

**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

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 MANUAL
CP/M-80**

CONTENTS

CP/M-80 SOFTWARE FOR INPUT/OUTPUT

Logical Disk Layout	3-3
Flexible Disk (5 1/4-inch)	3-3
Winchester Disk	3-6
I/O Functions	3-7
BDOS Entry Functions	3-8
File Information	3-12
Disk Information	3-14
Logical Assignment of I/O Devices	3-17
Terminal Functions	3-19
The BIOS Program	3-22
How to Read the BIOS Program	3-22
Displaying the BIOS Program on the Screen	3-24
The BIOS Entry Points	3-26
The BIOS Sections	3-28
Making Use of the I/O Software	3-29
Some I/O Examples	3-30
Interfacing Printers	3-44
Tables-Routines-Ports	3-46
Level Zero Diagnostics	3-56
Color	3-56
Graphics	3-57
The Graphics Display Controller	3-58
Some GDC Programming Examples	3-70
Color Graphics	3-100

APPENDIX B

CP/M-80 BIOS Program	B-1
CP/M-80 Linklist	B-187

CP/M-80 SOFTWARE FOR INPUT/OUTPUT

CP/M-80™ Software consists of a collection of file processing utility programs, an assembler for the conversion of mnemonic assembler language into machine code, and a debugging tool with which programs can be run and tested at machine code level. CP/M is described as an “operating system” as it provides the framework in which the user can write, store, and run programs.

Essential to any operating system software is the provision of communication facilities between CPU, memory, keyboard, CRT, and/or print-list device. It is also desirable to provide interfaces to additional peripheral devices and to facilitate DMA. CP/M fulfills all these functions.

Typically, CP/M is present on a flexible disk. As soon as the above-mentioned basic functions are present in memory, individual CP/M utilities can be loaded into memory and, when these utilities are no longer required, other CP/M utilities or user programs can take their place. Because CP/M does not execute from ROM but from read/write memory, it is possible to accommodate operating system variables within the system functions with which they are most immediately concerned.

CP/M regards the memory as four distinct areas, as shown in Figure 3.1.

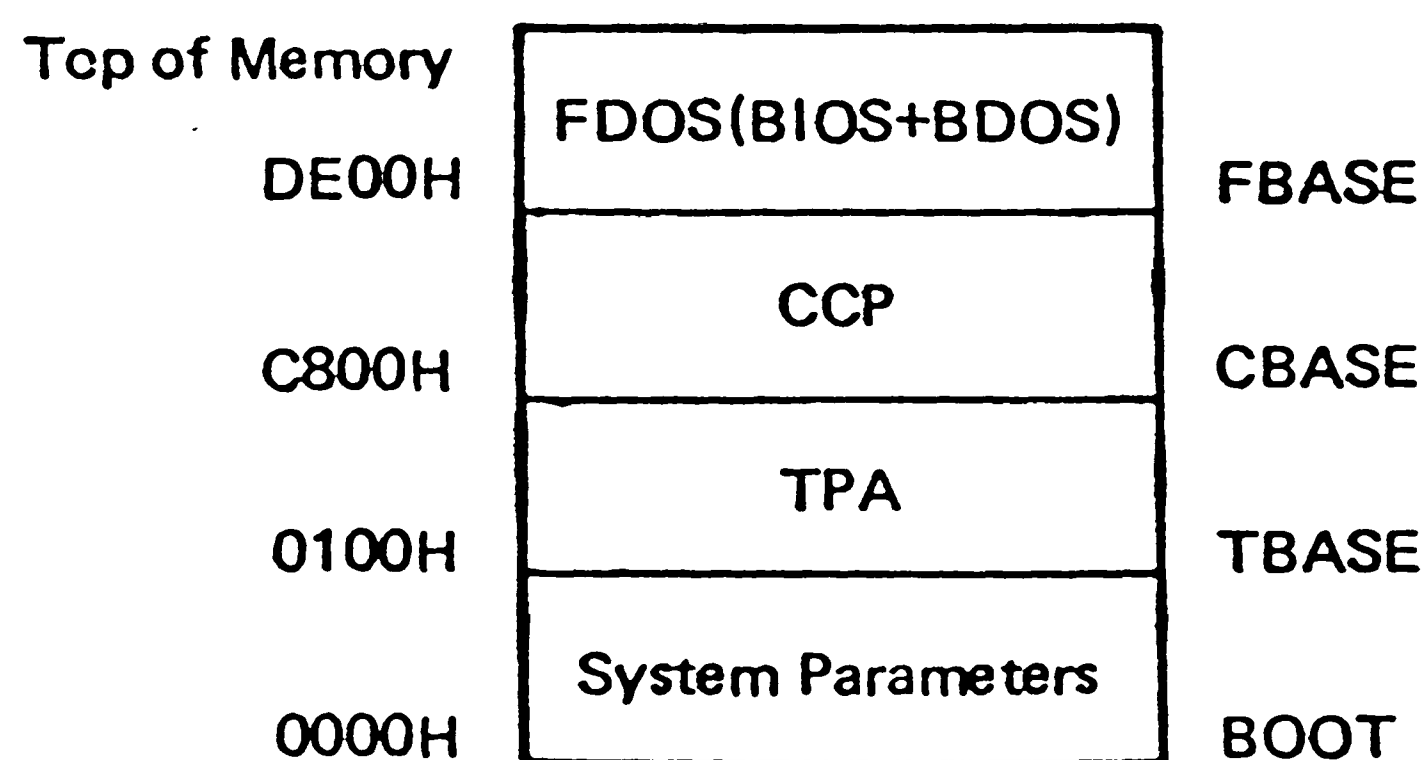


Figure 3.1

At cold start the first sector on the flexible disk (Track 0, Sector 1) loads. The BOOT program from this sector then loads the BIOS to top of memory. The BDOS and CCP are loaded into

the memory area immediately below the BIOS. All these communication functions of the system software are present on the first three tracks of the CP/M flexible disk, as summarized in Figure 3.2.

Function	Disk Sectors
BOOT	Track 0 : Sector 1
BIOS	Track 0 : Sector 2 through Track 1 : Sector 5
CCP	Track 1 : Sector 6 through Track 2 : Sector 1
BDOS	Track 2 : Sector 2 through Track 2 : Sector 8

Figure 3.2

The only additional memory which is required by these system components is the area from address 0 through 0FFH (TBASE). Especially useful addresses in this area are shown in Figure 3.3.

The remaining memory (TPA) is available to CP/M utilities and user programs. Assuming that the BIOS and BDOS areas are not overwritten, it is possible to make use of the CP/M I/O functions in user programs. Overwriting the CCP area does not mean irretrievable loss of system control, as this area is restored following a warm boot (jump to 0000H). In the event of the BIOS or the warm boot jump at 0000H being overwritten, a cold start will be necessary.

Memory Location	Use by CP/M
0000H-0002H	Unconditional jump (Opcode C3) to warm start entry point in BIOS *
0003H	IOBYTE: governs interface to physical devices
0004H	Default drive number (0 = A etc)
0006H-0007H	Address for access to BDOS (FBASE). Machine code instruction "2A0600" returns access address in HL, unless DDT utility is loaded.* Byte 0005 provides the jump instruction to that address.
0008H – 0037H	Interrupt locations 1 through 6, not used, but location 6 to be considered as reserved.
0038H – 003AH	Restart 7: Unconditional jump to DDT, used for breakpoints when DDT is running.
003BH – 003FH	To be considered as reserved.
0040H – 004FH	Scratchpad area, at present not used.
0050H – 005BH	To be considered as reserved.
005CH – 007CH	Default FCB produced for a transient program by the CCP. (See "File information.")
007DH – 007FH	Optional default random record position. (See "File Information.")

* machine addresses are stored with most significant byte nearest to top of memory.

Figure 3.3

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 formatting utility included in the CP/M describes the surface 1 tracks with the numbers 40 through 79. This is to facilitate user identification of a faulty

track. The convention of flexible disk description is to designate the tracks on both surfaces 0 through 39. Figure 3.2 has shown that tracks 0, 1, and 2 of surface 0 contain the I/O software. With the exception of track 3, which is reserved for the CP/M directory (128 entries, each 32 bytes), the remaining tracks (surface 0:4 . . . 39, surface 1:0 . . . 39) are available for data storage. 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 operating system. Figure 3.4 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, 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.

