
                    ;
  000D              CR      EQU     0DH     ;CARRIAGE RETURN
  000A              LF      EQU     0AH     ;LINE FEED
  000A              COMLEN  EQU     0AH     ;CCP BUFFER LENGTH
                    ;
 4302 0D0A494E5445  INT_TRP DB      CR,LF,'INTERRUPT TRAP HALT',CR,LF,0FFH
      525255505420
      545241502048
      414C540D0AFF
                    ;
 431A 1A            SIGNON  DB      01AH    ; CLEAR SCREEN
 431B 43502F4D2038          DB      'CP/M-86 (R) 1.1 for NCR DECISION MATE V',CR,LF
      362028522920
      312E3120666F
      72204E435220
      444543495349
      4F4E204D4154
      4520560D0A
```

```

4344 203634482042      DISPMEM DB      ' 64K Byte Memory',CR,LF
     797465204D65
     6D6F72790D0A
4356 443030362D30              DB      'D006-0065-0000',CR,LF
     303635203030
     30300D0A
4366 436F70797269              DB      'Copyright (c) 1982, DIGITAL RESEARCH',CR,LF
     676874202863
     292031393832
     2C2044494749
     54414C205245
     534541524348
     0D0A
438C 53657269616C              DB      'Serial Number '
     204E756D6265
     7220
439A 2020202020        D_SER_NUM DB    '     '
439F 0D0A                      DB      CR,LF
43A1 4669726D7761      FWMESS1 DB      'Firmware Version: '
     726520566572
     73696F6E3A20
43B3 202020202020      FWMESS2 DB      '        '
     2020
43BB 0D0AFF                    DB      CR,LF,0FFH

  0FF7                 FWVERSION       EQU     0FF7H   ; FIRMWARE VERSION
  0010                 RAMSELECT       EQU     10H     ; SWITCH TO RAM
  0011                 ROMSELECT       EQU     11H     ; SWITCH TO ROM

43BE                   LOC_STK RW      32      ;LOCAL STACK FOR INITIALIZATION
  43FE                 STKBASE EQU     OFFSET $
```

```



                        ;
                        ;       ***     INCLUDE DATA AREAS FOR DRIVERS AND MANAGERS
                        ;
=                               INCLUDE C:CRTMGRD.SEG
=                       ;***************************************************************************
=                       ;
=                       ;       CRT MANAGER DATA AREA
=                       ;
=                       ;***************************************************************************
=                       ;
=                       ;
=43FE 00                CRTPB   DB      0       ; CURCOL
=43FF 00                        DB      0       ; CURROW
=4400 E8                        DB      0E8H    ; ATTRIBUTE
=4401 00                        DB      0       ; ESCAPE CODE
=4402 00                        DB      0       ; FREQUENCY (for music)
=4403 00                        DB      0       ; FREQUENCY LENGTH (for music)
=4404 00                STATUS_FLAG DB  0       ; STATUS FLAG (01=DATA REQUEST,02=ESCAPE SEQUENCE)
=  0001                 DRQFLG  EQU     01      ; DATA REQUEST FLAG
=  FFFE                 NOT_DRQFLG      EQU     0FFFEH
=  0002                 ESCFLG  EQU     02      ; ESCAPE SEQUENCE FLAG
=  FFFD                 NOT_ESCFLG      EQU     0FFFDH
=  0004                 HALF_INTENSITY  EQU     4
=  FFFB                 NOT_HALF_INTENSITY      EQU     0FFFBH
=  0001                 INVERSE         EQU     1
=  FFFE                 NOT_INVERSE     EQU     0FFFEH
=  0002                 BLINKING        EQU     2
=  FFFD                 NOT_BLINKING    EQU     0FFFDH
=  0005                 COLOUR_HALF_I   EQU     5
=  FFFA                 NOT_COLOUR_HALF_I EQU   0FFFAH
=4405 00                COLOUR_INDEX    DB      0
=4406 00                REV_VID         DB      0
=4407 0000              DRQ_ADRS DW     0       ; DATA REQUEST ADDRESS
=                       ;
=                       ;       Line Buffer for ROW move operations
=                       ;
=                       ;
=                       ;
=4409                   LINBUF  RB      160
```

```
 
 =
 =                    ; TABLE FOR LANGUAGES VEPSION 1
 =                    ;
 =44A9 00             LANG_T1 DB     00H           ; US / HEBREW
 =44AA 01                     DB     01H           ; UK
 =44AB 05                     DB     05H           ; DANSK
 =44AC 03                     DB     03H           ; GERMANY
 =44AD 04                     DB     04H           ; SWEDEN
 =44AE 05                     DB     05H           ; DANSK
 =44AF 06                     DB     06H           ; SPAIN
 =44B0 07                     DB     07H           ; ITALY
 =
 =                    ; TABLE FOR LANGUAGES VERSION 2
 =                    ;
 =44B1 08             LANG_T2 DB     08H           ; SWISS
 =44B2 08                     DB     08H           ; SWISS
 =44B3 02                     DB     02H           ; FRANCE
 =44B4 09                     DB     09H           ; CANADA1
 =44B5 0A                     DB     0AH           ; CANADA2
 =44B6 0B                     DB     0BH           ; SAFRICA
 =44B7 0C                     DB     0CH           ; PORTUGAL
 =44B8 0D                     DB     0DH           ; YUGOSLAVIA
 =
 =                    ;
 =                    ;
 =                    ; GRAPHIC MODE DATA
 =                    ;
 =                    ;
 =  0000              RESETCMD      EQU    00H    ;
 =  000D              STARTCMD      EQU    0DH    ;
 =  000C              STOPCMD       EQU    0CH    ;
 =  006F              MSTRCMD       EQU    6FH    ;
 =  006E              SLVCMD        EQU    6EH    ;
 =  006C              VECTECMD      EQU    6CH    ;
 =  0068              TEXTECMD      EQU    68H    ;
 =  0070              SCROLLCMD     EQU    70H    ;
 =  0020              GRWRTW        EQU    20H    ;
 =  0030              GRWRTL        EQU    30H    ;
 =  0038              GRWRTH        EQU    38H    ;
 =  00A0              GRREADW       EQU    0A0H   ;
 =  00B0              GRREADL       EQU    0B0H   ;
 =  00B8              GRREADH       EQU    0B8H   ;
 =  00E0              CSRRCMD       EQU    0E0H   ;
 =                    ;
 =  0028              GRPITCH       EQU    28H    ;
 =  00A1              GRCMD         EQU    0A1H    ;
 =  00A0              GRPARA        EQU    0A0H    ;
 =  00A0              GRSTATUS      EQU    0A0H    ;
 =  00A1              GRRDATA       EQU    0A1H    ;
 =                    ;
 =                    ;
 =                    ;
```

```

=  44B9            INITSCR             EQU OFFSET$
=44B9 70                               DB      70H     ;PRAM+0
=44BA 0000         GDC_SP1             DW      0
=44BC 00           GDC_LP11            DB      0
=44BD 00           GDC_LP12            DB      0
=44BE 0000         GDC_SP2             DW      0
=44C0 00           GDC_LP21            DB      0
=44C1 01           GDC_LP22            DB      1
=                  ;
=  44C2            ERROR_CUR_START     EQU OFFSET$
=44C2 49                               DB      49H     ;CURS
=44C3 803E                             DW      3E80H   ;WORD ADDRESS
=44C5 00                               DB      0       ;DOT ADDRESS
=                  ;
=  44C6            MASK_OUT            EQU OFFSET$
=44C6 4A                               DB      4AH     ;MASK
=44C7 FFFF                             DW      0FFFFH
=                  ;
=  44C9            FIGS_OUT            EQU OFFSET$
=44C9 4C                               DB      4CH     ;FIGS
=44CA 02                               DB      2       ;DIRECTION = 2
=44CB 4F00                             DW      80-1    ;LENGTH
=                  ;
=  44CD            WDAT_OUT            EQU OFFSET$
=44CD 20                               DB      20H     ;WDAT
=44CE 20                               DB      20H     ;SPACE CHARACTER
=44CF E0                               DB      0E0H    ;ATTRIBUTE
=                  ;
=44D0 70           ALPHA_PARTITION     DB      70H
=44D1 0000                             DW      0
=44D3 0019                             DW      1900H
=44D5 0000                             DW      0
=44D7 0000                             DW      0
=                  ;
=44D9 00           GRAPHIC_FLAG        DB      0
=



 =                              INCLUDE C:CRTPIMD.SEG
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;
 =                        ;*****************************************************************
 =                        ;*                                                              *
 =                        ;*                    EQUATES used by the CRT PIM               *
 =                        ;*                                                              *
 =                        ;*****************************************************************
 =                        ;
 =                        ; EQUATES to the CRT Paramater Block (CRTPB)
 =                        ;
 =  0000                  CPB_COL EQU     0       ; column
 =  0001                  CPB_ROW EQU     1       ; row
 =  0002                  CPB_ATTR EQU    2       ; attribute
 =  0003                  CPB_ESC EQU     3       ; PIM escape code
 =  0004                  CPB_FREQ EQU    4       ; Music frequency
 =  0004                  CPB_RES1 EQU    4       ; reserved
 =  0005                  CPB_FLEN EQU    5       ; Length of Music frequency
 =  0005                  CPB_RES2 EQU    5       ; reserved
 =                        ;
 =                        ;  General EQUATES
 =                        ;
 =  0018                  ROWS    EQU     24      ; Rows on the screen
 =  0050                  SCWID   EQU     80      ; Screen width
 =  0040                  CL_MASK EQU     040H    ; "Send Character" Mask
 =  0080                  ATTR_MASK EQU   80H     ; Set Attribute Bit of Escape Byte
 =  000F                  ESC_MASK  EQU   0FH     ; Mask to isolate Escape Code of Escape Byte
 =                        ;
 =                        ;               MACRO LIBRARY FOR NCR DM-5
 =                        ;               --------------------------------------------
 =                        ;
 =                        ;
```

```
5955
5956  =                         ;           ============================
5957  =                         ;
5958  =                         ;       READ
5959  =  00A0                   GDCSTA  EQU     0A0H            ;STATUS PORT
5960  =  00A1                   FIFO    EQU     0A1H            ;GDC FIFO PORT ADDR
5961  =                         ;
5962  =                         ;
5963  =                         ;       WRITE
5964  =  00A0                   GDCPAR  EQU     0A0H            ;PARAMETER INTO FIFO
5965  =  00A1                   GDCCOM  EQU     0A1H            ;COMMAND INTO FIFO
5966  =                         ;
5967  =                         ;
5968  =                         ;       ORGANISATION OF GRAPHIC RAM
5969  =                         ;
5970  =                         ;       576 X 400  PIXELS
5971  =                         ;
5972  =  1FFF                   GRAEND  EQU     1FFFH           ;END ADDRESS OF GRAPHIC RAM
5973  =  0048                   NRWAPL  EQU     72              ;NUMBER OF WORD ADDR PER LINE
5974  =  0024                   WPL     EQU     NRWAPL/2        ;WORDS / LINE
5975  =  000A                   LPC     EQU     10              ;LINES / CHARACTER
5976  =                         ;
5977  =                         ;       MEANING OF GDC STATUS BITS
5978  =                         ;
5979  =  0001                   DATRDY  EQU     01H             ;A BYTE IS AVAILABLE TO READ
5980  =  0002                   FIFULL  EQU     02H             ;FIFO IS FULL
5981  =  0004                   FIFEMP  EQU     04H             ;FIFO IS EMPTY
5982  =  0008                   DRWINP  EQU     08H             ;DRAWING IN PROCESS
5983  =  0010                   DMAEXC  EQU     10H             ;DMA DATA TRANSFER IN PROCESS
5984  =  0020                   VERETR  EQU     20H             ;VERTICAL RETRACE IN PROCESS
5985  =  0040                   HORETR  EQU     40H             ;HORIZONTAL RETRACE IN PROCESS
5986  =  0080                   LIPDET  EQU     80H             ;LIGHT PEN DETECT (ADDRESS VALID)
5987  =                         ;
5988  =                         ;
5989  =                         ;       COMMANDS
5990  =                         ;
5991  =  0000                   GDCRES  EQU     0               ;RESET - BLANK DISPLAY, IDLE MODE, INITIALIZE
5992  =  006E                   VSYNCS  EQU     06EH            ;SLAVE MODE
5993  =  006F                   VSYNCM  EQU     06FH            ;MASTER MODE
5994  =  004B                   CCHAR   EQU     04BH            ;CURSOR & CHARACTER CHARACTERISTICS
5995  =  006B                   START   EQU     06BH            ;START DISPLAY & END IDLE MODE
5996  =  0046                   ZOOM    EQU     046H            ;SPECIFY ZOOM FACTOR
5997  =  0049                   CURS    EQU     049H            ;SPECIFY CURSOR POSITION
5998  =  0047                   PITCH   EQU     047H            ;PITCH SPECIFICATION
5999  =  004A                   MASKREG EQU     04AH            ;LOAD MASK REGISTER
6000  =  004C                   FIGS    EQU     04CH            ;SPECIFY FIGURE DRAWING PARAMETER
6001  =  006C                   FIGD    EQU     06CH            ;START FIGURE DRAW
6002  =  0068                   GCHRD   EQU     068H            ;START GRAPHICS CHARACTER DRAW
6003  =  00E0                   CURD    EQU     0E0H            ;READ CURSOR ADDRESS
6004  =  00C0                   LPRD    EQU     0C0H            ;READ LIGHT PEN ADDRESS
6005  =                         ;
6006  =  0070                   PRAM    EQU     070H            ;LOAD PARAMETER RAM
6007  =  0000                   PRAMSA  EQU     0               ;LOWER 4 BITS ARE STARTING ADDRESS IN RAM
```

```
6008
6009  =                                              ;( COMMAND + SA )
6010  =                 ;
6011  =  0020           WDAT    EQU     020H         ;WRITE DATA INTO DISPLAY MEMORY
6012  =                                              ;( COMMAND + TYPE + MODE )
6013  =                                              ;DATA TRANSFER TYPES
6014  =  0000           TYWORD  EQU     0            ;WORD, LOW THEN HIGH BYTE
6015  =  0010           TYLOBY  EQU     010H         ;LOW BYTE OF THE WORD
6016  =  0018           TYHIBY  EQU     018H         ;HIGH BYTE OF THE WORD
6017  =                                              ;MODE OF RMW MEMORY CYCLE
6018  =  0000           MOREPL  EQU     0            ;REPLACE WITH PATTERN
6019  =  0001           MOCOMP  EQU     01H          ;COMPLEMENT
6020  =  0002           MORES   EQU     02H          ;RESET TO 0
6021  =  0003           MOSET   EQU     03H          ;SET TO 1
6022  =                 ;
6023  =  00A0           RDAT    EQU     0A0H         ;READ DATA FROM DISPLAY MEMORY
6024  =                                              ;( COMMAND + TYPE )
6025  =                                              ;TYPES AS AT WDAT
6026  =                 ;
6027  =  00A4           DMAR    EQU     0A4H         ;DMA READ REQUEST
6028  =                                              ;( COMMAND + TYPE )
6029  =                                              ;TYPES AS AT WDAT
6030  =                 ;
6031  =  0024           DMAW    EQU     024H         ;DMA WRITE REQUEST
6032  =                                              ;( COMMAND + TYPE + MODE )
6033  =                                              ;TYPES AND MODES AS AT WDAT
6034  =                 ;
6035  =                 ;       PARAMETERS
6036  =                 ;
6037  =                 ;       RESET
6038  =                 ;
6039  =  0000           RESMOP  EQU     0            ;MODE OF OPERATION SELECT BITS
6040  =                                              ;( RESMOP + DISPLAY + FRAME + DYNRAM + WINDOW )
6041  =                                              ;DISPLAY MODE
6042  =  0000           MIXGAC  EQU     0H           ;MIXED GRAPHICS & CHARACTER
6043  =  0002           GRAMOD  EQU     02H          ;GRAPHICS MODE
6044  =  0020           CHAMOD  EQU     020H         ;CHARACTER MODE
6045  =                                              ;VIDEO FRAMING
6046  =  0000           NOINTL  EQU     0            ;NON-INTERLACED
6047  =  0008           INLRPF  EQU     08H          ;INTERLACED REPEAT FIELD FOR CHARACTER DISPLAYS
6048  =  0009           INTLAC  EQU     09H          ;INTERLACED
6049  =                                              ;DYNAMIC RAM REFRESH CYCLES ENABLE
6050  =  0000           SATRM   EQU     0            ;NO REFRESH - STATIC RAM
6051  =  0004           DYNRAM  EQU     04H          ;REFRESH - DYNAMIC RAM
6052  =                                              ;DRAWING TIME WINDOW
6053  =  0000           DRWALL  EQU     0            ;DRAWING DURING ACTIVE DISPLAY TIME AND RETRACE
6054  =  0010           DRWRET  EQU     010H         ;DRAWING ONLY DURING RETRACE BLANKING
6055  =                 ;
6056  =                 ;
6057  =                 ;
6058  =                 ;*** CRT PERIPHERAL INTERFACE MODULE    DATA AREA
6059  =                 ;
6060  =                 ;
```

```

 =                      ;  CURSOR POSITION VARIABLES
 =                      ;
 =                      ;                 for performance reasons these bytes are sometimes loaded
 =                      ;                 in pairs!!
 =44DA 00               CURCOL   DB      0
 =44DB 00               CURROW   DB      0
 =44DC 00               ATTRIBUTE DB     0
 =44DD 00               OUTCHAR DB       0
 =                      ;
 =                      ;
 =                      ;  DEFINITION OF CRT PAGE VARIABLES
 =                      ;
 =44DE 0000             SP1      DW      0       ; START OF PAGE 1
 =44E0 00               LP11     DB      0       ; LENGTH OF PAGE1 LOW
 =44E1 00               LP12     DB      0       ; LENGTH OF PAGE1 HIGH
 =44E2 0000             SP2      DW      0       ; START OF PAGE 2
 =44E4 00               LP21     DB      0       ; LENGTH OF PAGE2 LOW
 =44E5 00               LP22     DB      0       ; LENGTH OF PAGE2 HIGH
 =                      ;
 =                      ;



 =                            INCLUDE C:DISKMGRD.SEG
 =
 =                     ;*****************************************************************
 =                     ;                                                                *
 =                     ;      DISK MANAGER    DATA SEGMENT                              *
 =                     ;                                                                *
 =                     ;*****************************************************************
 =                     ;
 =                     ;
 =                     ;
 =                     ;
 =                     ;
 =                     ;  DISK BUFFER, CHECK AND ALLOCATION VECTORS
 =                     ;
 =
 =                            IF NOT LOADER_BIOS
 =
 =  4C00               HSTBUF  EQU     4C00H
 =
 =                            ENDIF
 =
 =
 =                            IF LOADER_BIOS
 =
 =                     HSTBUF  EQU     3000H
 =
 =                            ENDIF
 =
 =
 =  5000               DIRBUF  EQU     HSTBUF+400H
 =  5080               ALV0    EQU     DIRBUF+128
 =  50A0               CSV0    EQU     ALV0+32
 =  50E0               ALV1    EQU     CSV0+64
 =  5131               CSV1    EQU     ALV1+81
 =  5171               ALV2    EQU     CSV1+64
 =  0000               CSV2    EQU     0
 =  51C2               ALV3    EQU     ALV2+81
 =  0000               CSV3    EQU     0
 =  5213               ALV4    EQU     ALV3+81
 =  0000               CSV4    EQU     0
 =  5264               ALV5    EQU     ALV4+81
 =  0000               CSV5    EQU     0
 =  52B5               ALV6    EQU     ALV5+81
 =  0000               CSV6    EQU     0
 =  5306               ALV7    EQU     ALV6+81
 =  0000               CSV7    EQU     0
 =  5357               ALV8    EQU     ALV7+81
 =  0000               CSV8    EQU     0
 =  53A8               ALV9    EQU     ALV8+81
 =  0000               CSV9    EQU     0
 =  53F9               ALV10   EQU     ALV9+81
```

```
6136
6137  =  0000               CSV10   EQU     0
6138  =  544A               ALV11   EQU     ALV10+81
6139  =  0000               CSV11   EQU     0
6140  =  549B               ALV12   EQU     ALV11+81
6141  =  0000               CSV12   EQU     0
6142  =  54EC               ALV13   EQU     ALV12+81
6143  =  0000               CSV13   EQU     0
6144  =  553D               ALV14   EQU     ALV13+81
6145  =  0000               CSV14   EQU     0
6146  =  558E               ALV15   EQU     ALV14+81
6147  =  0000               CSV15   EQU     0
6148  =                       ;
6149  =                       ;
6150  =                       ;  WRITE TYPES PASSED BY BDOS
6151  =                       ;
6152  =  0000               WRALL   EQU     0       ;WRITE TO ALLOCATED
6153  =  0001               WRDIR   EQU     1       ;WRITE TO DIRECTORY
6154  =  0002               WRUAL   EQU     2       ;WRITE TO UNALLOCATED
6155  =                       ;
6156  =                       ;
6157  =                       ;  MISC EQUATES
6158  =                       ;
6159  =  0000               UNA     EQU     BYTE PTR [BX]   ;NAME FOR BYTE AT BX
```

```

=                       ;*****************************************************************
=                       ;                                                                *
=                       ;  DISK PARAMETER BLOCKS                                         *
=                       ;                                                                *
=                       ;*****************************************************************
=                       ;
=  44E6                 DPBASE  EQU     $               ;BASE OF DISK PARAMETER BLOCKS
=44E6 00000000          DPE0    DW      0000H,0000H     ;TRANSLATE TABLE
=44EA 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=44EE 0050E645                  DW      DIRBUF,DPB0     ;DIR BUFF, PARM BLOCK
=44F2 A0508050                  DW      CSV0,ALV0       ;CHECK, ALLOC VECTORS
=                       ;
=44F6 00000000          DPE1    DW      0000H,0000H     ;TRANSLATE TABLE
=44FA 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=44FE 0050F545                  DW      DIRBUF,DPB1     ;DIR BUFF, PARM BLOCK
=4502 3151E050                  DW      CSV1,ALV1       ;CHECK, ALLOC VECTORS
=                       ;
=4506 00000000          DPE2    DW      0000H,0000H     ;TRANSLATE TABLE
=450A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=450E 00500446                  DW      DIRBUF,DPB2     ;DIR BUFF, PARM BLOCK
=4512 00007151                  DW      CSV2,ALV2       ;CHECK, ALLOC VECTORS
=                       ;
=4516 00000000          DPE3    DW      0000H,0000H     ;TRANSLATE TABLE
=451A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=451E 00501346                  DW      DIRBUF,DPB3     ;DIR BUFF, PARM BLOCK
=4522 0000C251                  DW      CSV3,ALV3       ;CHECK, ALLOC VECTORS
=                       ;
=4526 00000000          DPE4    DW      0000H,0000H     ;TRANSLATE TABLE
=452A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=452E 00502246                  DW      DIRBUF,DPB4     ;DIR BUFF, PARM BLOCK
=4532 00001352                  DW      CSV4,ALV4       ;CHECK, ALLOC VECTORS
=                       ;
=4536 00000000          DPE5    DW      0000H,0000H     ;TRANSLATE TABLE
=453A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=453E 00503146                  DW      DIRBUF,DPB5     ;DIR BUFF, PARM BLOCK
=4542 00006452                  DW      CSV5,ALV5       ;CHECK, ALLOC VECTORS
=                       ;
=4546 00000000          DPE6    DW      0000H,0000H     ;TRANSLATE TABLE
=454A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=454E 00504046                  DW      DIRBUF,DPB6     ;DIR BUFF, PARM BLOCK
=4552 00008552                  DW      CSV6,ALV6       ;CHECK, ALLOC VECTORS
=                       ;
=4556 00000000          DPE7    DW      0000H,0000H     ;TRANSLATE TABLE
=455A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=455E 00504F46                  DW      DIRBUF,DPB7     ;DIR BUFF, PARM BLOCK
=4562 00000653                  DW      CSV7,ALV7       ;CHECK, ALLOC VECTORS
=                       ;
=4566 00000000          DPE8    DW      0000H,0000H     ;TRANSLATE TABLE
=456A 00000000                  DW      0000H,0000H     ;SCRATCH AREA
=456E 00505E46                  DW      DIRBUF,DPB8     ;DIR BUFF, PARM BLOCK
=4572 00005753                  DW      CSV8,ALV8       ;CHECK, ALLOC VECTORS
```

```

=                          ;
=4576 00000000             DPE9    DW      0000H,0000H      ;TRANSLATE TABLE
=457A 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=457E 00506D46                     DW      DIRBUF,DPB9      ;DIR BUFF, PARM BLOCK
=4582 0000A853                     DW      CSV9,ALV9        ;CHECK, ALLOC VECTORS
=                          ;
=4586 00000000             DPE10   DW      0000H,0000H      ;TRANSLATE TABLE
=458A 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=458E 00507C46                     DW      DIRBUF,DPB10     ;DIR BUFF, PARM BLOCK
=4592 0000F953                     DW      CSV10,ALV10      ;CHECK, ALLOC VECTORS
=                          ;
=4596 00000000             DPE11   DW      0000H,0000H      ;TRANSLATE TABLE
=459A 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=459E 00508B46                     DW      DIRBUF,DPB11     ;DIR BUFF, PARM BLOCK
=45A2 00004A54                     DW      CSV11,ALV11      ;CHECK, ALLOC VECTORS
=                          ;
=45A6 00000000             DPE12   DW      0000H,0000H      ;TRANSLATE TABLE
=45AA 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=45AE 00509A46                     DW      DIRBUF,DPB12     ;DIR BUFF, PARM BLOCK
=45B2 00009B54                     DW      CSV12,ALV12      ;CHECK, ALLOC VECTORS
=                          ;
=45B6 00000000             DPE13   DW      0000H,0000H      ;TRANSLATE TABLE
=45BA 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=45BE 0050A946                     DW      DIRBUF,DPB13     ;DIR BUFF, PARM BLOCK
=45C2 0000EC54                     DW      CSV13,ALV13      ;CHECK, ALLOC VECTORS
=                          ;
=45C6 00000000             DPE14   DW      0000H,0000H      ;TRANSLATE TABLE
=45CA 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=45CE 0050B846                     DW      DIRBUF,DPB14     ;DIR BUFF, PARM BLOCK
=45D2 00003D55                     DW      CSV14,ALV14      ;CHECK, ALLOC VECTORS
=                          ;
=45D6 00000000             DPE15   DW      0000H,0000H      ;TRANSLATE TABLE
=45DA 00000000                     DW      0000H,0000H      ;SCRATCH AREA
=45DE 0050C746                     DW      DIRBUF,DPB15     ;DIR BUFF, PARM BLOCK
=45E2 00008E55                     DW      CSV15,ALV15      ;CHECK, ALLOC VECTORS
=                          ;
=45E6                      DPB0    RS      15               ;INITIALIZED FROM TYPE DEFINITION TABLE
=45F5                      DPB1    RS      15               ;     BY SELDSK
=4604                      DPB2    RS      15
=4613                      DPB3    RS      15
=4622                      DPB4    RS      15
=4631                      DPB5    RS      15
=4640                      DPB6    RS      15
=464F                      DPB7    RS      15
=465E                      DPB8    RS      15
=466D                      DPB9    RS      15
=467C                      DPB10   RS      15
=468B                      DPB11   RS      15
=469A                      DPB12   RS      15
=46A9                      DPB13   RS      15
=46B8                      DPB14   RS      15
=46C7                      DPB15   RS      15
```