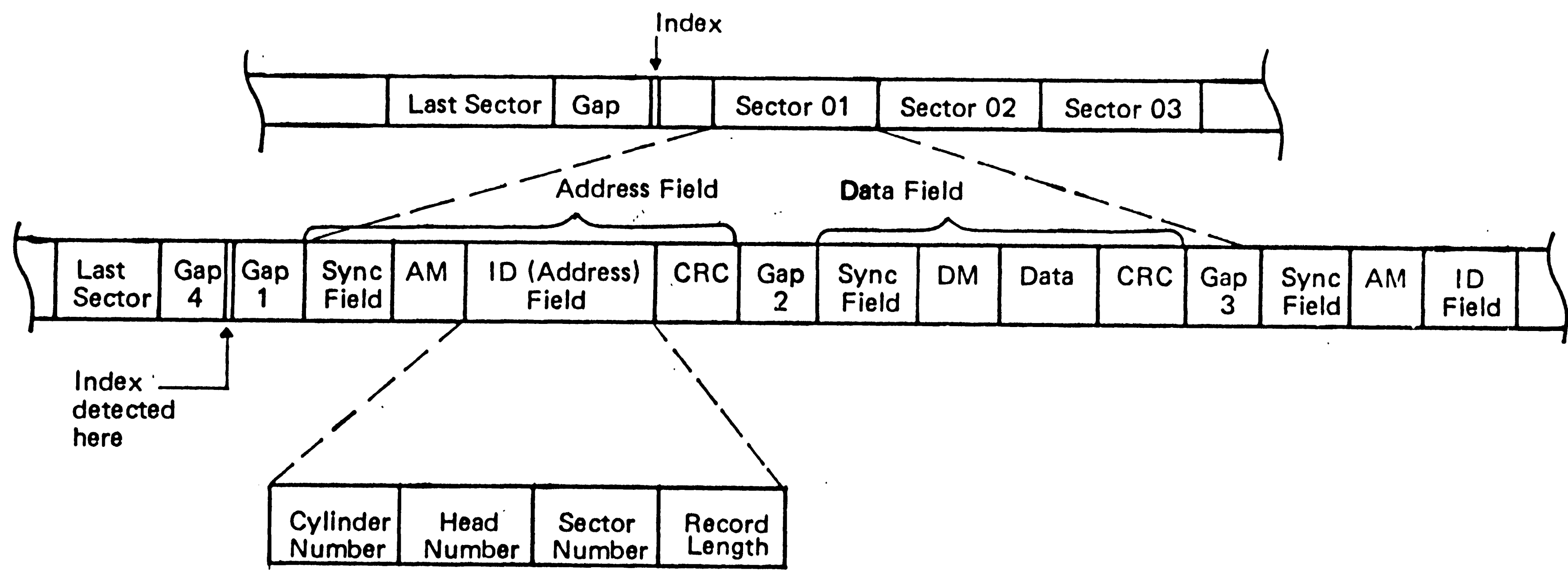


Figure 3.4

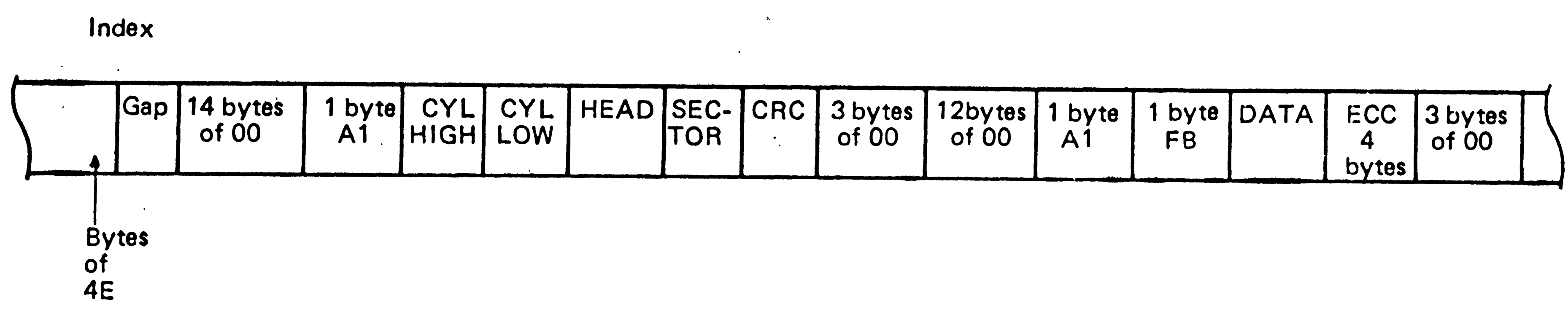


Figure 3.5

DM (Data Marker)

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

CM (Control Marker)

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

ID (Address) Field

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

Byte 1 Track (cylinder) number zero through 27H.

Byte 2 Surface (head) number: 00 = surface 0, 01 = surface 1.

Byte 3 Sector number 01 through 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).

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. Figure 3.5 shows this layout.

Gap

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

CYL HIGH

Value FF: cylinders 256 to 511

Value FE: cylinders 0 to 255

Value FD: cylinders 768 to 1023

Value FC: cylinders 512 to 767

HEAD

Bit 7 set indicates a bad block

Bytes of 4E

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

I/O FUNCTIONS

CP/M provides software control over the following I/O and system functions:

- RAM integrity check
- System reset
- Disk drive selection
- Set disk status
- Search for and access disk files
- Create and restore disk files
- Disk drive motor and head travel
- Reader Input and Punch Output
- Console I/O
- Keyboard code translation
- Set function keys
- List Output
- Serial and parallel I/O
- DMA
- Drive loudspeaker
- Display error messages

When BIOS (with BDOS) has loaded, the software which executes these functions is available from FBASE upwards. There is, however, one exception: the RAM integrity check routine along with the copyright message on the CRT (approximately 390 bytes) are overwritten as soon as the check has been passed.

BDOS ENTRY FUNCTIONS

Important I/O functions can be accessed through a machine call instruction to the address 0005H in page zero of the memory. This, in turn effects a jump to BDOS, and from there to the specified function within the I/O memory area. Before the call is made to page zero, certain CPU registers have to be loaded in accordance with the I/O function to be carried out. Certain functions return a result value in A (one-byte result) or HL (two-byte result). Figure 3.6 explains briefly the nature of each function, the function number which must be loaded in Register C, 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.

Function no. in Reg. C (Hex)	Description	Additional Entry Parameters	Return Value
00	System reset.	—	—
01	Console input — waits for character, which is echoed to console.	—	Reg. A: ASCII character
02	Console output — tabs expanded, print control chars. recognized.	Reg. E: ASCII character	—
03	Reader input — waits for character	—	Reg. A: ASCII character
04	Punch output.	Reg. E: ASCII character	—
05	List output.	Reg. E: ASCII character	—
06	Direct console I/O — in contrast to 01, control characters are ignored.	Reg. E: 0FFH	Reg. A: ASCII character from console, 00H if none ready
		Reg. E: <> 0FFH	console: contents of Reg. E
07	Get I/O Byte.	—	Reg. A: IOBYTE
08	Set I/O Byte.	Reg. E: IOBYTE	—

Figure 3.6 (1 of 4)

Function no. in Reg. C (Hex)	Description	Additional Entry Parameters	Return Value
09	Print string until \$ encountered — tabs and control chars. as in 02.	Reg. DE: String address	—
0A	Read console buffer — reads console input into buffer at address DE until CR (0DH) or LF (0AH) or overflow. Other control chars. recognized.	Reg. DE: Buffer address (DE+0): Buffer length	(DE+1) number of characters in buffer
0B	Get console status.	—	Reg. A: <> 0 if char. ready, other- wise 0
0C	Return version number.	—	Reg. H: 0 = CP/M, 1 = MP/M Reg. L: 0 = version be- fore 2.0, lower nibble = release 2.n
0D	Reset disk system — all disks read/write, disk A selected, default DMA address = 80H.	—	—
0E	Select default disk.	Reg. E: Drive A = 0 .. Drive P = 0FH	—
0F	Open existing file — if found, directory information copied to FCB.	Reg. DE: FCB address	Reg. A: 0,1,2, or 3 = found, other- wise 0FFH.
10	Close existing file — new FCB recorded in disk directory.	Reg. DE: FCB address	Reg. A: 0,1,2, or 3 = old directory entry found, otherwise 0FFH
11	Search for first file entry in directory corresponding to FCB.	Reg. DE: FCB address	Reg. A: 0,1,2, or 3 = found, other- wise 0FFH
12	Search for next file entry after last matched entry.	—	Reg. A: 0,1,2, or 3 = found, other- wise 0FFH
13	Delete file matching FCB.	Reg. DE: FCB address	Reg A: 0,1,2, or 3 = found, other- wise 0FFH

Figure 3.6 (2 of 4)

Function no. in Reg. C (Hex)	Description	Additional Entry Parameters	Return Value
14	Read next record of opened file (function 0F or 16) to DMA address (function 1A).	Reg. DE: FCB address	Reg. A: 0 = read successful, otherwise no data at record position
15	Write next record of opened file (function 0F or 16) from DMA address (function 1A).	Reg. DE: FCB address	Reg. A: C = write successful, otherwise disk full
16	Make file which does not already exist.	Reg. DE: FCB address	0,1,2, or 3 = successful, 0FFH = no directory space
17	Rename file.	Reg. DE: address of FCB inc. old name. (DE+10H): new name	0,1,2, or 3 = successful, 0FFH = old name not found
18	Drive on line.	—	Reg. HL: bit significance 0 . . . 15 corresponds to drive A . . . P, bit set = drive on line
19	Current default drive.	—	Reg. A: 0 . . . 0FH corresponding to drive A . . . P
1A	Set DMA address — i.e. address of data record for read or write operation. Functions 0,0D, and cold start reset this address to 80H. The first byte is used to hold the number of DMA chars.	Reg. DE: DMA address	—
1B	Get address of drive allocation vector.	—	Reg. HL: address at vector
1C	Temporary disk write protection.	—	—

Figure 3.6 (3 of 4)

Function no. in Reg. C (Hex)	Description	Additional Entry Parameters	Return Value
1D	Get read only vector.	—	Reg. HL: bit significance 0...15 corre- sponds to drive A...P, bit set = R/O
1E	Set file attributes in directory in accordance with attributes in FCB.	Reg. DE: FCB address	Eeg. A: 0FFH = file named in FCB not found
1F	Get address of disk parameter block		Reg. HL: block address
20	Get/set current user number.	Reg. E: 0FFH	Reg. A: user number
		Reg. E: < > 0FFH	new user num- ber = content of Reg. E
21, 22 23, 24	These functions concern random file access — see next section "File Information."		
25	Reset drives specified in vector.	Reg. DE: bit significance 0...15 corre- sponds to drive A...P, bit set = drive to be reset	Reg. A: 00H
26, 27	Not in use		
28	Used in random file access — see next section "File Infor- mation."		