```
6266
6267  =
6268  =                    ;*****************************************************************
6269  =                    ;                                                                 *
6270  =                    ;  TYPE DEFINITION TABLE                                          *
6271  =                    ;                                                                 *
6272  =                    ;    TYPE 0 - OTHER, FILLED BY EXCHANGE                           *
6273  =                    ;    TYPE 1 - NCR FORMAT DDSS                                     *
6274  =                    ;    TYPE 2 - NCR FORMAT DDDS                                     *
6275  =                    ;    TYPE 3 - NCR FORMAT WINCHESTER                               *
6276  =                    ;                                                                 *
6277  =                    ;      TYPE  =    0     1     2     3                             *
6278  =                    ;                                                                 *
6279  =                    ;*****************************************************************
6280  =                    ;
6281  =46D6 09080909       DSKSID  DB      009H, 008H, 009H, 009H  ;BIT 0 = 1, DOUBLE SIDED DISK
6282  =                                                            ;BIT 1 OR 2 = 1,CYLINDER MODE RECORDING
6283  =                                                            ;BIT 3 = 1, FIRST PHYSICAL SECTOR IS 1
6284  =                                                            ;BIT 6 = 1, SECOND SIDE OF DISK RECORDED
6285  =                                                            ;          IN REVERSE DIRECTION
6286  =                                                            ;BIT 7 = 1, DATA FORMAT IS COMPLEMENTED
6287  =46DA 01020202       DSKSLC  DB      001H, 002H, 002H, 002H  ;SECTOR LENGTH (1=256,2=512,3=1024)
6288  =46DE 01030303       DSKSMA  DB      001H, 003H, 003H, 003H  ;SECTOR MASK (FOR BLOCKING/DEBLOCKING)
6289  =46E2 10202044       DSKSPT  DB      010H, 020H, 020H, 044H  ;CPM SECTORS PER TRACK (LOW BYTE)
6290  =46E6 00000000       DSKSPH  DB      000H, 000H, 000H, 000H  ;CPM SECTORS PER TRACK (HI BYTE)
6291  =46EA 04040406       DSKBSH  DB      004H, 004H, 004H, 006H  ;DATA ALLOCATION BLOCK SHIFT FACTOR
6292  =46EE 0F0F0F3F       DSKBLM  DB      00FH, 00FH, 00FH, 03FH  ;BLOCK MASK
6293  =46F2 01000103       DSKEXT  DB      001H, 000H, 001H, 003H  ;EXTENT MASK
6294  =46F6 4C499987       DSKDSL  DB      04CH, 049H, 099H, 087H  ;DISK SIZE (LOW BYTE)
6295  =46FA 00000002       DSKDSH  DB      000H, 000H, 000H, 002H  ;DISK SIZE (HI BYTE)
6296  =46FE 7F7F7FFF       DSKMDL  DB      07FH, 07FH, 07FH, 0FFH  ;MAX NO OF DIRECTORY ENTRIES (LOW BYTE)
6297  =4702 00000001       DSKMDH  DB      000H, 000H, 000H, 001H  ;MAX NO OF DIRECTORY ENTRIES (HI BYTE)
6298  =4706 C0C0C0C0       DSKAL0  DB      0C0H, 0C0H, 0C0H, 0C0H  ;ALLOC0 FOR DIRECTORY
6299  =470A 00000000       DSKAL1  DB      000H, 000H, 000H, 000H  ;ALLOC1 FOR DIRECTORY
6300  =470E 20202000       DSKCSL  DB      020H, 020H, 020H, 000H  ;SIZE OF DIR CHECK VECTOR (LOW BYTE)
6301  =4712 00000000       DSKCSH  DB      000H, 000H, 000H, 000H  ;SIZE OF DIR CHECK VECTOR (HI BYTE)
6302  =4716 03030300       DSKOFL  DB      003H, 003H, 003H, 000H  ;NO OF RESERVED TRACKS (LOW BYTE)
6303  =471A 00000000       DSKOFH  DB      000H, 000H, 000H, 000H  ;NO OF RESERVED TRACKS (HI BYTE)
6304  =471E 00000000       DSKDBL  DB      000H, 000H, 000H, 000H  ;LOGICAL SECTOR LACING?
6305  =4722 10101040       DSKCNT  DB      010H, 010H, 010H, 040H  ;CPM SECTORS PER ALLOCATION BLOCK
6306  =4726 50285062       DSKTRK  DB      050H, 028H, 050H, 062H  ;NO OF TRACKS ON DISK (LOW BYTE)
6307  =472A 00000002       DSKTRH  DB      000H, 000H, 000H, 002H  ;NO OF TRACKS ON DISK (HI BYTE)
6308  =472E 08080811       DSKMSC  DB      008H, 008H, 008H, 011H  ;MAXIMUM SECTOR NUMBER
6309  =                    ;
6310  =                    ;
6311  =                    ;
6312  =                    ;*****************************************************************
6313  =                    ;                                                                 *
6314  =                    ;  DISK TYPE TABLE  -  TYPE IS FILLED IN WHEN DISK IS "SELECTED"   *
6315  =                    ;                                                                 *
6316  =                    ;*****************************************************************
6317  =                    ;
6318  =4732 FF             DSKTYP  DB      0FFH    ;DRIVE A

6319
6320  =4733 FF                     DB      0FFH    ;DRIVE B
6321  =4734 FF                     DB      0FFH    ;DRIVE C
6322  =4735 FF                     DB      0FFH    ;DRIVE D
6323  =4736 FF                     DB      0FFH    ;DRIVE E
6324  =4737 FF                     DB      0FFH    ;DRIVE F
6325  =4738 FF                     DB      0FFH    ;DRIVE G
6326  =4739 FF                     DB      0FFH    ;DRIVE H
6327  =473A FF                     DB      0FFH    ;DRIVE I
6328  =473B FF                     DB      0FFH    ;DRIVE J
6329  =473C FF                     DB      0FFH    ;DRIVE K
6330  =473D FF                     DB      0FFH    ;DRIVE L
6331  =473E FF                     DB      0FFH    ;DRIVE M
```

```
=473F FF                      DB      0FFH    ;DRIVE N
=4740 FF                      DB      0FFH    ;DRIVE O
=4741 FF                      DB      0FFH    ;DRIVE P
=                     ;
=                     ;
=                     ;  TRANSLATION TABLE FOR LOGICAL LACING
=                     ;
=  0020               LEN     EQU     32      ;LENGTH OF XLT
=  0000               VER     EQU     0       ;BIOS VERSION
=  0020               VERLEN  EQU     (VER*256)+LEN   ;VERSION AND LENGTH TOGETHER
=                     ;
=4742                 XLT     RS      LEN

=
=                     ;*********************************************************************
=                     ;                                                                   *
=                     ;  WORK AREA FOR BLOCKING/DEBLOCKING                                *
=                     ;                                                                   *
=                     ;*********************************************************************
=                     ;
=4762                 SEKDSK  RB      1       ;SEEK DISK NUMBER
=4763                 SEKTRK  RW      1       ;SEEK TRACK NUMBER
=4765                 SEKSEC  RB      1       ;SEEK SECTOR NUMBER
=                     ;
=4766                 HSTDSK  RB      1       ;HOST DISK NUMBER
=4767                 HSTTRK  RW      1       ;HOST TRACK NUMBER
=4769                 HSTSEC  RB      1       ;HOST SECTOR NUMBER
=                     ;
=476A                 SEKHST  RB      1       ;SEEK SHR SECSHF
=476B                 HSTACT  RB      1       ;HOST ACTIVE FLAG
=476C                 HSTWRT  RB      1       ;HOST WRITTEN FLAG
=                     ;
=476D                 UNACNT  RB      1       ;UNALLOC REC CNT
=476E                 UNADSK  RB      1       ;LAST UNALLOC DISK
=476F                 UNATRK  RW      1       ;LAST UNALLOC TRACK
=4771                 UNASEC  RB      1       ;LAST UNALLOC SECTOR
=                     ;
=4772                 ERFLAG  RB      1       ;ERROR REPORTING
=4773                 RSFLAG  RB      1       ;READ SECTOR FLAG
=4774                 READOP  RB      1       ;1 IF READ OPERATION
=4775                 WRTYPE  RB      1       ;WRITE OPERATION TYPE
=4776                 DMASEG  RW      1       ;DMA SEGMENT
=4778                 DMAOFF  RW      1       ;DMA OFFSET
=477A                 PMAADDR RW      1       ;PHYSICAL DMA SEGMENT
=477C                         RW      1       ;PHYSICAL DMA OFFSET
=                     ;
=                     ;
=                     ;
=                     ;*********************************************************************
=                     ;                                                                   *
=                     ;  ERROR MESSAGES                                                   *
=                     ;                                                                   *
=                     ;*********************************************************************
=                     ;
=477E 00              DISPFLAG        DB      0       ; 0 = DISPLAY ERROR MESSAGES
=                                                     ; FF = DO NOT DISPLAY ERROR MESSAGES
=                                                     ;       (USED BY SOME UTILITIES)
=477F 203A204E4F54    NOTRDY  DB      ' : NOT READY (R/X)'
      205245414459
      2028522F5829
=4791 FF                      DB      0FFH
=4792 203A20575249    PROTECT DB      ' : WRITE PROTECT (R/O/X)'
      54452050524F
      544543542028
      522F4F2F5829
```

```

=47AA FF                      DB     0FFH
=47AB 203A20464154   FATAL    DB     ' : FATAL ERROR (R/O/X)'
      414C20455252
      4F522028522F
      4F2F5829
=47C1 FF                      DB     0FFH
=47C2 203A20492F4F   IOERR    DB     ' : I/O ERROR (R/O/X)'
      204552524F52
      2028522F4F2F
      5829
=4706 FF                      DB     0FFH
=                    ;
=4707 1B3D3820FF     POSMSG   DB     1BH,3DH,38H,20H,0FFH    ;POSITION TO COLUMN 0, ROW 25
=47DC 1B3D3820       RESMSG   DB     1BH,3DH,38H,20H         ;POSITION TO COLUMN 0, ROW 25,
=47E0 1B54FF                  DB     1BH,54H,0FFH            ;   ERASE TO END OF LINE
```

```

=

=                           INCLUDE C:FLEXPIMD.SEG
=                   ;       TITLE   FLEX DISK DRIVER PIM (DATA SEGMENT)
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=
=
=
=                   ;
=                   ;
=                   ;
=                   ;##################
=                   ;###  I/O PORTS  ###
=                   ;##################
=                   ;
=                   ;
=                   ; FDC
=                   ; ---
=                   ;
=  0051             DCOMD   EQU     51H             ; DISK COMMAND PORT
=  0050             DSTAT   EQU     50H             ; DISK STATUS PORT
=  0051             FDCRA   EQU     51H             ; READ DMA 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
 =  0014            MOTORON EQU     14H         ; MOTOR ON PORT
 =                  ;
 =                  ;
 =                  ;
 =                  ; BANK SELECT
 =                  ; -----------
 =                  ;
 =  00E0            BANK    EQU     0E0H        ; BANK SELECT E0 :   0K -  64K
 =                                              ;             E1 :  64K - 128K
 =                                              ;             E2 : 128K - 196K
 =                                              ;             E3 : 196K - 256K
 =                  ;
 =                  ;
 =                  ;
 =
 =                  ;
 =                  ;
 =                  ;
 =                  ;**********************
 =                  ;***  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
 =                  ;
 =                  ;
 =                  ;
 =                  ;
 =                  ;***********************
 =                  ;***  FDC VARIABLES  ***
 =                  ;***********************
 =                  ;
 =                  ;
 =47E3 00           CYLMODE DB      00          ; 0 = CYLINDER MODE, 1 = not CYLINDER MODE
 =47E4 00           DRV     DB      00          ; DRIVE NUMBER
 =47E5 00           HEAD    DB      00          ; HEAD NUMBER
```

```
6519
6520  =47E6 00           TRACK   DB      00          ; TRACK NUMBER
6521  =47E7 00           SECTOR  DB      00          ; SECTOR NUMBER
6522  =                                              ;
6523  =47E8 0000         SECCNT  DW      0000        ; Number of sectors for I/O
6524  =                  ;
6525  =                  ;
6526  =47EA 00           COMSTR  DB      00          ; COMMAND STRING LENGTH
6527  =47EB 00                   DB      00          ; COMMAND STRING (max. 9 bytes)
6528  =47EC 00                   DB      00          ;
6529  =47ED 00                   DB      00          ;
6530  =47EE 00                   DB      00          ;
6531  =47EF 00                   DB      00          ;
6532  =47F0 00                   DB      00          ;
6533  =47F1 00                   DB      00          ;
6534  =47F2 00                   DB      00          ;
6535  =47F3 00                   DB      00          ;
6536  =                                              ;
6537  =47F4 00           ERRBUF  DB      00          ; STATUS BYTE 0
6538  =47F5 00                   DB      00          ; STATUS BYTE 1
6539  =47F6 00                   DB      00          ; STATUS BYTE 2
6540  =47F7 00                   DB      00          ; CYLINDER/TRACK
6541  =47F8 00                   DB      00          ; HEAD 0 or HEAD 1
6542  =47F9 00                   DB      00          ; SECTOR
6543  =47FA 00                   DB      00          ; SECTOR SIZE
6544  =                                              ;
6545  =                                              ;
6546  =                                              ;
6547  =
6548  =                  ;
6549  =                  ;
6550  =                  ;
6551  =                  ;********************
6552  =                  ;***  DMA COMMANDS  ***
6553  =                  ;********************
6554  =                  ;
6555  =                  ;
6556  =  0047            DMAWRT  EQU     47H         ; WRITE DMA COMMAND
6557  =  004B            DMAREAD EQU     4BH         ; READ DMA COMMAND
6558  =                  ;
6559  =                  ;
6560  =                  ;
6561  =                  ;
6562  =                  ;**********************
6563  =                  ;***  DMA VARIABLES  ***
6564  =                  ;**********************
6565  =                  ;
6566  =                  ;
6567  =47FB 0000         DMAADDR DW      0000        ; DMA ADDR OFFSET
6568  =47FD 0000                 DW      0000        ;          SEGMENT
6569  =                                              ;
6570  =47FF 0000         DMALENG DW      0000        ; DMA LENGTH
6571  =4801 00           DMAFUNC DB      00          ; DMA FUNCTION
```

```

 =                      ;
 =                      ;
 =                      ;
 =
 =                      ;
 =                      ;
 =                      ;
 =                      ;************************
 =                      ;***  DISK VARIABLES  ***
 =                      ;************************
 =                      ;
 =                      ;
 =4802 08               SECTRK  DB      08              ; SECTORS PER TRACK
 =4803 40               DENSITY DB      40H             ; DOUBLE DENSITY BIT (MFM)
 =4804 02               BYTSEC  DB      02              ; BYTES PER SECTOR (N): 00 - 128 bytes
 =                                                      ;                       01 - 256 bytes
 =                                                      ;                       02 - 512 bytes
 =                                                      ;                           .
 =                                                      ;                           .
 =                                                      ;                           .
 =4805 1B               GPL     DB      1BH             ; GAP LENGTH
 =                                                      ;
 =4806 F6               PATTERN DB      0F6H            ; FORMAT PATTERN
 =                                                      ;
 =4807 05               RETRIES DB      05              ; Number of retries
 =                      ;
 =                      ;
 =                      ;
 =
 =                      ;
 =                      ;
 =                      ;
 =4808 0000             SSB     DW      0000            ; Special Sector Buffer for BANK conflict
 =                                                      ;

```

```


                            IF NOT LOADER_BIOS
 =                          INCLUDE C:WIPIMD.SEG
 =
 =                  ;
 =                  ;
 =                  ;
 =                  ;*********************************************
 =                  ;*                                           *
 =                  ;*      PERIPHERAL INTERFACE MODULE (PIM)    *
 =                  ;*                                           *
 =                  ;*             WINCHESTER DISK               *
 =                  ;*                                           *
 =                  ;*********************************************
 =                  ;
 =                  ;UNIT 0= HEAD 0 AND 1
 =                  ;UNIT 1= HEAD 2 AND 3
 =                  ;
 =                  ;
 =                  ;          WINCHESTER DISK PARAMETER BLOCK
 =                  ;          ===============================
 =                  ;
 =                  ;
 =480A 00           WIPAR   DB      0               ; WIPAR + 0     DISK UNIT
 =480B 10                   DB      REST            ; WIPAR + 1     FUNCTION
 =480C 0000                 DW      0               ; WIPAR + 2     SECTOR LO
 =                                                  ; WIPAR + 3     SECTOR HI
 =480E 00                   DB      0               ; WIPAR + 4     STATUS 1
 =480F 00                   DB      0               ; WIPAR + 5     STATUS 2
 =4810 0000                 DW      0               ; WIPAR + 6     BUFFER ADDR.(SEGMENT)
 =4812 0000                 DW      0               ; WIPAR + 8     BUFFER ADDR.(OFFSET)
 =                  ;
 =                  ;
 =                  ;
 =                  ;          WINCHESTER DISK DEFINITIONS
 =                  ;          ===========================
 =                  ;
 =                  ;*********************************************
 =                  ;*                                           *
 =                  ;*      PORT DEFINITIONS                     *
 =                  ;*                                           *
 =                  ;*********************************************
 =                  ;
 =  00C0            HBASE   EQU     0C0H            ;     CONTROLLER BASE ADDR.
 =  00C0            DATA    EQU     HBASE           ; R/W DATA REGISTER
 =  00C1            WIERROR 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               ;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          WIREAD  EQU     20H             ;READ COMMAND
   =  0030          WIWRITE 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
   =                ;
   =                ;

                            ENDIF
```

```


=                               INCLUDE C:KBDMGRD.SEG
=
=                       ;************************************************************
=                       ;************************************************************
=                       ;**                                                        **
=                       ;**               KBD MANAGER DATA AREA                    **
=                       ;**                                                        **
=                       ;************************************************************
=                       ;************************************************************
=
=
=4814 00                FUNACT          DB      0
=
=4815 0000              FPOINTER        DW      0
=
=4817 0000              FCHARCNT        DW      0
=
=4819 00                FNCCHAR         DB      00H
=
=481A 0000              FNSTR           DW      00H
=
=481C 0000              FNACT           DW      00H
=
=481E 0000              RSTLEN          DW      00H
=
=4820 0000              FNEND           DW      00H
=
=4822 0000              FNLEN           DW      00H
=
=4824 00                FN_ERR          DB      00H
=
=4825 464E43542054      FN_ERR_MESS     DB      'FNCT TABLE FULL  (CR)'
      41424C452046
      554C4C202028
      435229
=483A FF                                DB      0FFH
=
=483B 00                HEBREW          DB      0
=
=
=



 =                          INCLUDE C:KBDPIMD.SEG
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;       ********************
 =                   ;       *  keyboard equates *
 =                   ;       ********************
 =                   ;
 =  0040                     keybase        equ    40h           ; no of controller
 =  0040                     wdkey          equ    keybase       ; output to keyboard
 =  0040                     rdkey          equ    keybase       ; input from keyboard
 =  0041                     rskey          equ    keybase+1     ; status addr of keyboard
 =  0041                     kbell          equ    keybase+1     ; addr for output a bell
 =  0041                     kcount         equ    keybase+1     ; kbd output of language number
 =                   ;
 =                   ;
 =                   ;
 =  0001                     country        equ    01h           ; command to get country code
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =                   ;
 =  0080                     lgdat86        equ    80h    ; flag for language byte ready
 =  0002                     inpbuff86      equ    02h    ; flag for output to kbd full
 =  0001                     kbdat86        equ    01h    ; flag for input from kbd ready
 =                   ;
 =                   ;
 =                   ;
 =483C 00                    language       db     00h    ; language code :
 =                                                        ; OLD KBD    NEW KBD I    NEW KBD II
 =                                                        ; 00 U.S.    10 U.S.      20 SWITZERLAND 1

 =                                                        ; 01 U.K.        11 U.K.      21 SWITZERLAND 2
 =                                                        ; 02 FRANCE      12 DENMARK   22 FRANCE
 =                                                        ; 03 GERMANY     13 GERMANY   23 CANADA
 =                                                        ; 04 SWED/FIN    14 SWED/FIN  24 SOUTH AFRICA
 =                                                        ; 05 NORW/DENM   15 NORWAY    25 PORTUGAL
 =                                                        ; 06 SPAIN       16 SPAIN     26 BRAZIL
 =                                                        ; 07 ITALY       17 ITALY     27 YUGOSLAVIA
 =483D F8                    kbd_var        db     0f8h   ; variante of keyboard

```

```


                                IF NOT LOADER_BIOS
=                               INCLUDE C:SERPIMD.SEG
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=
=                       ;
=                       ;***********************************************************************
=                       ;*                                                                     *
=                       ;*                   EQUATES used by the SER PIM                       *
=                       ;*                                                                     *
=                       ;***********************************************************************
=                       ;
=
=                       ;
=                       ;       PORT ADDRESSES FOR SERIAL IF RS232 (2651)
=                       ;
= 0060                          SPRDATA EQU     60H     ;READ DATA
= 0061                          SPRSTAT EQU     61H     ;READ STATUS
= 0063                          SPRCOM  EQU     63H     ;READ COMMAND
= 0064                          SPWDATA EQU     64H     ;WRITE DATA
= 0066                          SPWMODE EQU     66H     ;WRITE MODE
= 0067                          SPWCOM  EQU     67H     ;WRITE COMMAND
=                       ;
=                       ;       XON-XOFF VALUES
=                       ;
= 0011                          XON     EQU     11H
= 0013                          XOFF    EQU     13H
=                       ;
=                       ;       STATUS EQUATES FOR SERIAL IF RS232 (2651)...BIT MAPPED
=                       ;
= 0001                          TXRDY   EQU     01H     ;TRANSMIT HOLDING REGISTER EMPTY
= 0002                          RXRDY   EQU     02H     ;RECEIVE HOLDING REGISTER EMPTY
```

```

  = 0004              TXEMT   EQU     04H     ;CHANGE IN DSR OR DCD OR TRANSMIT
  = 0008              PARITY  EQU     08H     ;PARITY ERROR
  = 0010              OVERRUN EQU     10H     ;OVERRUN ERROR
  = 0020              FRAMING EQU     20H     ;FRAMING ERROR
  = 0040              DCD     EQU     40H     ;DATA CARRIER DETECT
  = 0080              DSR     EQU     80H     ;DATA SET READY
  =           ;
  =           ;*************************************************************
  =           ;*                                                           *
  =           ;*         VARIABLES TO BE PROVIDED BY THE USER              *
  =           ;*                                                           *

  =           ;*************************************************************
  =           ;
  =           ;       M1RS232 BYTE    BIT MAPPED : NUMBER OF STOP BITS
  =           ;                                    PARITY EVEN OR ODD
  =           ;                                    PARITY ENABLE OR DISABLE
  =           ;                                    BITS PER CHARACTER
  =           ;                                    ASYNC OR SYNC COMMUNICATION
  =           ;
  =           ;       M2RS232 BYTE    BIT MAPPED : INTERNAL OR EXTERNAL CLOCKS
  =           ;                                    BAUD RATE
  =           ;
  =           ;       PVRS232 BYTE    00H          PROTOKOL VECTOR (FOR FUTURE EXPANSION)
  =           ;                                    CURRENTLY 00H
  =           ;
  =           ;                                    THE SERIAL INTERFACE
  =           ;
  =           ;-------------------------------------------------------------------
  =           ;
  =           ;*******************************************************************
  =           ;*                                                                 *
  =           ;*                    INTERNAL VARIABLES                           *
  =           ;*                                                                 *
  =           ;*******************************************************************
  =           ;
  =483E 00    SACTIVE       DB      0       ;SERIAL I/F ACTIVE FLAG
  =483F 00    PACTIVE       DB      0       ;PARALLEL I/F ACTIVE FLAG
  =4840 00    XOFFFLG       DB      0       ;XOFF FLAG
  =           ;
  =           ;-------------------------------------------------------------------

```

```


=                           INCLUDE C:PARPIMO.SEG
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;
=                   ;.
=
=                   ;*********************************************************************
=                   ;*********************************************************************
=                   ;
=                   ;               PARALLEL INTERFACE (CENTRONICS)
=                   ;
=                   ;*********************************************************************
=                   ;*********************************************************************
=                   ;
=                   ;
=                   ;
=                   ;*********************************************************************
=                   ;*                                                                   *
=                   ;*                   EQUATES used by the PAR PIM                     *
=                   ;*                                                                   *
=                   ;*********************************************************************
=                   ;
=
=                   ;
=                   ;       PORT ADDRESSES FOR PARALLEL I/F (CENTRONICS)
=                   ;
=  0060                     PBDA    EQU     60H     ;DATA PORT
=  0061                     PBSTA   EQU     61H     ;STATUS PORT
=  0063                     PBCOM   EQU     63H     ;CONTROL PORT
=                   ;
=                   ;       STATUS EQUATES FOR PARALLEL I/F (CENTRONICS)
=                   ;
=  0020                     BUSY    EQU     20H     ;PRINTER BUSY
=  0002                     POBF    EQU     02H     ;OUTPUT BUFFER FULL

=                   ;
=                   ;-----------------------------------------------------------------------------
=



  03BF                      PATCHSIZE       EQU     4C00H - OFFSET $
4841                                        RS      PATCHSIZE
                                    ENDIF


  4C00                      ENDBIOS EQU     OFFSET $
                                    CSEG
                                    ORG     ENDBIOS
                            ;*****************************************************************
                            ;*                                                               *

                            ;*    MOVCPM  - ROUTINE TO SET UP INTERRUPT VECTORS AND MOVE THE O.S  *
                            ;*                                                               *
                            ;*****************************************************************
                            ;
                            ;    This routine is entered immediately upon a JMP 2500 (INIT) and is
                            ;       executed only at start-of-day.
                            ;    The code will be overlayed by a disk buffer.
                            ;    Entry parameters - CS is set up correctly (to 600H)
                            ;                       All other segment registers are unpredictable
                            ;                       (WHEN LOADED WITH DDT, SET CS=DS=ES=TPA+8)
                            ;
                            MOVCPM:

                                    IF NOT LOADER_BIOS

4C00 FC                             CLD                     ;SET FORWARD DIRECTION
4C01 2E803E7F2500                   CMP     DEBUG_FLG,0
4C07 7405           4C0E            JE      NO_SET_SEG
4C09 8CC8                           MOV     AX,CS
4C0B A3FB42                         MOV     WORD PTR PARA40+2,AX
                            NO_SET_SEG:
4C0E B80000                         MOV     AX,0
4C11 8ED8                           MOV     DS,AX           ;SET DS TO ZERO
4C13 BB06FE                         MOV     BX,MEMSIZ
4C16 8A1F                           MOV     BL,[BX]         ;GET MEMORY SIZE BYTE (=0-7)
4C18 8CC8                           MOV     AX,CS
4C1A 8ED8                           MOV     DS,AX           ;SET REAL DS VALUE
4C1C 8EC0                           MOV     ES,AX
4C1E B004                           MOV     AL,4
4C20 F6E3                           MUL     BL              ;MEMSIZ*4
4C22 98                             CBW
4C23 BEDC4C                         MOV     SI,OFFSET MEMTAB
4C26 03F0                           ADD     SI,AX
4C28 BF4443                         MOV     DI,OFFSET DISPMEM
4C2B B90400                         MOV     CX,4
4C2E F3A4                           REP MOVS AL,AL          ;MOVE ASCII MEMORY SIZE INTO SIGN ON MESSAGE
4C30 B104                           MOV     CL,4
4C32 8AC3                           MOV     AL,BL
4C34 D2E0                           SHL     AL,CL
4C36 08060143                       OR      BYTE PTR MRTLEN+1,AL;ADD MEMORY SIZE TO TPA LENGTH
```

```
4C3A BA4000                    MOV     DX,40H
4C3D 2E803E7F2500              CMP     DEBUG_FLG,0
4C43 7411         4C56         JE      SET_INT        ;FALL THRU IF SYSTEM WITH DDT86 LOADED
4C45 B80009                    MOV     AX,900H        ;SET SIZE FOR 2. OS + DDT86
4C48 29060043                  SUB     WORD PTR MRTLEN,AX  ;REDUCE TPA SIZE
4C4C 8CC8                      MOV     AX,CS
4C4E 050005                    ADD     AX,500H
4C51 A3FE42                    MOV     WORD PTR MRTLEN-2,AX  ;SET NEW TPA START
4C54 8CCA                      MOV     DX,CS
                               ;
                               ;SET UP ALL INTERRUPT VECTORS IN LOW MEMORY TO ADDRESS TRAP
                               ;
                       SET_INT:
4C56 B80000                    MOV     AX,0
4C59 8ED8                      MOV     DS,AX
4C5B 8EC0                      MOV     ES,AX          ;SET ES AND DS TO ZERO
                               ;
                               ;SETUP INTERRUPT 0 TO ADDRESS TRAP ROUTINE
                               ;
4C5D C70600002528              MOV     INT0_OFFSET,OFFSET INT_TRAP
4C63 89160200                  MOV     INT0_SEGMENT,DX
4C67 BF0400                    MOV     DI,4
4C6A BE0000                    MOV     SI,0           ;THEN PROPAGATE
4C6D B9FF00                    MOV     CX,255         ;TRAP VECTOR TO

4C70 2E803E7F2500              CMP     DEBUG_FLG,0
4C76 7428         4CA0         JE      ALL_LOOP
4C78 BF0000                    MOV     DI,0

                       DEBUG_LOOP:
4C7B 83C704                    ADD     DI,4
                       DEBUG_LOOPM:
4C7E BE0000                    MOV     SI,0
4C81 83E901                    SUB     CX,1
4C84 83FF04                    CMP     DI,4
4C87 74F2         4C7B         JE      DEBUG_LOOP     ;JUMP IF TRAP INTERRUPT
4C89 83FF0C                    CMP     DI,0CH
4C8C 74ED         4C7B         JE      DEBUG_LOOP     ;JUMP IF ONE BYTE INTERRUPT
4C8E 81FF8403                  CMP     DI,384H
4C92 74E7         4C7B         JE      DEBUG_LOOP     ;JUMP IF DDT86 BDOS CALL IR
4C94 83C101                    ADD     CX,1
4C97 A5                        MOVSW
4C98 A5                        MOVSW
4C99 E2E3         4C7E         LOOP    DEBUG_LOOPM
4C9B 8CC8                      MOV     AX,CS
4C9D E90700       4CA7         JMP     SET_BDOS