NOTE: Functions 1C and 20 may not be supported in later CP/M versions.

Figure 3.6 (4 of 4)

FILE INFORMATION

A disk file is identified by the drive select code, the file name, and the file type. 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 interprets the hexadecimal value 1A as end-of-file except in machine-executable files (e.g. COM). A file is divided into 16 Kbyte logical extents automatically accessed in both sequential and access modes.

A CP/M utility or user program may make use of the default file control block situated at address 005CH. The basic unit used in the reading and writing of files is the 128-byte record, for which CP/M provides a default location at 0080H. The DMA address automatically points to this location after either of the Page Zero functions 0 and 0D, or after a cold start.

The FCB data area (i.e. from 005CH onwards) uses 33 bytes for sequential, and 35 bytes (i.e. up to and including 007FH) for random file access. The FCB layout is shown in Figure 3.7. The numbers 00 to 35 in the layout denote the offsets of the individual bytes to the FCB beginning.

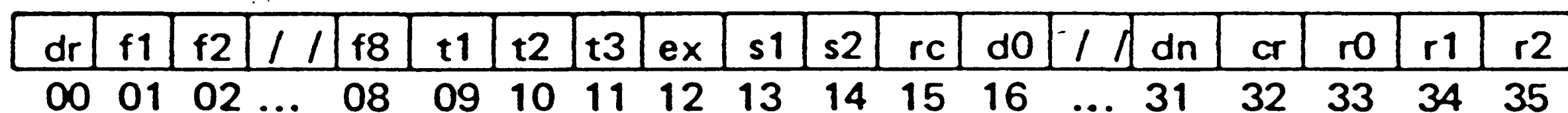


Figure 3.7

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. 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 0-127.
- 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 10).

Function 21 has as its entry parameter an FCB address in the register pair HL. 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 A 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 A are:

- 01 or 04 Read attempted beyond last file extent.
- 06 Read attempted beyond physical end of disk.

Full details of error codes are to be found in the NCR CP/M Manual.

Function 22 is a write-random facility, using data from the current DMA address. The information given above about

Function 21 applies analogously to this function. In addition, error code 05 indicates failure to write due to directory overflow.

Function 23 refers to the FCB addressed by the DE register pair 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 24 is used to set a random record number in bytes r0 and r1 of the FCB addressed by the DE register pair. 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. 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

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

DPB

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

CSV

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

ALV

Address of a scratchpad area used by the BDOS to keep disk storage allocation information. This address 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 base address of the DPH table.

A responsibility of the SELDSK subroutine (see "The BIOS Entry Points") is to return the base address 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 F, for fixed disk a value of 3F).

EXM

The extent mask, determined by the data block allocation size and the number of disk blocks.

DSM

Determines the total storage capacity of the disk drive.

DRM

Determines the total number of directory entries that can be stored on this drive. (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 = 2048 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.

AL0								AL1							
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15

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, resulting in the following tabulation.

BLS Directory Entry

2048 64 times # bits

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 3.8 shows the possible bit settings within the IOBYTE at address 0003H, 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 3.8 (1 of 4)

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

Figure 3.8 (2 of 4)

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

Figure 3.8 (3 of 8)

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

Figure 3.8 (4 of 4)

TERMINAL FUNCTIONS

This section concerns the possibilities of software manipulation of the CRT display. CP/M 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 3.9 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 3.10.

TERMINAL FUNCTION CODES	
Function	Hexadecimal Code
POSITION CURSOR ROW + Offset COL + Offset	1B 3D followed by ROW + 20 followed by COL + 20
CURSOR LEFT (non-destructive backspace)	08
CURSOR DOWN (line feed)	0A
CURSOR RIGHT (non-destructive forward space)	0C
CURSOR UP (reverse line feed)	0B
CURSOR HOME (top left corner)	1E
CLEAR SCREEN and CURSOR HOME	1A or 1B 2A or 1B 3A
CLEAR TO END OF LINE	17 or 1B 54 or 1B 74
CLEAR TO END OF SCREEN	1B 59 or 1B 79
CARRIAGE RETURN	0D
ESCAPE	1B
INSERT LINE	1B 45
INSERT CHARACTER	1B 51
DELETE LINE	1B 52
DELETE CHARACTER	1B 57
HALF INTENSITY OFF	1B 28
HALF INTENSITY ON (Yellow on color CRT)	1B 29
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)
NOTE: Combination of HALF INTENSITY ON/OFF and VIDEO INVERSE ON/OFF create different foreground/background colors on color CRT.	

Figure 3.9

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 3.10

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

1B

46

Function number, using the appropriate value E0 (Function 0) . . . F3 (Function 20)

New function: a string of values 0 . . . 7F, including control characters except 09.

Function number, as above.

For example, the following hexadecimal sequence sets F2 to the function DIR with a concluding carriage return:

1B 46 E1 44 49 52 0D E1

The advantage to the programmer of this method is that there is no need to return to CP/M-80 system level in order to program a Function Key via the CONFIG utility.

THE BIOS PROGRAM

In the Appendix B of this guide you will find the listing of the Basic Input/Output System in Intel® 8080 assembly language. Before attempting to study this listing, it is, however, advisable to read the following two sections with regard to the addresses produced in this assembly.

HOW TO READ THE BIOS PROGRAM

The assembly listing has been produced on the basis of six logically separable aspects of the BIOS software which have been entitled MBIOS, NBIOS, OBIOS, RBIOS, PBIOS and QBIOS. Preceding each of these sections of assembly instructions are lists of the equates relating to the use of constants and symbolic addresses. For example, the symbol for the function code used to clear the screen is "CLRSCN." Under "Control Character Equates" on page 7 (page number appears as three digits at the top of the page) in MBIOS, you will find in line 175 that the actual hex value for this symbolic constant which is to be created at the time of the assembly is 1A. (Compare this value with the appropriate entry in Figure 3.9!). Therefore, the byte definition "DB CLRSCN" in line 307 is effected using the value 1A. An example of a symbolic address equate is present on page 3 in line 73, where "NCON-

OUT,” the address of the routine which displays a character on the screen, is given the value “STBIOS+12.” Line 72 already tells us that STBIOS has the hex value DE00. Therefore, in line 342 (page 14) the assembler replaces “NCONOUT” with the hex value DE0C. (The value 12 without the suffix “H” in line 73 is regarded by the assembler as a decimal and not a hexadecimal number.)

It will have become apparent from reading the assembler instructions used in these examples (lines 307 and 342), that the following format is used:

```
hhhh  object code  label: (if any)  instruction ; comment (if
                                           any)
```

where hhhh represents the hexadecimal address relative to the beginning of the assembly section, the object code is the hex code produced by the assembler, label is a symbolic address, and the instruction is derived from Intel 8080 assembly language. An example where all these components are present is line 183 (the beginning of MBIOS):

```
0000 C32100 BEGIN: JMP BMOVE ; jump move routine
```

At assembly, the symbolic address was interpreted as 0021, i.e. 33 bytes higher in memory than the beginning of this assembly section. Note that addresses in object code are assembled with their two bytes “in reverse order,” that is, the most significant byte of the two has the higher address. For a detailed summary of 8080 assembly language, the user should refer to the NCR CP/M manual. (The fact that the CPU in your machine is a Z-80® microprocessor is of no detriment to the use of 8080 assembly language in the CP/M “ASM” and “DDT” utilities.)

In the examples we have seen so far, it is evident that the assembler had already been provided with a value to use in place of the symbol in the generation of the object code. There are, however, occasions when an assembler cannot replace the symbol at the time of producing a listing such as this one. Instead, the listing shows a zero value in one or two bytes, according to the type of assembly instruction. The circumstances in which a zero value is listed in place of a symbol, is where the symbol definition occurs externally to the section being assembled. An example of this can be found in NBIOS in line 1144 (page 33). Note that the assembler has entered a zero value in two bytes instead of the address “READOP,” and that the list of symbols similarly shows

a zero value for this symbol (page 54). To avoid an error code of unresolved symbol, the assembler had to be told that "READOP" is an "external definition" (page 3). "READOP" appears with an actual value for the first time in OBIOS (see symbol list on OBIOS page 43). It would be tedious to have to check through a number of lists to find this actual value. For this reason, the BIOS link-list is included (Appendix B) immediately following the BIOS listing. This is the list of external definitions which were finally resolved when the sections of BIOS were put together.