                       ALL_LOOP:
4CA0 A5                        MOVSW                  ;ALL 256 INTERRUPTS
4CA1 A5                        MOVSW
4CA2 E2FC         4CA0         LOOP    ALL_LOOP
```

```

4CA4 B84000                        MOV     AX,40H
                                   ;
                                   ;BDOS OFFSET TO PROPER INTERRUPT
                                   ;
                           SET_BDOS:
4CA7 C70678036020C                 MOV     BIOS_OFFSET,OFFSET BIOS_INT_ROUTINE
4CAD A37A03                        MOV     BIOS_SEGMENT,AX
4CB0 C70680030608                  MOV     BDOS_OFFSET,BDOS_OFST
4CB6 A38203                        MOV     BDOS_SEGMENT,AX
                                   ;
                                   ;NOW MOVE THE CCP, BDOS, AND BIOS TO ABSOLUTE PARAGRAPH 40H
                                   ;
4CB9 8ED8                          MOV     DS,AX
4CBB 2E803E7F2500                  CMP     DEBUG_FLG,0
4CC1 7515          4CD8            JNE     MOV_END         ;JUMP IF DEBUG FLAG SET

4CC3 B80006                        MOV     AX,600H         ; SOURCE IS PARAGRAPH 600
4CC6 8ED8                          MOV     DS,AX
4CC8 B84000                        MOV     AX,40H          ; DESTINATION IS PARAGRAPH 40
4CCB 8EC0                          MOV     ES,AX
4CCD BE0000                        MOV     SI,0
4CD0 BF0000                        MOV     DI,0
4CD3 B90028                        MOV     CX,2800H
4CD6 F3A5                          REP     MOVS AX,AX

                                   ENDIF

                           MOV_END:
4CD8 FF2EF942                      JMPF    BIOS40          ; NEXT INSTRUCTION IS RELATIVE TO PARAGRAPH 40!
                           ;
                           MEMTAB:                         ;TABLE OF ASCII MEMORY SIZES FOR SIGN ON MESS
4CDC 2036344B                      DB      ' 64K'          ;MEMSIZ = 0
4CE0 3132384B                      DB      '128K'          ;       = 1
4CE4 3139324B                      DB      '192K'          ;       = 2
4CE8 3235364B                      DB      '256K'          ;       = 3
4CEC 3332304B                      DB      '320K'          ;       = 4
4CF0 3338344B                      DB      '384K'          ;       = 5
4CF4 3434384B                      DB      '448K'          ;       = 6
4CF8 3531324B                      DB      '512K'          ;       = 7


                           ;*****************************************************************
                           ;*                                                               *
                           ;*         DUMMY DATA SECTION                                    *
                           ;*                                                               *
                           ;*****************************************************************
  0000                             DSEG    0               ;ABSOLUTE LOW MEMORY
                                   ORG     0               ;(INTERRUPT VECTORS)
0000                       INT0_OFFSET     RW      1
0002                       INT0_SEGMENT    RW      1
0004                       INT1_OFFSET     RW      1
0006                       INT1_SEGMENT    RW      1
                           ;PAD TO SYSTEM CALL VECTOR
0008                               RW      2*(BIOS_INT-2)
0378                       BIOS_OFFSET     RW      1
037A                       BIOS_SEGMENT    RW      1
037C                       UNUSED_OFFSET   RW      1
037E                       UNUSED_SEGMENT  RW      1
0380                       BDOS_OFFSET     RW      1
0382                       BDOS_SEGMENT    RW      1


```

```
ABC              0004 N  6688#
ALLLOOP          4CA0 L  7063  7084# 7087
ALLOC            3798 L  2853  2862  2867  2873  2893#
ALPHAPARTITION   4400 V  1447  5893#
ALV0             5080 N  6115# 6116  6172
ALV1             50E0 N  6117# 6118  6177
ALV10            53F9 N  6135# 6138  6223
ALV11            544A N  6138# 6140  6228
ALV12            549B N  6140# 6142  6233
ALV13            54EC N  6142# 6144  6238
ALV14            553D N  6144# 6146  6243
ALV15            558E N  6146# 6248
ALV2             5171 N  6119# 6121  6182
ALV3             51C2 N  6121# 6123  6187
ALV4             5213 N  6123# 6125  6192
ALV5             5264 N  6125# 6127  6197
ALV6             5285 N  6127# 6129  6202
ALV7             5306 N  6129# 6131  6207
ALV8             5357 N  6131# 6133  6212
ALV9             53A8 N  6133# 6135  6218
ATTRIBUTE        44DC V   453  1535  1556  1590  1740  2082  6068#
ATTRMASK         0080 N  1061  1200  1532  5948#
AUTOLOAD         2B40 L   469   519#  581
BANK             00E0 N  4642  6483#
BBD              0080 N  6692#
BDOSINT          00E0 N    56#   66#
BDOSOFFSET       0380 V   481  7096  7147#
BDOSOFST         0B06 N    55#   65#  481  7096
BDOSSEGMENT      0382 V   482  7097  7148#
BIOS40           42F9 V  5713# 7117
BIOSCODE         2500 N    57#   67#   73
BIOSINT          000E N    60# 7142
BIOSINTRET       2C8C L   800   804   808   810#
BIOSINTRET1      2C8E L   785   814#
BIOSINTROUTINE   2C60 L   783# 7094
BIOSOFFSET       0378 V  7094  7143#
BIOSSEGMENT      037A V  7095  7144#
BLANKONE         33B0 L  2058  2079# 2093  2113
BLINKING         0002 N  1181  5798#
BMPCR1           3347 L  1561  1595  2004  2006#
BMPCR2           334F L  2008  2010#
BUMPCUR          3338 L  2000#
BUSY             0020 N  5671  6975#
BYTSEC           4804 V  2575  3184  3624  3652  3741  4265  4334  6587#
CANADA1          2A60 L   358#
CANADA2          2A61 V   362#
CBUSY            0080 N  4731  4782  6706#
CCHAR            0048 N  1915  1930  5994#
CCP              0000 L    72#  467   470   484   583
CCPOFFSET        0000 N    54#   64#   71
CDRQ             0008 N  6702#
CERR             0001 N  6700#
CHAMOD           0020 N  6044#
CHANCHAR         40F1 L  5078  5099#
CHANCHAREND      412B L  5103  5123#
CHANEND          4131 L  5101  5130#
CHARREADY        4012 L  4879  4883  4887#
CHKUNA           374A L  2834  2848#
CHRTRAN          2EB9 L   874  1228#
CINIT            2B58 L   451   533#
CLEOS            3416 L  1608  2148#
CLEOS1           3438 L  2151  2163#
CLMASK           0040 N  1549  5947#
CLOSE            3581 L  2394  2427  2443# 2513  3429
CLRLIN           3472 L  2202# 2246  2288
CMDBUF           259C V   159#  521   524
COAD             0026 N  4631  4633  6467#
```

```
COLFULLI        2DB1 L  1068  1072#
COLHALFI        2D9A L  1053  1057#
COLOURHALFI     0005 N  1058  5800#
COLOURINDEX     4405 V   540  1052  1067  1147  1188  5802#
COMLEN          000A N   525  5723#
COMMD           00C7 N  4734  6663#
COMPLOOP        386A L  3003# 3007
COMSTR          47EA V  3926  3927  3929  3999  4000  4005  4007  4067  4070  4075
                        4128  4129  4245  4253  4258  4260  4262  4264  4266  4268
                        4270  4271  4326  4328  4333  4335  4337  4338  4340  4392
                        4393  6526#
CONFIGFL        258F V   148#  803  4938
CONFVER         2592 V   152#
CONIN           2B0A L    80   608# 1334  1384
CONLAC          3977 L  3156  3159#
CONOUT          2BCB L    81   597#  748
CONST           2BBC L    79   587#
CORRD           0004 N  6701#
COTC            0027 N  4637  4639  6468#
COUNTRY         0001 N  5250  6803#
CPBATTR         0002 N  1054  1058  1069  1073  1146  1154  1161  1168  1171  1175
                        1181  1190  1309  1312  1534  5936#
CPBCOL          0000 N   980   985   995  1009  1015  1016  1018  1094  1527  1568
                        1582  1599  5934#
CPBESC          0003 N  1310  1529  1549  5937#
CPBFLEN         0005 N  1213  2049  5940#
CPBFREQ         0004 N  1209  2047  5938#
CPBRES1         0004 N  5939#
CPBRES2         0005 N  5941#
CPBROW          0001 N   982  1001  1003  1086  5935#
CR              000D N   185   190   195   200   205   210   216   221   232  5721#
                        5725  5725  5731  5739  5742  5745  5756  5762
CRCID           0020 N  6690#
CRTATTR         2595 V   156#  452
CRTEXIT         3098 L  1550  1560  1562#
CRTMGR          2C8F L   601   602   656   658   677   865#
CRTMGREND       2CAE L   873   877#
CRTPB           43FE V   875   980   982   985   995  1001  1003  1009  1015  1016
                        1018  1054  1058  1069  1073  1086  1094  1146  1154  1161

                        1168  1171  1175  1181  1190  1209  1213  1308  1326  1342
                        5783#
CRTPIM          3041 L  1311  1519#
CRTTABLE        29CE L   132   299# 1266
CS              SREG V   440   482   492   788   793  5410  7013  7020  7042  7045
                        7081
CSRRCMD         00E0 N  5854#
CSV0            50A0 N  6116# 6117  6172
CSV1            5131 N  6118# 6119  6177
CSV10           0000 N  6137# 6223
CSV11           0000 N  6139# 6228
CSV12           0000 N  6141# 6233
CSV13           0000 N  6143# 6238
CSV14           0000 N  6145# 6243
CSV15           0000 N  6147# 6248
CSV2            0000 N  6120# 6182
CSV3            0000 N  6122# 6187
CSV4            0000 N  6124# 6192
CSV5            0000 N  6126# 6197
CSV6            0000 N  6128# 6202
CSV7            0000 N  6130# 6207
CSV8            0000 N  6132# 6212
CSV9            0000 N  6134# 6218
CTLTRANS        2CE1 L   906   912#
CURCOL          440A V  1528  1552  1558  1559  1567  1583  1586  1592  1593  1598
                        2001  2003  2011  2022  2121  2124  2130  2170  2247  2251
                        2289  6066#
```

```
CURD           00E0 N  6003#
CUROFF         329F L  1911# 2062  2095  2159  2190  2236  2253  2279
CURON          3284 L  1926# 2077  2111  2162  2197  2242  2264  2285
CURROW         44D8 V  2007  2012  2059  2138  2150  2152  2177  2181  2228  2273
                       6067#
CURS           0049 N  1781  5997#
CURSOR         259B V   158# 1940
CYLHI          00C5 N  4728  6659#
CYLLO          00C4 N  4676  4679  4725  6658#
CYLMOD         39FC L  3191  3210#
CYLMODE        47E3 V  2569  3173  3211  3604  3830  4247  6516#
DAMNFD         0001 N  6686#
DAMSK          2A14 V   332#
DATA           00C0 N  4763  4771  4806  6653#
DATASEG        42F9 N  5687# 5689
DATE           2579 V   127#
DATRDY         0001 N  1746  5979#
DCD            0040 N  6886#
DCOMD          0051 N  4398  6455#
DDSS           3663 L  2588  2591#
DEBUGFLG       257F V   128# 7011  7038  7062  7102
DEBUGLOOP      4C78 L  7066# 7072  7074  7076
DEBUGLOOPM     4C7E L  7068# 7080
DECSIGN1       2AC4 V   410# 5285  5299
DECSIGN2       2AD4 V   426# 5286  5300
DELAY          302F L  1451  1459#

DELAY1         3032 L  1461# 1464  1469
DELAY2         3038 L  1465# 1468
DELCHR         33C0 L  1613  2089#
DELCHR1        33C9 L  2096# 2110
DENSITY        4803 V  4069  4246  4327  6586#
DFORMAT        3E06 L  3389  4181#
DHOME          369F L  2525  2625# 2634  2648
DIRBUF         5000 N  6114# 6115  6171  6176  6181  6186  6191  6196  6201  6206
                       6211  6217  6222  6227  6232  6237  6242  6247
DISKINIT       3548 L   462   477  2386#
DISKWBOOT      3565 L   578  2423#
DISP           3AAE L  3316# 3323
DISPERR        3A9E L  2632  3241  3255  3298  3309#
DISPFLAG       477E V  3310  3465  3467  3472  3474  6386#
DISPMEM        4344 V  5739# 7028
DMA            3EFA L  4273  4342  4620#
DMAADDR        47FB V  2567  2568  3170  3172  3617  3622  3669  3675  3689  3691
                       3695  3697  3707  3710  3721  3824  4625  4630  4645  6567#
DMAEXC         0010 N  5983#
DMAFUNC        4801 V  3588  3592  4183  4622  6571#
DMALENG        47FF V  3659  3671  3722  3749  3819  4188  4635  6570#
DMAMB          002A N  4405  4655  6464#
DMAMO          002B N  4623  6465#
DMAOFF         4778 V  2722  2974  6374#
DMAR           00A4 N  6027#
DMAREAD        0048 N  3592  4183  6557#
DMASEG         4776 V  2747  2977  2988  6373#
DMAW           0024 N  6031#
DMAWRT         0047 N  3588  6556#
DOESC          3061 L  1533  1536#
DOOUTCHAR      3079 L  1531  1551#
DOPIMESC       2F1E L   988  1025  1040  1215  1305#
DOPIMESCEND    2F34 L  1307  1313#
DPB0           45E6 V  2608  2609  2614  6171  6250#
DPB1           45F5 V  6176  6251#
DPB10          467C V  6222  6260#
DPB11          468B V  6227  6261#
DPB12          469A V  6232  6262#
DPB13          46A9 V  6237  6263#
DPB14          46B8 V  6242  6264#
DPB15          46C7 V  6247  6265#
```

```
DPB2            4604 V  6181  6252#
DPB3            4613 V  6186  6253#
DPB4            4622 V  6191  6254#
DPB5            4631 V  6196  6255#
DPB6            4640 V  6201  6256#
DPB7            464F V  6206  6257#
DPB8            465E V  6211  6258#
DPB9            466D V  6217  6259#
DPBASE          44E6 L  2535  6168#
DPBMOV          3698 L  2617# 2621
DPE0            44E6 V  6169#
DPE1            44F6 V  6174#
DPE10           4586 V  6220#
DPE11           4596 V  6225#
DPE12           45A6 V  6230#
DPE13           45B6 V  6235#
DPE14           45C6 V  6240#
DPE15           45D6 V  6245#
DPE2            4506 V  6179#
DPE3            4516 V  6184#
DPE4            4526 V  6189#
DPE5            4536 V  6194#
DPE6            4546 V  6199#
DPE7            4556 V  6204#
DPE8            4566 V  6209#
DPE9            4576 V  6215#
DREAD           3BA5 L  2576  3044  3410  3586#
DREADID         3D02 L  2562  4065#
DREADY          0040 N  6705#
DREST           3D7B L  2558  3460  3800  3921#
DREST1          3D7D L  3924# 3947
DREST2          3D91 L  3934# 3937
DREST3          3DA6 L  3944  3948#
DRQADRS         4407 V   887  1079  1088  1138  1206  1210  1221  5078  5804#
DRQFLG          0001 N  1140  5079  5124  5154  5790#
DRV             47E4 V  2556  3177  3180  3192  3459  3928  4004  4074  4257  4332
                        6517#
DRWALL          0000 N  6053#
DRWINP          0008 N  5982#
DRWRET          0010 N  6054#
DS              SREG V   442   478   480   483   493   787   789   811  1244  1260
                        2568  2976  2986  2988  2992  2993  3001  3009  3036  3090
                        3674  3676  3683  3696  4759  4761  4773  4801  4803  4809
                        7017  7021  7051  7101  7106
DSEEC           0010 N  6703#
DSEEK           3DA7 L  3997# 4243  4324
DSEEK1          3DC8 L  4010# 4013
DSERNUM         439A V   456  5755#
DSIS            3DF5 L  3939  4015  4126#
DSKAL0          4706 V  6298#
DSKAL1          470A V  6299#
DSKBLM          46EE V  3502  6292#
DSKBSH          46EA V  6291#
DSKCNT          4722 V  2840  6305#
DSKCSH          4712 V  6301#
DSKCSL          470E V  6300#
DSKDBL          471E V  3149  6304#
DSKDSH          46FA V  6295#
DSKDSL          46F6 V  6294#
DSKEXT          46F2 V  6293#
DSKMDH          4702 V  6297#
DSKMDL          46FE V  6296#
DSKMSC          472E V  3185  6308#
DSKOFH          471A V  6303#
DSKOFL          4716 V  6302#
```

```
DSKSID          46D6 V  2997  3187  3202  3505  6281#
DSKSLC          460A V  2914  3183  6287#
DSKSMA          46DE V  2965  6288#
DSKSPH          46E6 V  6290#
DSKSPT          46E2 V  2606  2615  2882  6289#
DSKTRH          472A V  6307#
DSKTRK          4726 V  3195  6306#
DSKTYP          4732 V  2437  2604  2839  2881  2913  2964  2996  3148  3182  3194
                        6318#
DSPACH0         2C35 L   589   599   610   627   654   715#
DSPACH6         2C31 L   504   675   695   712#
DSR             0080 N  6887#
DSTAT           0050 N  4509  6456#
DWRFA           0020 N  6704#
DWRITE          3BAF L  3098  3590#
DYNRAM          0004 N  6051#
ENDBIOS         4C00 N  6989# 6991
ENDFLUSH        35F3 L  2545  2547#
ENDHOME         360F L  2635  2653#
ER1             3FE1 L  4787  4790#
ERFLAG          4772 V  2910  3014  3022  3082  3135  6369#
ERRBUF          47F4 V  2574  3224  3228  3232  3772  3781  3786  3791  3943  4454
                        6537#
ERRDISP         2F35 L  1324# 3317  5162
ERRDISP1        2FCD L  1392# 3328  5163
ERRDISPEND      2FE4 L  1394  1407#
ERREND          3AC0 L  3319  3321  3326#
ERRORCURSTART   44C2 N  1362  1375  5874#
ERRRET          3AC5 L  3311  3330#
ES              SREG V   443   534   536   539   543   550   563  2975  2977  2987
                        3000  3010  3677  3678  3708  7022  7052  7108
ESCFLG          0002 N   880  1131  5792#
ESCMASK         000F N  1537  5949#
ESCTABLE        3003 L  1541  1605#
ESCTRANS        2D02 L   892   939#
FALSE           0000 N    48#   50
FATAL           47AB V  3237  3295  6399#
FCHARCNT        4817 V  4951  4994  6738#
FDCRA           0051 N  4456  6457#
FDCRDY          3EE1 L  4396  4402  4459  4508# 4511
FDCSIS          0008 N  4129  6503#
FIFEMP          0004 N  5981#
FIFO            00A1 N  1748  5960#
FIFULL          0002 N  1641  1646  1651  1665  1670  1675  1680  1707  1712  1717
                        1722  1727  1732  1738  1759  1779  1784  1789  1795  1806
                        1811  1816  1830  1835  1840  1846  1851  1872  1877  1882
                        1887  1892  1901  1913  1918  1928  1933  1938  1943  1959
                        1964  1987  5980#
FIGD            006C N  6001#
FIGS            004C N  1667  1709  1832  1874  1961  6000#
FIGSOUT         44C9 N  1369  5883#
FILHST          37EC L  2923  2943  2945#
FILHSTL         3808 L  2956  2958#

FILLER          002B N   114#  119#  123
FIXCON          3940 L  3116  3118  3124#
FIXCONT         38DC L  3062  3064  3070#
FIXD3           3FA4 L  4732  4743# 4774  4811
FIXDISP         3A9A L  3294  3296#
FIXDR           3F48 L  2397  2645  3058  3067  3112  3121  3395  4699#
FIXEND          3A9D L  3290  3299#
FIXERR          3A84 L  2646  3071  3125  3283# 3396
FIXREAD         38AE L  3042  3047#
FIXREADY        3F38 L  2395  2629  3252  4674#
FIXREADY1       3F4A L  4681  4684#
FIXRETRY        38B7 L  3054# 3069
FIXRTRY         391B L  3108# 3123
FIXWRITE        3912 L  3096  3101#
```

```
FIXWRTRK     3809 L  3386  3392#
FLEXDISP     3A3A L  3231  3235  3238  3240#
FLEXEND      3A3D L  3227  3242#
FLEXERR      3A13 L  2559  2563  2577  3045  3099  3223# 3390  3411
FLUSH        35E9 L  2512  2542# 3428
FNACT        481C V  5063  5119  5121  5146  6744#
FNCCHAR      4819 V  5058  5100  6740#
FNEND        4820 V  5076  5106  5108  5111  6748#
FNERR        4824 V  1222  5102  5131  5143  6752#
FNERRMESS    4825 V  5161  6754#
FNLEN        4822 V  5077  5105  5133  6750#
FNSTR        481A V  5062  5073  5134  5145  5149  6742#
FORMAT       0050 N  4740  6677#
FPOINTER     4815 V  4947  4950  4995  6736#
FRAMING      0020 N  5551  6885#
FRANCE       2909 V   313#
FUN1         25BD V   184#  187
FUN10        261F V   231#  234
FUN11        2628 V   236#  240
FUN12        2639 V   242#  244
FUN13        263E V   246#  248
FUN14        2643 V   250#  252
FUN15        2648 V   254#  256
FUN16        264D V   258#  260
FUN17        2652 V   262#  264
FUN18        2657 V   266#  269
FUN19        265C V   271#  273
FUN2         25C6 V   189#  192
FUN20        2661 V   275#  277
FUN21        2666 V   279#
FUN3         25CF V   194#  197
FUN4         25DA V   199#  202
FUN5         25E3 V   204#  207
FUN6         25EC V   209#  212
FUN7         25F7 V   214#  218
FUN8         2601 V   220#  223
FUN9         260E V   225#  229
FUNACT       4814 V  4878  4916  4943  4953  4996  6734#
FUNC0        2C7C L   796   798#
FUNC1        2C82 L   796   802#
FUNC2        2C89 L   796   806#
FUNCHECK     403F L  4927  4931  4937#
FUNCTAB      2C76 V   793   796#
FUNCTABLE    25BD L   131   180#  281  5022
FUNEND       29CD V   286# 5107
FUNERR       4144 L  5109  5142#
FUNERRDISP   416A L  5132  5160#
FUNFILL      0366 N   281#  282
FUNSET       407A L  4940  4987#
FUNSETEND    4093 L  4990  4998#
FWMESS1      43A1 V   559  5757#
FWMESS2      43B3 V   547  5760#
FWMOVE       2B7B L   549#  555
FWVERSION    0FF7 N   538   542   548  5764#
GCHRD        0068 N  6002#
GDCCOM       00A1 N  1644  1668  1683  1710  1730  1782  1809  1833  1854  1875
                     1895  1916  1931  1962  1990  5965#
GDCLP11      44BC V  5868#
GDCLP12      44BD V  1357  1401  5869#
GDCLP21      44C0 V  5871#
GDCLP22      44C1 V  5872#
GDCPAR       00A0 N  1649  1654  1673  1678  1715  1720  1725  1735  1741  1762
                     1787  1793  1798  1814  1819  1838  1844  1849  1880  1885
                     1890  1904  1921  1936  1941  1946  1967  5964#
GDCRES       0000 N  5991#
GDCSP1       44BA V  5867#
GDCSP2       44BE V  1358  5870#
```

```
GDCSTA         00A0 N  1640  1645  1650  1664  1669  1674  1679  1706  1711  1716
                       1721  1726  1731  1736  1745  1758  1778  1783  1788  1794
                       1805  1810  1815  1829  1834  1839  1845  1850  1871  1876
                       1881  1886  1891  1900  1912  1917  1927  1932  1937  1942
                       1958  1963  1986  5959#
GERMANY        29EE V   320#
GETBYT         3ED1 L  3769  4078  4132  4192  4453#
GETBYT1        3ED4 L  4455# 4461
GETCRT         2EFD L  1269# 1273
GETDPH         3506 L  2505  2517  2530#
GETFCHAR       40AB L  1221  5057#
GETFPOS        4094 L  4993  5021# 5061  5067
GETFUN         409C L  5026# 5029
GETIOBF        2C3F L    96   727#
GETSEGT        2C4A L    95   736#
GETTYP         35F4 L  2514  2553# 2561  2565
GETX           2D06 L  1088  1090#
GETX1          2DE2 L  1093  1095#
GETY           2DC1 L  1079  1082#
GETY1          2DCD L  1085  1087#
GFPEND         40AA L  5037#
GO1            2B1F L   464   466   468#
GO2            2B89 L   580   582#
GPL            4805 V  4269  6593#
GRAEND         1FFF N  5972#
GRAMOD         0002 N  6043#
GRAPHIC        2F65 L  1329  1352#
GRAPHICFLAG    4409 V   573   575   799   872  1306  1328  1354  1388  1393  5899#
GRCMD          00A1 N  1417  1432  1438  5857#
GRFXOFF        3013 L   576  1445#
GRFXOFF1       3025 L  1452# 1455
GRGDCC1        300C L  1415  1425  1430  1436  1440# 1443
GRMOUT         2FE5 L  1361  1364  1367  1371  1374  1377  1380  1404  1414# 1449
GRMOUT010      2FF1 L  1420# 1426
GRMOUTRET      2FFB L  1419  1427#
GRPARA         00A0 N  1424  5858#
GRPITCH        0028 N  5856#
GRRDATA        00A1 N  5860#
GRREADH        0088 N  5853#
GRREADL        0080 N  5852#
GRREADW        00A0 N  5851#
GRSTART        2FFC L  1429# 1456
GRSTATUS       00A0 N  1441  1453  1462  1466  5859#
GRSTOP         3004 L  1435# 1446
GRWRTH         0038 N  5850#
GRWRTL         0030 N  5849#
GRWRTW         0020 N  5848#
H1             3087 L  1587  1589#
H2             30CC L  1594  1596#
HALFINTENSITY  0004 N  1054  5794#
HARDDISK       35CA L  2509  2518#
HBASE          00C0 N  6652# 6653  6654  6655  6656  6657  6658  6659  6660  6662
                       6663
HEAD           47E5 V  2557  2570  3175  3200  3216  3605  3847  3858  3867  3871
                       4001  4071  4254  4261  4329  6518#
HEBREW         4838 V  1229  4959  4963  6760#
HEBREWOFF      4073 L  4923  4962#
HEBREWON       406C L  4921  4958#
HIPOUT         30A3 L   876  1581#
HOME           358D L    85  2468#
HOMED          3599 L  2471  2473#
HORETR         0040 N  5985#
HSTACT         4768 V  2448  2472  2921  6361#
HSTBUF         4C00 N  2567  2581  2972  3037  3091  6102# 6109# 6114
HSTDSK         4766 V  2555  2602  2637  2927  2948  2962  2994  3040  3094  3146
                       3176  3264  3312  3383  3458  6356#
HSTSEC         4769 V  2937  2953  3145  3151  3269  3452  6358#
```