Symbolic references within individual sections, including forward references, could be resolved before the assembler produced the listing.

The equate (EQU) statements at the front of each BIOS section give information regarding the use within the program of particular symbols. In addition, the list of symbols following each BIOS section provides the resolution of each non-external symbol. In these lists you will also find the line number of each occurrence of a symbol.

It is important to remember that the BIOS program is hardware related, and that variations in the hardware between models mean that there are different versions of BIOS. For example, the BIOS must carry out slightly different functions in a system with fixed disk drives to those required in a system which uses only flexible disks; a color CRT requires more extensive drive parameters than a monochrome tube. For this reason, the use of absolute machine addresses has been kept to a minimum in the BIOS listing. The reader will note that the starting address of the BIOS in memory is given as hexadecimal DE00. This is the address at which NBIOS is loaded from flexible disk. The remaining sections OBIOS, RBIOS, PBIOS and QBIOS, are loaded after NBIOS concatenated in that order. This means that a hardware determined variation in NBIOS could change the starting addresses of all the other BIOS sections. The following section "Displaying the BIOS Program on the Screen" gives the user useful instructions on how to work with the addresses in the BIOS listing. MBIOS is present in memory only during power-up initialization and is subsequently overwritten by NBIOS.

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 (DDT) which is provided as part of the CP/M operating system. A full description of DDT is contained in the NCR CP/M manual. It suffices here to say that with the aid of DDT, 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 from FBASE upwards, it is only necessary to load the DDT utility.

The first thing the user can check is the starting address of BIOS. CP/M has made this check easy by recording this address in page zero of the memory in bytes 0001H and 0002H. (Remember: Byte 0002H is the most significant byte of the address.) With DDT loaded, the user can enter D0 or L0. The former will produce lines, each of 16 bytes of memory. The latter will disassemble memory into 8080 assembly language. Assuming that the BIOS actually begins at DE00, byte 0001H contains the value 03, while byte 0002H contains the value DE. To obtain the BIOS starting address, the value 3 must be subtracted from the address contained in these two bytes. To check the starting addresses of this and other BIOS sections, the user can refer to the link-list, where these addresses are resolved, and observe the instructions starting at each address.

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 DS, 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. (Already bytes 0003H and 0004H are data and not instruction bytes.) The disassembler does not provide symbols.

THE BIOS ENTRY POINTS

The section "BDOS Entry Functions" demonstrated the range of I/O functions which can be used by loading the C and other registers with entry parameters and calling address 0005H. The BIOS in upper memory includes a similar vector, from which I/O functions can be activated by means of a programmed call to one of twenty-one 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 (see NBIOS, page 10).

CBOOT

(0000)

DE58

Cold start initialization.

WBOOT

(0003)

DE6B

Warm start — BIOS is not reloaded.

CONST

(0006)

EB59

Console status — returns 0FFH in register A if character is ready, otherwise zero.

CONIN

(0009)

EB73

Console character returned in register A. Bit 7 is reset. No return until a character is typed.

CONOUT

(000C)

EB8D

Contents of register C is sent to the console device.

LIST

(000F)

EBCC

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

PUNCH

(0012)

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

READER

(0015)

Reader character returned in register A. 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 C:

0 = drive A . . . 3 = drive D. Register HL 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 BC.

SETSEC**(0021)**

Selects sector number contained in registers BC.

SETDMA**(0024)**

Sets DMA address to contents of BC registers. (Register B must contain the most significant byte of this address.)

The automatic warm boot setting is 0080H.

READ**(0027)**

Using the set drive, track, sector, and DMA address, one disk sector is read. Normally, register A 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. Settings and returns as in READ.

LISTST**(002D)**

Returns status of list device: 0FFH in register A indicates that the device is ready to receive a character.

Useful for background printing.

SECTRAN**(0030)**

Moves sector number in BC to HL.

SPECFUN**(0033)**

Set up parameters for BIOS functions (see NBIOS, page 16).
Non-standard BIOS function.

SELTYP

(0036)

Returns parameters for EXCHANGE utility (see NBIOS, page 47). Non-standard BIOS function.

DISPERO

(0039)

Displays error message. Non-standard BIOS function.

CLRLI1

(003C)

Clears error line. Non-standard BIOS function.

THE BIOS SECTIONS

This section presents a summary of contents of the BIOS listing. Numbers in parentheses refer to pages within the relevant section (M, N, O, P, Q, or R) of the BIOS. The subsequent section "Making Use of the I/O Functions" will present examples of how to make use of the BIOS.

MBIOS:

RAM integrity check and display of error if occurring. Signing on message. Assuming no error occurs, this section is overwritten by NBIOS.

NBIOS:

BDOS initialization (11).

Cold boot and warm boot (12).

Page zero initialization (13).

Disk, track, and sector selection (17-18).

Set DMA address (18).

CONFIG data, including serial interface parameters (23).

Function Key table (24).

Language translation table for the various keyboards (27).

Translation table: key to control character (29).

Data for error messages (30).

Memory addresses for disk information (31).

Read from disk (33, 35ff, 40).

Write to disk, including check for prior allocation (33, 35ff, 40).

Read and write include data transfer to and from host buffer.

Drive selection and disk initialization (39, 41ff).

Parameters for EXCHANGE utility (47).

Data table for different types of disk and deblocking.

OBIOS:

Regulate number of read/write attempts, in accordance with content of retry/restore counters at 020CH and 020DH, relative to the beginning of NBIOS (10).

Disk formatting (12).

Physical read-from and write-to disk (13ff), including check for NCR disk (29) and disk error routine (38).

RBIOS:

Winchester Disk read and write, and status check.

PBIOS:

Keyboard status and input (10).

Punch output, reader input (11).

Scrolling and cursor manipulation (38-54).

Selecting routine according to a control character (26-27, 30-31).

Translation of character in accordance with special keyboard (28).

CRT inversion and intensity (33-35).

Driving the loudspeaker (36-37).

Final output to CRT (55).

QBIOS:

Routines used in acceptance and interpretation of keyboard input (5-10).

Expansion of function keys (11-13).

RS-232 interface: Port addresses (22), status data (22), initialization (23), check printer input status (25), check if printer ready to receive (26), input and output of data (27).

Parallel (Centronics) interface: disable serial interface (29), output character (30), check if printer ready to receive (30).

MAKING USE OF THE I/O SOFTWARE

The CP/M input/output software operates from read/write memory to which the user has full access. Within the limitations indicated in the section "I/O Functions," it is possible to retrieve the system through a programmatic call, even after parts of the I/O system have been overwritten by a user program. 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. In addition, the programmer can refer to the final part of this section, "Tables-Routines-Ports."

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 facilities 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 CONIN. 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. If BIOS entry points have been used, 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. The current vector address is hexadecimal DE00.

SOME I/O EXAMPLES

The following examples are designed to give the user an initial introduction to activating I/O functions at machine code level. In particular, this section cannot provide an extensive guide to programming style. The example instructions are given as hexadecimal machine code written to machine address 100H, i.e. starting immediately after Page Zero. The corresponding Intel 8080 and Zilog Z-80 assembly language instructions are also given. The examples can be assembled using the CP/M ASM utility. (This utility understands only 8080 instructions.) Alternatively, the Set function of the DDT utility can be used for direct input in hexadecimal code. This is particularly useful for very short programs. The Z-80 jump relative (JR) and block transfer (LDD, LDDR, LDI, LDIR) instructions are not used in the examples as there are no single instruction equivalents in 8080 assembly language. They would, of course, execute normally if the user were to enter them via the DDT Set function.

Remember to set suitable breakpoints when using DDT for test runs. Otherwise the CPU will attempt to read the bytes following your program as instructions.

CAUTION

For experimenting with the I/O software, it is advisable to use an additional copy of your CP/M flexible disk. In the event of file damage due to the accidental or incorrect activation of a disk file handling routine, a replacement copy can be made using the master flexible disk.

Sound

This example drives the loudspeaker by using the BIOS CON-OUT function. The tones produced are an ascending scale. The equate statements:

```
BIOS EQU 0DE00H    ; BIOS vector
SPEAK EQU BIOS + 0CH ; offset in vector of CONOUT jump
```

Addr.	Obj. Code	Label	8080	Z-80	comment
0200			ORG 200H	ORG 200H	
0200	0E1B	ACCEPT:	MVI C,1BH	LD C,1BH	
0202	CD0CDE		CALL SPEAK	CALL SPEAK	
0205	0E4D		MVI C,4DH	LD C,4DH	
0207	CD0CDE		CALL SPEAK	CALL SPEAK	
020A	C9		RET	RET	
					; The above subroutine is used by the following
					; main routine to tell the console device that the
					; next two bytes represent frequency and length
					; (see section "Terminal Functions").
0100			ORG 100H	ORG 100H	
0100	3E20		MVI A,20H	LD A,20H	; starting fre-
					; quency
0102	3C	SCALE:	INR A	INC A	; next frequency
0103	FE4B		CPI 4BH	CP 4BH	; end of scale?
0105	CA1B01		JZ DONE	JP Z,DONE	; then terminate
0108	F5		PUSH PSW	PUSH AF	
0109	CD0002		CALL ACCEPT	CALL ACCEPT	
010C	F1		POP PSW	POP AF	
010D	4F		MOV C,A	LD C,A	; frequency
010E	F5		PUSH PSW	PUSH AF	
010F	CD0CDE		CALL SPEAK	CALL SPEAK	
0112	0E29		MVI C,29H	LD C,29H	; length 100ms
0114	CD0CDE		CALL SPEAK	CALL SPEAK	; per tone
0117	F1		POP PSW	POP AF	
0118	C30201		JMP SCALE	JP SCALE	
011B	00	DONE:	NOP	NOP	

Invert

This example uses the CONOUT function to invert the CRT display (dark on light background).

The equate statements:

```
BIOS      EQU 0DE00H    ; BIOS vector
CRTOUT    EQU BIOS+0CH ; offset in vector of CONOUT jump
CBYTE1    EQU 1BH      ; 3 byte control sequence
CBYTE2    EQU 47H      ; for Video Inverse (see
CBYTE3    EQU 34H      ; "Terminal Functions")
```

Addr.	Obj. Code	Label	8080	Z-80	comment
0100			ORG 100H	ORG 100H	
0100	0E1B		MVI C,CBYTE1	LD C,CBYTE1	
0102	CD0CDE		CALL CRTOUT	CALL CRTOUT	
0105	0E47		MVI C,CBYTE2	LD C,CBYTE2	
0107	CD0CDE		CALL CRTOUT	CALL CRTOUT	
010A	0E34		MVI C,CBYTE3	LD C,CBYTE3	
010C	CD0CDE		CALL CRTOUT	CALL CRTOUT	

Running this routine a second time will reset the CRT display contrast to normal if the value of CBYTE3 is changed to 30H.

Dim

This example sets the CRT writing to half intensity. The equate statements:

```
BIOS      EQU 0DE00H    ; BIOS vector
CRTOUT    EQU BIOS+0CH ; offset in vector of CONOUT jump
CBYTE1    EQU 1BH      ; 2 byte control sequence
CBYTE2    EQU 29H      ; for Half Intensity (see "Terminal
                       ; Functions")
```

Addr.	Obj. Code	Label	8080	Z-80	comment
0100			ORG 100	ORG 100	
0100	0E1B		MVI C,CBYTE1	LD C,CBYTE1	
0102	CD0CDE		CALL CRTOUT	CALL CRTOUT	
0105	0E29		MVI C,CBYTE2	LD C,CBYTE2	
0107	CD0CDE		CALL CRTOUT	CALL CRTOUT	

Running this routine a second time will reset the CRT display to normal intensity if the value of CBYTE2 is changed to 28H.

Read Keyboard

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

The equate statements:

```

BIOS      EQU 0DE00H      ; BIOS vector
INKEY     EQU BIOS+9      ; offset in vector of CONIN jump
CRTOUT    EQU BIOS+0CH    ; offset in vector of CONOUT jump
CBYTE1    EQU 1BH        ; 1st byte of control sequence
CBYTEX    EQU 3AH        ; clear screen and cursor home
CBYTE2    EQU 47H        ; 2nd byte of control sequence
CBYTEI    EQU 34H        ; used for inverse
Cbyter    EQU 30H        ; used for reset inverse
ZERO      EQU '0'
NINE      EQU '9'
EOTEXT    EQU '$'        ; end of text sign

```

Addr.	Obj. Code	Label	8080	Z-80	comment
0200			ORG 200H	ORG 200H	
0200	0E1B	INVERS:	MVI C,CBYTE1	LD C,CBYTE1	; produces
0202	CD0CDE		CALL CRTOUT	CALL CRTOUT	; inverse video
0205	0E47		MVI C,CBYTE2	LD C,CBYTE2	
0207	CD0CDE		CALL CRTOUT	CALL CRTOUT	
020A	0E34		MVI C,CBYTEI	LD C,CBYTEI	
020C	CD0CDE		CALL CRTOUT	CALL CRTOUT	
020F	C9		RET	RET	
0210	0E1B	REVERS:	MVI C,CBYTE1	LD C,CBYTE1	; reverts to
0212	CD0CDE		CALL CRTOUT	CALL CRTOUT	; normal video
0215	0E47		MVI C,CBYTE2	LD C,CBYTE2	
0217	CD0CDE		CALL CRTOUT	CALL CRTOUT	
021A	0E30		MVI C,CBYTER	LD C,CBYTER	
021C	CD0CDE		CALL CRTOUT	CALL CRTOUT	
021F	C9		RET	RET	
0220	0E1B	CLSCRN:	MVI C,CBYTE1	LD C,CBYTE1	; two control
0222	CD0CDE		CALL CRTOUT	CALL CRTOUT	; characters for
0225	0E2A		MVI C,CBYTEX	LD C,CBYTEX	; clear screen
0227	CD0CDE		CALL CRTOUT	CALL CRTOUT	; and cursor
022A	C9		RET	RET	; h

Addr.	Obj. Code	Label	8080	Z-80	comment
0100			ORG 100H	ORG 100H	
0100	CD2002		CALL CLSCRN	CALL CLSCRN	
0103	CD1002	NEXT:	CALL REVERS	CALL REVERS	; normal video
0106	CD09DE		CALL INKEY	CALL INKEY	; read key
0109	FE24		CPI EOTEXT	CP EOTEXT	; \$?
010B	CA2701		JZ DONE	JP Z,DONE	; then end
010E	FE30		CPI ZERO	CP ZERO	; if > = 0
0110	DA2001		JC PRINT	JP C,PRINT	; and < = 9
0113	FE39		CPI NINE	CP NINE	; then inverse
0115	CA1B01		JZ INVT	JP Z,INVT	; video
0118	D22001		JNC PRINT	JP NC,PRINT	
011B	F5	INVT:	PUSH PSW	PUSH AF	
011C	CD0002		CALL INVERS	CALL INVERS	
011F	F1		POP PSW	POP PSW	
0120	4F	PRINT:	MOV C,A	LD C,A	; character
0121	CD0CDE		CALL CRTOUT	CALL CRTOUT	; on screen
0124	C30301		JMP NEXT	JP NEXT	
0127	00	DONE:	NOP	NOP	

String

This example displays a character string using the BDOS Function 9. Note that each character is printed until a dollar sign (\$) is encountered. The dollar sign itself is not displayed.

The equate statements:

```

PZERO EQU 0005      ; address of entry point
OUTSTR EQU 09       ; function number
BIOS EQU 0DE00H     ; BIOS vector
CRTOUT EQU BIOS+0CH ; BIOS CONOUT jump
STRING EQU 0200H    ; address of string
CR EQU 0DH          ; control character
LF EQU 0AH          ; control character

```

Addr.	Obj. Code	Label	8080	Z-80	comment
0200			ORG 200H	ORG 200H	
0200	594F552053		DB 'YOU SEE, THE DOLLAR SIGN DOES NOT APPEAR'		
0250			ORG 250H	ORG 250H	
0250	0E0D	CRLF:	MVI C,CR	LD C,CR	; subroutine for
0252	CD0CDE		CALL CRTOUT	CALL CRTOUT	; carriage return
0255	0E0A		MVI C,LF	LD C,LF	; and line feed
0257	CD0CDE		CALL CRTOUT	CALL CRTOUT	
025A	C9		RET	RET	

Addr.	Obj. Code	Label	8080	Z-80	comment
0100			ORG 100H	ORG 100H	
0100	CD5002		CALL CRLF	CALL CRLF	; blank line
0103	110002		LXI D,STRDE	LD HL,STRDE	; string addr. ; in DE
0106	0E09		MVI C,OUTSTR	LD C,OUTSTR	
0108	CD0500		CALL PZERO	CALL PZERO	
010B	CD5002		CALL CRLF	CALL CRLF	; blank line
010E	CD5002		CALL CRLF	CALL CRLF	

Duplicate

This is a sample program which stores keyboard input in memory and duplicates the stored data on the printer as often as you wish. Starting with a clean 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. You may enter very large amounts of data — if this program alone is loaded, you can fill the memory from 300H up to the input/output system (or up to DDT, if you are testing with this utility)!

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

This example makes use of Page Zero functions for reading the keyboard, creating backspaces, and final printing. The BIOS entry points are used for screen clearing, screen line feed, and printer line feed.