```
HSTTRK          4767 V  2931  2950  3178  3268  3447  6357#
HSTWRT          476C V  2449  2469  2543  2941  2959  2984  3020  6362#
ICLEOL          33F5 L  1609  2129# 2164
ICLEOLRET       3415 L  2134  2144#
IDNFD           0010 N  6689#
IDREAD          000A N  4068  6504#
IHOME           343C L  2169# 2196
ILF             3446 L  1617  2176#
ILF1            3456 L  2180  2184#
INIT            2ADA L    77   435#
INIT10          320F L  1952# 1981  2191
INIT40          2ADD L   439# 5711
INITDPB         3670 L  2516  2524  2601# 3440
INITEND         3562 L  2396  2401#
INITSCR         4489 N  1359  1402  5865#
INITTYP         3572 L  2390  2431#
INLRPF          0008 N  6047#
INPAR           31A8 L  1685  1687  1744# 1747  1858
INPBUFF86       0002 N  5472  6810#
INSCH1          3395 L  2063# 2076
INSCHR          3384 L  1612  2056#
INT0OFFSET      0000 V  7056  7137#
INT0SEGMENT     0002 V  7057  7138#
INT1OFFSET      0004 V  7139#
INT1SEGMENT     0006 V  7140#
INTLAC          0009 N  6048#
INTTRAP         2B25 L   490# 7056
INTTRP          4302 V   494  5725#
INVERSE         0001 N  1190  5796#
I0              3CB5 L  3699  3751  3762#
I01             3B86 L  3589  3593# 3838  3845  3849  3856  3868  3873
I010            3C69 L  3701  3705#
I011            3C8D L  3716  3731#
I015            3C93 L  3638  3644  3737#
I016            3CA5 L  3744  3747#
I017            3CB1 L  3753  3756#
I02             3B8E L  3595  3597#
I020            3CB8 L  3765# 3802
I021            3CCC L  3774  3780#
I022            3CD5 L  3782  3785#
I023            3CDE L  3787  3790#
I024            3CE7 L  3792  3795#
I025            3CF3 L  3797  3806#
I03             3BD7 L  3606  3608#
I030            3CF5 L  3733  3758  3811#
I031            3CFD L  3816  3818#
I032            3D24 L  3836  3839#
I033            3D37 L  3842  3846#
I034            3D44 L  3831  3850#
I035            3D4D L  3854  3857#
I036            3D6A L  3859  3863  3869#
I04             3BE1 L  3610  3612#
I05             3BFF L  3627  3630#
I06             3C0D L  3636  3639#
I07             3C16 L  3642  3648#
I08             3C28 L  3655  3658#
I09             3C4E L  3663  3688#
IOBYTE          2588 V   138#  503   588   598   609   624   649   674   694   728
                         731
IOERR           47C2 V  3239  3293  6404#
ITALY           2A34 V   344#
ITLOOP          3578 L  2436# 2439
KBDAT86         0001 N  4882  5254  5401  5466  6811#
KBDIN           4201 L  4919  5399# 5402
```

```
KBDIN2          4218 L  5405  5407  5412#
KBDINIT         4174 L   450  5249#
KBDINIT1        4178 L  5252# 5255  5260
KBDINIT2        4189 L  5258  5261#
KBDINIT3        41FD L  5291  5302#
KBDINIT4        4199 L  5267# 5272
KBDINIT5        41AB L  5269  5273#
KBDINIT6        41F1 L  5276  5278  5280  5282  5298#
KBDINIT7        41E7 L  5284  5292#
KBDOUT          4219 L  1032  2046  2048  2050  5462#
KBDOUT1         4221 L  5467  5470#
KBDOUT2         4219 L  5463# 5474
KBDST           41FE L  5348#
KBDTT           2ABA V   133   400# 5287  5293  5409  5410
KBDVAR          483D V  5268  5271  6826#
KBELL           0041 N  5476  6798#
KCOUNT          0041 N  5251  6799#
KEYBASE         0040 N  6794# 6795  6796  6797  6798  6799
KEYIN           4015 L   613   614   630   631  4915#
KEYIN1          401C L  4918# 4929  4933  4935  4944  4960  4964
KEYIN2          4056 L  4917  4942  4946#
KEYINEND        406B L  4925  4939  4952  4955#
KEYST           4004 L   591   592   697  4877#
KEYSTEND        4014 L  4890#
KSPAIN          2A27 V   338#
LANGT1          44A9 V  1259  1260  5815#
LANGT2          44B1 V  1243  1244  5826#
LANGUAGE        483C V  1232  5263  5264  5270  5275  5277  5279  5281  5283  6815#
LEN             0020 N  6339# 6341  6343
LEN1            0009 N   184   187#
LEN10           0009 N   231   234#
LEN11           0011 N   236   240#
LEN12           0005 N   242   244#
LEN13           0005 N   246   248#
LEN14           0005 N   250   252#
LEN15           0005 N   254   256#
LEN16           0005 N   258   260#
LEN17           0005 N   262   264#
LEN18           0005 N   266   269#
LEN19           0005 N   271   273#
LEN2            0009 N   189   192#
LEN20           0005 N   275   277#
LEN3            0008 N   194   197#
LEN4            0009 N   199   202#
LEN5            0009 N   204   207#
LEN6            0008 N   209   212#
LEN7            000A N   214   218#
LEN8            000D N   220   223#
LEN9            0011 N   225   229#
LF              000A N  5722# 5725  5725  5731  5739  5742  5745  5756  5762
LGDAT86         0080 N  5257  6809#
LINBUF          4409 V  1855  1897  5810#
LIPDET          0080 N  5986#
LISTOUT         2C13 L    82   670#
LISTST          2C22 L    92   690#
LOADERBIOS      0000 N    50#   53    63   113   118   183   306   436   448   475
                         501   512   622   637   647   663   672   683   692   703
                        2388  2520  2627  3049  3103  3250  3285  3352  4660  5485
                        6100  6107  6610  6830  7008
LOCSTK          438E V  5768#
LOGLAC          3953 L  3038  3092  3142#
LP11            44E0 V  6075#
LP12            44E1 V  1957  1979  6076#
LP21            44E4 V  6078#
LP22            44E5 V  1956  1984  6079#
LPC             000A N  5975#
LPRD            00C0 N  6004#
```

```
LVAR0          29CE V   302÷  304
LVAR1          29D1 V   308÷  310
LVAR10         2A60 V   361÷  365
LVAR11         2A6F V   367÷  371
LVAR12         2A80 V   373÷  377
LVAR13         2A91 V   379÷  384
LVAR2          29D8 V   312÷  317
LVAR3          29ED V   319÷  323
LVAR4          2A00 V   325÷  329
LVAR5          2A13 V   331÷  335
LVAR6          2A26 V   337÷  341
LVAR7          2A33 V   343÷  348
LVAR8          2A4A V   350÷  355
LVAR9          2A5F V   357÷  359
M1RS232        2590 V   149÷ 5604
M2RS232        2591 V   151÷ 5606
MASKOUT        44C6 N  1365  1378  5879÷
MASKREG        004A N  1808  5999÷
MATCH          3800 L  2938  2960÷
MCRTTBL        2584 V   132÷
MDROW          3520 L  2281  2322÷
MEMSIZ         FE06 N  5707÷ 7018
MEMTAB         4CDC L  7026  7119÷
MFNTBL         2582 V   131÷
MGRBELL        2D7E L   924  1031÷
MGRBKSP        2D35 L   920   979÷
MGRBKSP2       2D41 L   984÷
MGRCALLESC     2D84 L  1039÷ 1047  1062  1102  1108  1114  1120  1126  1201
MGRCLEOS       2DE5 L   963   965  1100÷
MGRCLR         2D88 L   922   951   953  1045÷ 1450
MGRCOL1        2E28 L  1153÷ 1197
MGRCOL2        2E79 L  1189  1193÷
MGRCR          2D4C L   914   994÷
MGRDELCHR      2DED L   957  1112÷
MGRDELLIN      2DF5 L   961  1124÷
MGREEOL        2D82 L   930   967   969  1037÷
MGRESCSEQ      2DF9 L   918  1130÷
MGRFULLI       2DA3 L   945  1066÷
MGRFUNCCH      2EAB L   971  1220÷
MGRGETFLEN     2EA1 L  1210  1212÷
MGRGETFREQ     2E94 L  1206  1208÷
MGRHALFI       2D8C L   943  1051÷
MGRHOME        2D60 L   926   983  1008÷
MGRINSCHR      2DE9 L   955  1106÷
MGRINSLIN      2DF1 L   959  1118÷
MGRINV1        2E0B L  1138  1143÷
MGRINV2        2E53 L  1148  1174÷
MGRINV3        2E5A L  1145  1178÷
MGRINV4        2E66 L  1180  1184÷
MGRINVERSE     2DFF L   947  1137÷
MGRLF          2D78 L   916  1023÷
MGRMUSIC       2E8B L   949  1205÷
MGRNDFS        2D68 L   928  1014÷
MGRPOSCUR      2D88 L   941  1078÷
MGRRET         2D48 L   934   973   989÷
MGRRET2        2E0A L  1141÷ 1187  1195
MGRRLF         2D53 L   932  1000÷
MGRSETATTR     2D9F L  1055  1060÷ 1070  1074
MGRSETATTR1    2E86 L  1150  1172  1176  1182  1191  1199÷
MGRWRITEPOS    2D46 L   981   986÷  996  1002  1004  1010  1017  1096  1343
MIXGAC         0000 N  6042÷
MKEYTBL        2586 V   133÷
MMAREA         2580 V   130÷
MMESS          2588 V   134÷
MOCOMP         0001 N  6019÷
MODEFL         258E V   141÷  463   465   579
MOREPL         0000 N  1643  1729  1894  6018÷
MORES          0002 N  6020÷
```

```
MOSET            0003 N  6021#
MOTORCK          3EE8 L  4390  4559#
MOTORCK1         3EF1 L  4563  4565#
MOTORCK2         3EF4 L  4567# 4571
MOTORON          0014 N  4562  6476#
MOVCPM           4C00 L   437  7006#
MOVEND           4C08 L  7103  7116#
MRT              42FD V   737  5717#
MRTLEN           4300 V  5719# 7034  7041  7044
MSTRCMD          006F N  5843#
MWROW            350F L  2238  2299#
MUSIC            3372 L  1615  2044#
NBRFLEX          258A V   137# 2508  2638  3041  3095  3265  3384  3431
NCRTYP           3668 V  2582  2596#
NOCOMP           386F L  2999  3008#
NOINTL           0000 N  6046#
NOLAC            3975 L  3152  3157#
NOMATCH          37E2 L  2928  2933  2939#
NOOP             30F3 L  1606  1616  1618  1619  1620  1621  1625#
NOOVF            3791 L  2884  2889#
NOSETSEG         4C0E L  7012  7015#
NOTBLINKING      FFFD N  1146  5799#
NOTCOLOURHALFI   FFFA N  1073  5801#
NOTDRQFLG        FFFE N   886  5791#
NOTESCFLG        FFFD N   891  5793#
NOTHALFINTENSITY FFFB N  1069  5795#
NOTIMPL          3AEE L  3362  3378#
NOTINVERSE       FFFE N  1175  5797#
NOTNCR           3668 L  2585  2593#
NOTRAN           36FF L  2773  2778#
NOTRDY           477F V  2631  3230  3254  6389#
NRWAPL           0048 N  5973# 5974
NUMHDSK          2594 V   155# 2499
O1               3083 L  1553  1555#
OLDFW            2B89 L   545   558#
OUTCHAR          440D V  1526  1584  6069#
OVERRUN          0010 N  5551  6884#
P1CHROUT         42D9 L   657   678  5658# 5659
P1STA1           42ED L  5668  5670#
P1STATUS         42E3 L   698  5658  5666#
P1STATX          42F6 L  5672  5676#
PACTIVE          483F V   572  5611  5652  5666  6919#
PARA40           42F9 V  5711# 5713  7014
PARITY           0008 N  5551  6883#
PATCHSIZE        03BF N  6984# 6985
PATTERN          4806 V  3387  4339  6595#
PBCOM            0063 N  5650  6971#
PBDA             0060 N  5661  6969#
PBSTA            0061 N  5670  6970#
PINIT            42CA L   507  5649# 5669
PITCH            0047 N  5998#
PNAADDR          477A V  3036  3037  3090  3091  3169  3171  3260  3262  3422  3479
                         6375#
PMSG             2C4E L   459   495   742#  751  1331  1333  1338  1356  1383
POBF             0002 N  5671  6976#
PORTUG           2A81 V   374#
POSMSG           47D7 V  1330  1355  6410#
PRAM             0070 N  1989  6006#
PRAMSA           0000 N  6007#
PRINIT           2B31 L   461   499#
PROCCTL          2CDC L   871   905#
PROCDRQ          2CB6 L   885#
PROCESC          2CC1 L   881   890#
PROCSTATUS       2CAF L   867   879#
PROTECT          4792 V  3234  6393#
PUNCH            2BFC L    83   645#
PVRS232          2593 V   153# 5521
RAMSELECT        0010 N   562  5765#
```

```
RD              3FAF L  4737  4758#
RD2             3FBE L  4763# 4766
RD3             3FCA L  4771# 4772
RD4             3FCE L  4770  4773#
RDAT            00A0 N  1682  1853  6023#
RDGCHR          3113 L  1663# 2066  2099
RDKEY           0040 N  5259  5262  5403  5468  6796#
RDLIN           3218 L  1828# 2256  2308  2330
RDLIN1          3250 L  1857# 1861
READ            3702 L    90  2801#
READDAT         0006 N  3587  3715  6501#
READEND         38DF L  3046  3075#
READER          28E9 L    84   620#
READHST         388B L  2957  3035#
READOP          4774 V  2803  2831  2979  6371#
READRET         38EE L  3080  3083#
READSEC1        3616 L  2566# 2579
READTRK         0002 N  6499#
READY           3681 L  2630  2636#
RELID           2570 V   124#
RESETCMD        0000 N  5840#
RESMOP          0000 N  6039#
RESMSG          470C V  1337  6411#
REST            0010 N  6633  6673#
RESTORE         0007 N  3927  6502#
RET1            288E L   556   561#
RETRIES         4807 V  2393  2426  3764  6597#
RETRYC          258C V   139#
RETTYP          366A L  2590  2592  2594#
RETURN          2C5F L   745   752#
RETURNRW        388A L  3015  3019  3023#
REVVID          4406 V  1149  1151  1194  1196  5803#
ROMSELECT       0011 N   537  5766#
ROWS            0018 N  1084  2007  2149  2179  2194  2232  2245  2275  2278  5945#
RSFLAG          4773 V  2804  2891  2896  2954  6370#
RSKEY           0041 N  4880  5253  5256  5349  5400  5465  5471  6797#
RSTC            2580 V   140# 2391  2424  3051  3105
RSTLEN          481E V  5071  5116  5144  6746#
RWMOVE          384C L  2981  2989#
RWOPER          37A2 L  2806  2892  2908#
RXRDY           0002 N  5556  6880#
S1              3855 L  3432  3435#
S2              3857 L  3434  3437#
SACTIVE         483E V   571  5547  5569  5610  5651  6918#
SAFRICA         2A70 V   368#
SATRM           0000 N  6050#
SCLDN1          34F6 L  2280# 2284
SCLDN2          3502 L  2277  2286#
SCLUP1          349F L  2237# 2241
SCLUP2          34AB L  2234  2244#
SCLUP3          348F L  2185  2230  2252#
SCLUP4          3489 L  1554  1588  2250#
SCROL1          3327 L  1982  1985# 2192
SCROL2          3323 L  1980  1983#
SCROLLCMD       0070 N  5847#
SCROLLDN        34E7 L  1610  2272#
SCROLLUP        3488 L  1611  2227#
SCROLLX         3304 L  1972# 2260
SCWID           0050 N   985  1016  1092  1552  1559  1586  1593  2003  2035  2060
                        2123  2132  2156  2194  2194  2203  2206  2302  2305  2325
                        2332  5946#
SDH             00C6 N  4722  4743  4745  6660#
```