The equate statements:

```

BIOS      EQU 0DE00H      ; BIOS vector
CRTOUT    EQU BIOS+0CH    ; BIOS CONOUT jump
BLST      EQU BIOS+0FH    ; BIOS PRINT jump
CBYTE1    EQU 1BH        ; first byte of control sequence
CBYTEX    EQU 3AH        ; clear screen and cursor home
PZERO     EQU 0005        ; address of entry point
INKEY     EQU 01         ; Page Zero read keyboard
OUTTXT    EQU 02         ; Page Zero character to CRT
CR        EQU 0DH        ; Carriage Return
LF        EQU 0AH        ; Line Feed
EOTEXT    EQU '$'        ; end of text sign
STORE     EQU 0300H      ; first byte of storage

```

```

BACKSP EQU 08      ; backspace
SPACE   EQU 20H    ; ASCII space
LSTOUT  EQU 5      ; Page Zero character to printer
UPPERR  EQU 52H    ; upper case R
LOWERR  EQU 72H    ; lower case r

```

Addr.	Obj. Code	Label	8080	Z-80	comment
0200			ORG 200H	ORG 200H	
0200	E5	RFADIN:	PUSH H	PUSH HL	; wait for
0201	0E01		MVI C,INKEY	LD C,INKEY	; and read
0203	CD0500		CALL PZERO	CALL PZERO	; key
0206	E1		POP H	POP HL	
0207	C9		RET	RET	
0208	E5	CRTLF:	PUSH H	PUSH HL	; adds screen line
0209	0E0A		MVI C,LF	LD C,LF	; feed to CR
020B	CD0CDE		CALL CRTOUT	CALL CRTOUT	; detected
020E	E1		POP H	POP HL	; in main routine
020F	C9		RET	RET	
0210	E5	PRTCLR:	PUSH H	PUSH HL	; CR during
0211	0E0D		MVI C,CR	LD C,CR	; print
0213	CD0FDE		CALL BLST	CALL BLST	
0216	E1		POP H	POP HL	
0217	E5	PRTLFL:	PUSH H	PUSH HL	; LF during
0218	0E0A		MVI C,LF	LD C,LF	; print
021A	CD0FDE		CALL BLST	CALL BLST	
021D	E1		POP H	POP HL	
021E	C9		RET	RET	
021F	E5	BLANK:	PUSH H	PUSH HL	; backspace
0221	0E02		MVI C,OUTTXT	LD C,OUTTXT	; and erase
0223	1E20		MVI E,SPACE	LD E,SPACE	; one character
0224	CD0500		CALL PZERO	CALL PZERO	
0227	1E08		MVI E,BACKSP	LD E,BACKSP	
0229	0E02		MVI C,OUTTXT	LD C,OUTTXT	
022B	CD0500		CALL PZERO	CALL PZERO	
022E	E1		POP H	POP HL	
022F	C9		RET	RET	
0230	0E1B	CLSCRN:	MVI C,CBYTE1	LD C,CBYTE1	; clear screen
0232	CD0CDE		CALL CRTOUT	CALL CRTOUT	; and home cur- ; sor
0235	0E3A		MVI C,CBYTEX	LD C,CBYTEX	
0237	CD0CDE		CALL CRTOUT	CALL CRTOUT	
023A	C9		RET	RET	

Addr.	Obj. Code	Label	8080	Z-80	comment
0100			ORG 100H	ORG 100H	
0100	CD3002	START:	CALL CLSCRN	CALL CLSCRN	
0103	210003		LXI H,STORE	LD HL,STORE	
0106	CD0002	NEXT:	CALL READIN	CALL READIN	
0109	77		MOV M,A	LD (HL),A	; key code in ; storage
010A	23		INX H	INC HL	; point to next ; byte
010B	FE08		CPI BACKSP	CP BACKSP	
010D	C21801		JNZ NOBACK	JP NZ,NOBACK	; if backspace ; remove
0110	2B		DCX H	DEC HL	
0111	2B		DCX H	DEC HL	; character from ; screen
0112	CD1F02		CALL BLANK	CALL BLANK	; and storage
0115	C30601		JMP NEXT	JP NEXT	; if CR provide
0118	FE0D	NOBACK:	CPI CR	CP CR	; line feed on ; screen
011A	C22301		JNZ NOLF	JP NZ,NOLF	; but not in ; storage
011D	CD0802		CALL CRTLF	CALL CRTLF	
0120	C30601		JMP NEXT	JP NEXT	
0123	FE24	NOLF:	CPI EOTEXT	CP EOTEXT	; go to key input
0125	C20601		JNZ NEXT	JP NZ,NEXT	; if not eotext.
0128	210003	PRINT:	LXI H,STORE	LD HL,STORE	; reset storage ; pointer
012B	7E	NEXTP:	MOV A,M	LD A,(HL)	; fetch byte
012C	FE24		CPI EOTEXT	CP EOTEXT	; until eotext ; sign
012E	CA4601		JZ DONE	JP Z,DONE	
0131	0E05		MVI C,LSTOUT	LD C,LSTOUT	; call Page Zero
0133	5F		MOV E,A	LD E,A	; print routine
0134	E5		PUSH H	PUSH HL	
0135	CD0500		CALL PZERO	CALL PZERO	
0138	E1		POP H	POP HL	
0139	7E		MOV A,M	LD A,(HL)	; fetch same byte
013A	FE0D		CPI CR	CP CR	; if CR add LF
013C	C24201		JNZ NONLIN	JP NZ,NONLIN	
013F	CD1702		CALL PRTLF	CALL PRTLF	
0142	23	NONLIN:	INX H	INC HL	; point to next ; byte
0143	C32B01		JMP NEXTP	JP NEXTP	
0146	CD1002	DONE:	CALL PRTCLR	CALL PRTCLR	; blank lines
0149	CD1002		CALL PRTCLR	CALL PRTCLR	
014C	CD1002		CALL PRTCLR	CALL PRTCLR	
014F	CD1002		CALL PRTCLR	CALL PRTCLR	
0152	CD0002		CALL READIN	CALL READIN	; read keyboard
0155	FE52		CPI UPERR	CP UPERR	; print again if
0157	CA2801		JZ PRINT	JP Z,PRINT	; required key ; has
015A	FE72		CPI LOWERR	CP LOWERR	; been pressed.
015C	CA2801		JZ PRINT	JP Z,PRINT	

Disk File Random Access

The examples conclude with a more extensive program of file creation and retrieval. The program listed below performs the simple function of reading or writing random records upon command from the terminal. Given that the program has been created, assembled, and placed into a file labeled RANDOM.COM, the CCP level command

```
RANDOM X.DAT
```

starts the test program. The program looks for a file by the name X.DAT (in this particular case) and, if found, proceeds to prompt the console for input. If not found, the file is created before the prompt is given. Each prompt takes the form

```
next command?
```

and is followed by operator input, terminated by a carriage return. The input commands take the form

```
nW nR Q
```

where n is an integer value in the range 0 to 65535, and W, R, and Q are simple command characters corresponding to random write, random read, and quit processing, respectively. If the W command is issued, the RANDOM program issues the prompt

```
type data:.
```

The operator then responds by typing up to 127 characters, followed by a carriage return. RANDOM then writes the character string into the X.DAT file at record n. If the R command is issued, RANDOM reads record number n and displays the string value at the console. If the Q command is issued, the X.DAT file is closed, and the program returns to the CCP. In the interest of brevity, the only error message is

```
error, try again.
```

The program begins with an initialization section where the input file is opened or created, followed by a continuous loop at the label "ready" where the individual commands are interpreted. The default file control block at 005CH and the default buffer at 0080H are used in all disk operations. The utility subroutines then

follow, which contain the principal input line processor, called "readc." This particular program shows the elements of random access processing, and can be used as the basis for further program development.

The equate statements:

```

REBOOT EQU 0000H      ; system reboot
PZERO  EQU 0005H      ; Page Zero entry point
INKEY  EQU 1          ; console input function
CRTOUT EQU 2          ; console output function
OUTSTR EQU 9          ; print string until '$'
RSTRING EQU 10        ; read console buffer
VERSION EQU 12        ; return version number
OPENF  EQU 15         ; file open function
CLOSEF EQU 16         ; close function
MAKEF  EQU 22         ; make file function
READR  EQU 33         ; read random
WRITER EQU 34         ; write random

FCB     EQU 005CH     ; default file control block
RANREC  EQU FCB+33    ; random record position
RANOVF  EQU FCB+35    ; high order (overflow) byte
BUFF    EQU 0080H    ; buffer address

CR      EQU 0DH       ; carriage return
LF      EQU 0AH       ; line feed

```

Addr.	Obj. Code	Label	8080	Z-80	comment
Load SP, Set-Up File for Random Access					
0100	31BC02		ORG 100H LXI SP,STACK	ORG 100H LD SP,STACK	
		:			
		:			version 2.0 or better?
0103	0E0C		MVI C,VERSION	LD C,VERSION	
0105	CD0500		CALL PZERO	CALL PZERO	
0108	FE20		CPI 20H	CP 20H	
010A	D21601		JNC VERSOK	JP NC,VERSOK	
		:			bad version, message and go back
010D	111800		LXI D,BADVER	LD DE,BADVER	
0110	CDDA01		CALL PRINT	CALL PRINT	
0113	C30000		JMP REBOOT	JP REBOOT	
		:			

Addr.	Obj. Code	Label	8080	Z-80	comment
		VERSOK:			
		;			correct version for random access
0116	0EOF		MVI C,OPENF	LD C,OPENF	
0118	115C00		LXI D,FCB	LD DE,FCB	
011B	CD0500		CALL PZERO	CALL PZERO	
011E	3C		INR A	INC A	; err 255
011F	C23701		JNZ READY	JP NZ,READY	; becomes 0
		;			cannot open file, so create it
0122	0E16		MVI C,MAKEF	LD C,MAKEF	
0124	115C00		LXI D,FCB	LD DE,FCB	
0127	CD0500		CALL PZERO	CALL PZERO	
012A	3C		INR A	INC A	; err 255
012B	C23701		JNZ READY	JP NZ,READY	; becomes 0
		;			
		;			cannot create file, directory full
012E	113A02		LXI D,NOSPACE	LD DE,NOSPACE	
0131	CDDA01		CALL PRINT	CALL PRINT	
0134	C30000		JMP REBOOT	JP REBOOT	

Loop Back to Ready After Each Command

		;			
		READY:			
		;			file is ready for processing
		;			
0137	CDE501		CALL READCOM	CALL READCOM	;next com-
					;mand
013A	227D00		SHLD RANREC	LD(RANREC),HL	;store input
					;record #
013D	217F00		LXI H,RANOVF	LD HL,RANOVF	
0140	3600		MVI M,0	LD (HL),0	
0142	FE51		CPI 'Q'	CP 'Q'	; quit
0144	C25601		JNZ NOTQ	JP NZ,NOTQ	
		;			
		;			quit processing, close file
0147	0E10		MVI C,CLOSEF	LD C,CLOSEF	
0149	115C00		LXI D,FCB	LD DE,FCB	
014C	CD0500		CALL PZERO	CALL PZERO	; err 255
014F	3C		INR A	INC A	; becomes 0
0150	CAB901		JZ ERROR	JP Z,ERROR	
0153	C30000		JMP REBOOT	JP REBOOT	

End of Quit Command, Process Write

		NOTQ:			
		;			not the quit command, random write?
0156	FE57		CPI 'W'	CP 'W'	
0158	C28901		JNZ NOTW	JP NZ,NOTW	
		;			
		;			this is a random write, fill buffer until cr
015B	114D02		LXI D,DATMSG	LD DE,DATMSG	
015E	CDDA01		CALL PRINT	CALL PRINT	
0161	0E7F		MVI C,127	LD C,127	; max character
0163	218000		LXI H,BUFF	LD HL,BUFF	; destination

Addr.	Obj. Code	Label	8080	Z.80	comment
		RLOOP:	;read next character		
0166	C5		PUSH B	PUSH BC	
0167	E5		PUSH H	PUSH HL	
0168	CDC201		CALL GETCHR	CALL GETCHR	; char to reg A
016B	E1		POP H	POP HL	
016C	C1		POP B	POP BC	
016D	FE0D		CPI CR	CP CR	; end of line?
016F	CA7801		JZ ERLOOP	JP Z,ERLOOP	
		:	not end, store character		
0172	77		MOV M,A	LD (HL),A	
0173	23		INX H	INC HL	; next to fill
0174	0D		DCR C	DEC C	
0175	C26601		JNZ RLOOP	JP NZ,RLOOP	; end of buffer?
		ERLOOP:	;		
		:	end of read loop, store 00		
0178	3600		MVI M,0	LD (HL),0	
		:	;		
		:	write the record to selected record number		
017A	0E22		MVI C,WRITER	LD C,WRITER	
017C	115C00		LXI D,FCB	LD DE,FCB	
017F	CD0500		CALL PZERO	CALL PZERO	
0182	B7		ORA A	OR A	; error code 0?
0183	C2B901		JNZ ERROR	JP NZ,ERROR	
0186	C33701		JMP READY	JP READY	; another record
End of Write Command, Process Read					
		NOTW:	;		
		:	not a write command, read record?		
0189	FE52		CPI 'R'	CP 'R'	
018B	C2B901		JNZ ERROR	JP Z,ERROR	
		:	;		
		:	read reandom record		
018E	0E21		MVI C,READR	LD C,READR	
0190	115C00		LXI D,FCB	LD DE,FCB	
0193	CD0500		CALL PZERO	CALL PZERO	
0196	B7		ORA A	OR A	; return code 0?
0197	C2B901		JNZ ERROR	JP NZ,ERROR	
		:	;		
		:	read was successful, write to console		
019A	CDCF01		CALL CRLF	CALL CRLF	; new line
019D	0E80		MVI C,128	LD C,128	; max chars
019F	218000		LXI H,BUFF	LD HL,BUFF	; point to next
		WLOOP:	;		
01A2	7E		MOV A,M	LD A,(HL)	; next char
01A3	23		INX H	INC HL	; point to next
01A4	E67F		ANI 7FH	AND 7FH	; mask bit 7
01A6	CA3701		JZ READY	JP NZ,READY	; another com- ; mand
01A9	C5		PUSH B	PUSH BC	
01AA	E5		PUSH H	PUSH HL	
01AB	FE20		CPI ''	CP ''	; allowable char?
01AD	D4C801		CNC PUTCHR	CALL NC,PUTCHR	; only if allowa- ; ble

Addr.	Obj. Code	Label	8080	Z-80	comment
01B0	E1		POP H	POP HL	
01B1	C1		POP B	POP BC	
01B2	0D		DCR C	DEC C	; counter
01B3	C2A201		JNZ WLOOP	JP Z,WLOOP	
01B6	C33701		JMP READY	JP READY	

End of Read Command, All Errors End Up Here

```

;
ERROR:
01B9 115902 LXI D,ERRMSG LD DE,ERRMSG
01BC CDDA01 CALL PRINT CALL PRINT
01BF C33701 JMP READY JP READY

```

Utility Subroutines for Console I/O

```

GETCHR:
;read next console character to a
01C2 0E01 MVI C,INKEY LD C,INKEY
01C4 CD0500 CALL PZERO CALL PZERO
01C7 C9 RET RET

;
PUTCHR:
;write character from a to console
01C8 0E02 MIV C,CRTOUT LD C,CRTOUT ; character
01CA 5F MOV E,A LD E,A ; to screen
01CB CD0500 CALL PZERO CALL PZERO
01CE C9 RET RET

;
CRLF:
;send carriage return line feed
01CF 3E0D MVI A,CR LD A,CR
01D1 CDC801 CALL PUTCHR CALL PUTCHR
01D4 3E0A MVI A,LF LD A,LF
01D6 CDC801 CALL PUTCHR CALL PUTCHR
01D9 C9 RET RET

;
PRINT:
;print the buffer addressed by de until $
01DA D5 PUSH D PUSH DE
01DB CDCF01 CALL CRLF CALL CRLF
01DE D1 POP D POP DE
01DF 0E09 MVI C,OUTSTR LD C,OUTSTR
01E1 CD0500 CALL PZERO CALL PZERO
01E4 C9 RET RET

;
READCOM:
;read the next command line to the conbuf
01E5 116B02 LXI D,PROMPT LD DE,PROMPT
01E8 CDDA01 CALL PRINT CALL PRINT ; 'command?'
01EB 0E0A MVI C,RSTRING LD C,RSTRING
01ED 117A02 LXI D,CONBUF LD DE,CONBUF
01F0 CD0500 CALL PZERO CALL PZERO

```

Addr.	Obj. Code	Label	8080	Z-80	comment
					; command line is present, scan it
01F3	210000		LXI H,0	LD HL,0	
01F6	117C02		LXI D,CONLIN	LD DE,CONLIN	; command line
01F9	1A	READC:	LDAX D	LD A,(DE)	; command char
01FA	13		INX D	INC DE	; point to next ; char
01FB	B7		ORA A	OR A	; zero char?
01FC	C8		RZ	RET Z	
					; not zero, numeric?
01FD	D630		SUI '0'	SUB '0'	
01FF	FE0A		CPI 10	CP 10	
0201	D21302		JNC ENDRD	JP NC,ENDRD	; i.e. if not ; numeric
					; add-in next digit
0204	29		DAD H	ADD HL,HL	; *2
0205	4D		MOV C,L	LD C,L	
0206	44		MOV B,H	LD B,H	; bc = value *2
0207	29		DAD H	ADD HL,HL	; *4
0208	29		DAD H	ADD HL,HL	; *8
0209	09		DAD B	ADD HL,BC	; *10
020A	85		ADD L	ADD A,L	; + digit
020B	6F		MOV L,A	LD L,A	
020C	D2F901		JNC READC	JP NC,READC	; another char
020F	24		INR H	INC H	
0210	C3F901		JMP READC	JP READC	
		ENDRD:			
					; end of read, restore value in a
0213	C630		ADI '0'	ADD A,'0'	
0215	FE61		CPI 'A'	CP 'a'	; translate
0217	D8		RC	RET C	; case?
					; lower case, mask lower case bits
0218	E65F		ANI 01011111B	AND 01011111B	
021A	C9	RET	RET		

Areas for data storage required by the program:

String Data Area for Console Messages

021B	536F72	BADVER:	DB	'Sorry, you need cp/m version 2\$'
023A	4E6F20	NOSPACE:	DB	'No directory space\$'
024D	547970	DATMSG:	DB	'Type data: \$'
0259	457272	ERRMSG:	DB	'Error, try again \$'
026B	4E6578	PROMPT:	DB	'Next command? \$'
				;

Fixed and Variable Data Area

```
027A 21      CONBUF: DB      CONLEN      ; length of console buffer
027B        CONsiz: DS      1          ; resulting size after read
027C        CONLIN: DS     32         ; length 32 buffer
0021 =      CONLEN EQU    $-CONsiz
           ;
029C        DS      32          ; 16 level stack
           STACK:
02BC        END
```

INTERFACING PRINTERS

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

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

NCR DECISION MATE V

PRINTER

1. Printer sets XON signal to enable computer to transmit data.
2. Transmission is enabled, so data are transmitted bit by bit via the TxD line.
3. When the printer buffer is nearly (typically 3/4) full, an XOFF signal is generated.
4. The computer waits with further datawhile the printer empties its its buffer.
5. When the buffer is empty, XON is once again generated.
6. Data transmission is once again enabled.