```
SDHREG       00A0 N  4721  6716#
SECCNT       47E8 V  2573  3174  3409  3594  3609  3611  3650  3739  3815  6523#
SECNO        00C3 N  4678  4682  4709  6657#
SECNT        00C2 N  4799  6656#
SECTOR       47E7 V  2572  3160  3212  3218  3453  3602  3827  3835  3840  3853
                     3862  3866  3872  4263  6521#
SECTRAN      36F4 L    93  2770#
SECTRK       4802 V  3186  3600  3607  3828  4185  4267  4336  6585#
SEEK         0070 N  6674#
SEEKTRK      000F N  4000  6506#
SEKDSK       4762 V  2503  2515  2523  2532  2554  2610  2837  2842  2858  2879
                     2911  2926  2947  6352#
SEKHST       476A V  2917  2936  2952  6360#
SEKSEC       4765 V  2700  2846  2870  2915  2966  6354#
SEKTRK       4763 V  2474  2675  2844  2866  2932  2949  6353#
SELDSK       35A0 L    86  2498#
SELERR       35E5 L  2500  2537#
SELTYP       3898 L   100  3501#
SENPAR       31B1 L  1757# 1764  1993
SEREAD       3877 L  3373  3464#
SERNUMBER    2596 V   157#  455
SET          3A50 L  3253  3259#
SET1         3E3B L  4248  4252#
SETBDOS      4CA7 L  7082  7093#
SETCUR       31BF L  1772# 2024  2065  2070  2081  2098  2103  2217  2255  2262
                     2307  2310  2329  2333
SETCUR1      31CD L  1702  1775  1777#
SETDMA       36EA L    89  2721#
SETDMAB      36EF L    94  2746#
SETDRQFLG    2E05 L  1081  1089  1139# 1207  1211  1223
SETEND       3A83 L  3258  3275#
SETFIXVAR    3A3E L  3056  3110  3248# 3257  3393
SETFLXVAR    3978 L  3043  3097  3166# 3388  3408
SETINT       4C56 L  7039  7049#
SETIOBF      2C44 L    97   730#
SETMSK       31F9 L  1705  1799  1804#
SETRET       3A12 L  3189  3198  3205  3209  3214  3219#
SETSEC       36E5 L    88  2699#
SETTRK       36E0 L    87  2674#
SETUP6       3E7A L  4190  4323#
SETUP9       3E25 L  3767  4242#
SEWRITE      3885 L  3374  3471#
SFUNCTAB     3A02 L  3357  3361#
SHOME        386D L  3371  3457#
SIFDISP      422F L  5521# 5543
SIGNON       431A V   458  5730#
SIOINIT      42AF L   505   506   508  5549  5571  5604#
SLVCMD       006E N  5844#
SODISPTBL    423A L  5520  5527#
SP1          44DE V  1773  1954  1976  1978  1992  6074#
SP2          44E2 V  1955  6077#
SPA1         427F L  5570  5572#
SPA2         428A L  5573  5576#
SPA3         429A L  5577  5583#
SPA4         4299 L  5580  5582#
SPAI1        425A L  5548  5550#
SPAI2        4263 L  5552  5555#
SPAI3        426B L  5557  5559#
SPAIN        429D L   612   615   629   632  5574  5588# 5589
SPAIST       4250 L   590   593  5547# 5572  5588
SPAOST       4275 L  5534  5535  5536  5537  5569# 5596
SPAOUT       42A5 L   600   603   655  5528  5529  5530  5531  5596# 5597
SPAREA       258A L   130   136#
SPCLEAR      347F L  1975  2142  2160  2195  2207  2215# 2259
SPCLEAR1     3146 L  1694# 2219
SPCLEAR2     315C L  1697  1699  1703#
SPECFUN      3AC6 L    99  3350#
SPRCOM       0063 N  5562  6867#
```

```
SPRDATA         0060 N 5561  5590  6865#
SPRSTAT         0061 N 5550  5555  5578  6866#
SPWCOM          0067 N 5564  5609  6870#
SPWDATA         0064 N 5599  6868#
SPWMODE         0066 N 5605  5607  6869#
SRDRET          383B L 3415  3417#
SRDTRK          3822 L 3364  3407# 3413
SREAD           3898 L 3039# 3077  3369  3466
SRLOUT          422C L  676   679  5520#
SRLSTAT         424A L  696   699  5542#
SS              SREG V  441
SSB             4808 V 3670  3694  3719  6605#
SSELDSK         3841 L 3366  3426#
SSETDMA         383C L 3365  3421#
SSETDMAB        3893 L 3375  3478#
SSETSEC         3864 L 3368  3451#
SSETTRK         3850 L 3367  3444#
SSETTRK2        385F L 3372  3446#
SSTDISPTBL      4242 L 5533# 5542
START           006B N 5995#
STARTCMD        0000 N 1431  5841#
STAT            00C7 N 4729  4781  4784  6662#
STATUSFLAG      4404 V  866   880   886   891  1131  1140  5079  5124  5154  5789#
STKBASE         43FE N  445  5769#
STOPCMD         000C N 1437  5842#
STRATE          0000 N 6672# 6673  6674
SWEDEN          2A01 V  326#
SWISS12         2A4B V  351#
SWRITE          38FC L 3093# 3131  3370  3473
SWRTRK          3AEF L 3363  3382# 3399
SYSSTA          0013 N 3935  4011  4560  6475#
TESTPOS         33E7 L 2057  2090  2120#
TESTVIDOUT      3073 L 1538  1548#
TEXTECMD        0068 N 5846#
TPALENGTH       0980 N 5706# 5719
TPASTART        0680 N 5693# 5706  5718
TRO             0002 N 6687#
TRACK           47E6 V 2571  3179  3197  3199  3207  3208  3841  3844  3848  3870
                       4006  4259  6520#
TRAN1           2EE7 L 1234  1255#
TRAN2           2EF1 L 1238  1245  1257  1262#
TRAN3           2F10 L 1284# 1291
TRAN4           2ED0 L 1236  1241#
TRANEND         2F1D L 1249  1251  1253  1289  1296#
TRANHEBREW      2ED8 L 1230  1247#
TRANSLATE       2CC9 L  893#  899   907
TRANSMATCH      2CD7 L  895   897   900#
TRERR           426C L 5553  5561#
TRUE            FFFF N   47#   48
TXEMT           0004 N 6882#
TXRDY           0001 N 5579  6879#
TYHIBY          0018 N 6016#
TYLOBY          0010 N 6015#
TYWORD          0000 N 1643  1682  1729  1894  6014#
UK              2902 V  309#
UNA             0000 V 2851  2857  2861  2872  2876  2877  2887  6159#
UNACNT          4760 V 2447  2802  2841  2850  2895  6364#
UNADSK          476E V 2843  2859  6365#
UNASEC          4771 V 2847  2871  6367#
UNATRK          476F V 2845  2865  2888  6366#
UNCOR           0040 N 6691#
UNUSEDOFFSET    037C V 7145#
UNUSEDSEGMENT   037E V 7146#
US              29CF V  303#
VAR0L           0003 N  302   304#
VAR10L          000F N  361   365#
VAR11L          0011 N  367   371#
```

```
VAR12L            0011 N  373   377#
VAR13L            0015 N  379   384#
VAR1L             0007 N  308   310#
VAR2L             0015 N  312   317#
VAR3L             0013 N  319   323#
VAR4L             0013 N  325   329#
VAR5L             0013 N  331   335#
VAR6L             000D N  337   341#
VAR7L             0017 N  343   348#
VAR8L             0015 N  350   355#
VAR9L             0001 N  357   359#
VCLEAR            3459 L 1607  2189#
VECTECMD          006C N 5845#
VER               0000 N 6340# 6341
VERETR            0020 N 5984#
VERLEN            0020 N 3503  6341#
VSYNCM            006F N 5993#
VSYNCS            006E N 5992#
WAIT              3FD1 L 4758  4781# 4783  4810
WBOOT             2B92 L   78   570# 3324
WBOOT1            2BAB L  574   577#
WDAT              0020 N 1643  1729  1894  6011#
WDATOUT           44CD N 1372  5888#
WDKEY             0040 N 6795#
WIERROR           00C1 N 4790  6654#
WIPAR             480A V 2639  2640  2641  2642  2643  2644  3057  3059  3066  3111
                         3113  3120  3261  3263  3266  3272  3273  3274  3287  3291
                         3394  4703  4710  4730  4733  4760  4761  4785  4791  4802
                         4803  6632#
WIREAD            0020 N 4736  6675#
WIWRITE           0030 N 4738  6676#
WPC               00C1 N 6655#
WPL               0024 N 5974#
WR                3FEB L 4739  4801#
WR0               3FE7 L 4741  4798#
WR1               3FF7 L 4805# 4808
WR2               3FFF L 4742  4810#
WRALL             0000 N 6152#
WRDIR             0001 N 3013  6153#
WRGCHR            30F4 L 1557  1591  1639# 2071  2083  2104
WRHLPOS           3367 L 2023  2033# 2061  2122  2140  2155
WRITDAT           0005 N 3591  3662  6500#
WRITE             3719 L   91  2829#
WRITEEND          3943 L 3100  3129#
WRITEHST          38EF L 2546  2944  3021  3089#
WRITEPOS          335C L 1614  2013  2021# 2084  2143  2171  2182  2220  2248  2265
                         2290
WRITERET          3952 L 3133  3136#
WRITFMT           000D N 4182  6505#
WRLIN             3259 L 1870# 2263  2311  2334
WRLIN1            3291 L 1899# 1906
WRTRKEND          3B17 L 3391  3397#
WRTRKRET          3B21 L 3401  3403#
WRTYPE            4775 V 2805  2832  3013  6372#
WRUAL             0002 N 2805  2833  6154#
XLT               4742 V 3155  3504  6343#
XOFF              0013 N 5576  6875#
XOFFFLG           4840 V 5575  5576  6920#
XON               0011 N 6874#
XWAIT             3E83 L 3768  3933  4009  4077  4131  4191  4389#
XWAIT1            3EB0 L 4394# 4400
XX1               30F4 L 1640# 1642
XX10              3259 L 1871# 1873
XX11              3281 L 1891# 1893
XX12              329F L 1912# 1914
XX13              32B4 L 1927# 1929
XX14              32EF L 1958# 1960
```

```
XX15          3327 L  1986# 1988
XX16          30FE L  1645# 1647
XX17          3108 L  1650# 1652
XX18          3110 L  1669# 1671
XX19          3127 L  1674# 1676
XX2           3113 L  1664# 1666
XX20          316A L  1711# 1713
XX21          3174 L  1716# 1718
XX22          317E L  1721# 1723
XX23          3192 L  1731# 1733
XX24          319C L  1736# 1739
XX25          3181 L  1758# 1760
XX26          3107 L  1783# 1785
XX27          31E1 L  1788# 1791
XX28          31EB L  1794# 1796
XX29          3203 L  1810# 1812
XX3           3131 L  1679# 1681
XX30          3200 L  1815# 1817
XX31          3222 L  1834# 1836
XX32          322C L  1839# 1841
XX33          3236 L  1845# 1847
XX34          3263 L  1876# 1878
XX35          326D L  1881# 1883
XX36          3277 L  1886# 1888
XX37          3291 L  1900# 1902
XX38          32A9 L  1917# 1919
XX39          328E L  1932# 1934
XX4           3160 L  1706# 1708
XX40          32C8 L  1937# 1939
XX41          3204 L  1942# 1944
XX42          32F9 L  1963# 1965
XX5           3188 L  1726# 1728
XX6           31CD L  1778# 1780
XX7           31F9 L  1805# 1807
XX8           3218 L  1829# 1831
XX9           3240 L  1850# 1852
YUGOSL        2A92 V   380#
ZOOM          0046 N  5996#
```

```
                               TITLE 'CP/M-86   BOOT RECORD'
                       ;
                       ;
                               CSEG
                               ORG     0000H
                       ;
                       BOOT_REC_START:
0000 E91E00       0021         JMP     START          ;CANNOT DO THE JMPF HERE - ONLY 3 BYTES
                                                      ;ALLOWED HERE
                       ;
0003 313642495420              DB      '16BIT         ;ID FOR ROM BOOT
     20
000A 4E4352204633              DB      'NCR F3'       ;DISK FORMAT
0010 0000000000                DB      00H,00H,00H,00H,00H
0015 3030303030                DB      '0','0','0','0','0'     ;SERIAL NUMBER
001A 284329204E43              DB      '(C) NCR'      ;COPYRIGHT
     52
                       ;
                       START:
0021 8CC8                      MOV     AX,CS
0023 8ED8                      MOV     DS,AX
0025 FF2E2900                  JMPF    START_LDCPM
                       ;
                       ;
                       ;
  0029                 END_CSG EQU     OFFSET $
                               DSEG
                               ORG     END_CSG
                       ;
0029 0000              LDCPM_OFF      DW      0000H   ;LDCPM OFFSET
002B 2002              LDCPM_SEG      DW      0220H   ;LDCPM SEGMENT
  0029                 START_LDCPM    EQU     DWORD PTR LDCPM_OFF
                       ;
  002D                 END_OF_PROG    EQU     OFFSET $
  01D3                 PADLEN         EQU     200H-END_OF_PROG
002D                   PADAREA        RS      PADLEN  ;RESERVE THE REST OF THE FIRST SECTOR
  0200                 BOOT_REC_END   EQU     $
0200 00                DUMMY          DB      0
```

We Welcome Your Evaluations Of Our Publications. Please Answer This Questionnaire And Mail It To Us.

Thank You

NCR

NCR DECISION MATE V SYSTEM TECHNICAL MANUAL

› How Do You Rate This Document?

	Clarity	Accuracy	Organi-zation	Complete-ness	Usability
Excellent (Needs No Improvement)					
Good (Could Use Minor Improvements)					
Fair (Should Have Major Improvements)					
Poor (Should Be Completely Chang^d)					

How Often Do You Use This Document?

)ften Occasionally Rarely

'our Name:

osition:

Comments:

NCR GmbH
Technical Publications Department
Ulmer Strasse 160A
8900 Augsburg
Federal Republic of Germany