The XOFF status is equivalent to 13H being read IN at port 60. Otherwise XON is assumed. The DTR and DSR lines are connected together inside the serial printer interface kit. In addition CTS and

RTS should be connected together. Both these combinations and the CD line should be at +12V (i.e. ON).

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

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 K211 or K212 kit.

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

A 2651 is used not only for the serial printer interface, but also for the serial communications interface kit (K211, see Hardware Description). Figure 3.11 summarizes the actual port addresses used by these interfaces.

2651 REGISTER ADDRESSING						
Port (Hex) K212 K211 K213	Signals Required *				Function	
	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 - R/W.

Figure 3.11

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.

TABLES-ROUTINES-PORTS

Tables

The following areas of data storage are contained in the BIOS area. The numbers in parentheses following the BIOS section designations refer to line numbers in the BIOS program listing contained in the Appendix.

N(210)

The BIOS entry points: a vector of unconditional jump instructions.

N(450)

The final machine addresses for the BIOS functions.

N(710)

Addresses of NBIOS tables; I/O Byte data; number of flexible disk drives; retry/restore counters; system recovery parameters; standard parameters for serial I/O.

N(760)

Table of Function Key functions.

N(880)

Language translate table for specific keyboards.

N(960)

Keyboard translate table for hex values >7F.

N(1005)

Error messages.

N(1030)

Memory areas used for disk information.

N(1570)

Disk format data for NCR SS-DD, NCR DS-DD, non NCR, and Hard Disk.

O(1360)

FDC Command and Error storage.

O(1370)

Parameter block for sector one read.

O(1390)

Disk information storage.

P(1070)

Data table of CRT control functions.

P(2080)

Storage area for video data.

Q(630)

Storage area for Function Key data.

Q(940)

Serial/parallel interface active and XOFF flags.

Routines

This section presents tables of cross references regarding CALL instructions (including conditional calls) to routines external to the calling section. Calls to the basic arithmetic routines contained in NBIOS (MULT, MOVE, MOVEL, SUBHLDE, COMDEHL, CMPRE, OFST) are not included.

called by NBIOS			
ROUTINE	LOCATION IN BIOS	ROUTINE	LOCATION IN BIOS
CHECKOS	O	READID	O
CONIN	P	READSEC1	O
CONOUT	P	RDTRAK	O
CREAD	O	RESET	O
DISKIO	O	SECSET	O
DMASET	O	TRKSET	O
DSKERR	O	WINIT	P
DSKSET	O	WRCUPO	P

Figure 3.12 (1 of 4)

called by OBIOS	
ROUTINE	LOCATION IN BIOS
CLOSE	N
CLRLIN	N
CONIN	P
DISPERR	N
DSKGET	N
FIXINIT	N
FLUSH	N

Figure 3.12 (2 of 4)

called by PBIOS	
ROUTINE	LOCATION IN BIOS
CRLIN	N
GETLPOS	Q
KBDINIT	Q

Figure 3.12 (3 of 4)

called by QBIOS	
ROUTINE	LOCATION IN BIOS
ROUTE	P
TRNCHR	P

Figure 3.12 (4 of 4)

Ports

The following is a summary of the available I/O ports used by the BIOS. The following information regarding each port is given:

- Hexadecimal port number as used in assembler IN and OUT instructions, and whether used as Input or Output Port by the BIOS.
- The BIOS symbolic name for the port and the BIOS routines using it.
- Additional information regarding BIOS use of the port.

CAUTION

The ports in your NCR DECISION MATE V are used not only by your operating system, but also by the firmware which becomes active at power up. Under no circumstances should you attempt to make use of IN or OUT (including block transfer) instructions at ports which are connected to Timer functions, otherwise permanent damage to your computer may result. A detailed map of the NCR DECISION MATE V ports is given at the end of this section (Figure 3.13). Suggestions regarding the use of GDC ports are given in the section "Graphics."

IN 13

SYSSTA

Used by ID (OBIOS), DMA01 (OBIOS), FDCIN1 (OBIOS), MOTORCK (OBIOS), DSKERR (OBIOS).

ID and FDCIN1 use this port to check and wait for an interrupt from the disk controller. The interrupt is only valid if bit 3 in register A is set following the IN instruction. The routine will then proceed to read the disk identity.

DMA01 and DSKERR check bit 2. If bit 2 is not set, then the disk is not ready.

MOTORCK checks bit 0. If bit 0 is set, then the motor is not yet switched on.

OUT 14

MOTORON

Used by MOTORCK (OBIOS).

MOTORCK uses bit 0 to turn the motor on.

ZUM DM5-MONITOR MIT

eingeschalteter ROM: JP AAC

OUT 26

COAD

Used by DMA1 (OBIOS).

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

OUT 27

COTC

Used by DMA1 (OBIOS).

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

OUT 2A

DMAMB

Used by DMA1 (OBIOS) and DMA01 (OBIOS).

Bits 0 and 1 are set, other bits reset, to enable the FDC channel following initialization of DMA.

To disable the FDC channel, bits 0, 1, and 2 are set.

OUT 2B

DMAMO

Used by DMA1 (OBIOS) to set the DMA mode.

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

IN 40

RDKEY

Used by BELL (PBIOS), KBDINIT (QBIOS), and PKEYYIN (QBIOS).

Reads a character from the keyboard.

IN 41

RSKEY

Used by BELL (PBIOS), KBDINIT (QBIOS), and PKEYYST (QBIOS).

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

OUT 41

KBELL

Used by BELL (PBIOS). Also KCOUNT used by KBDINIT (QBIOS).

Drives the loudspeaker. If value Reg. A = 1, then it constitutes an instruction to return the country code during keyboard initialization.

IN 50

DSTAT

Used by FDCRD1 (OBIOS) and DMA01 (OBIOS).

Bit 7 set indicates that disk is ready.

OUT 51

DCOMD

Used by DHOME (OBIOS), FDCWAIT (OBIOS), FDCINI1 (OBIOS).

Used in the transmission of disk, head, and track number to the disk controller.

IN 51

FDCRA

Used by FDCINI1 (OBIOS) and GETBY (OBIOS) to read in information from the disk controller.

IN 60

SPRDATA

Used by SRLINST (QBIOS) and SRLIN (QBIOS).

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

OUT 60

PBDA

Used by PRTOUT (QBIOS) as output port for parallel data transmission.

IN 61

SPRSTAT

Used by SRLINST (QBIOS) and SRLPRST (QBIOS); also PBSTA used by PRTSTAT (QBIOS).

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 set indicates a parity error, and bit 4 set indicates an overrun. Bit 1 set is used to indicate that the transmitter is ready.

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

IN 63, OUT 67

SPRCOM, SPWCOM

Reads and transmits error information.

OUT 63

PBCOM

Used by PIOINIT to initialize the parallel interface.

OUT 64

SPWDATA

Used by SRLCOUT for serial output of data.

OUT 66

SPWMODE

Used by SIOINIT to initialize the serial interface. The first of the two output commands determines stop bits, parity, and character length, using the data stored in NBIOS (line 710, see "Tables"). The second command determines the baud rate.

OUT 67

SPWCOM

Used to enable transmitter and receiver in the serial interface.

IN A0

GDCSTA

Used in console I/O to determine whether the graphic display unit 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 graphic display. Bit 3 set means that drawing is actually being carried out.

OUT A0, OUT A1

GDPAR, GDCOM

Used for output of control and printable data to the graphic display in QBIOS.

IN C0

DATA

Used in RBIOS for block input of data from the Winchester disk controller (256 bytes at a time).

OUT C0

DATA

Used in RBIOS for block output of data to the Winchester disk controller (256 bytes at a time).

IN C1

ERROR

Used in RBIOS for detailed definition of 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 and is considered to be a fatal error.

OUT C2

SECNT

Used in RBIOS to set the sector number during formatting.

IN C3

SECNO

Used in RBIOS to check whether disk controller is ready.

OUT C3

SECNO

Used in RBIOS to set a sector number. Output 55H indicates that disk controller is ready.

IN C4

CYLLO

Used in RBIOS to check whether disk controller is ready.

OUT C4

CYLLO

Used in RBIOS to set a cylinder number. Output 0AAH is used for checking purposes.

OUT C5

CYLHI

Used in RBIOS to set the higher order part of the cylinder number.

OUT C6

SDH

Used in RBIOS to give the Winchester disk controller information regarding drive, head, sector size, and error checking. All this control information is passed in a single output instruction (see RBIOS, page 10).

IN C7

Used in RBIOS to accept 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.

OUT C7

Used in RBIOS to select the read (20H) or write (30H) function.

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.

OUT D0

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

Figure 3.13 provides a detailed map of the NCR DECISION MATE V ports. Before referring to this figure, you must read the cautionary note which immediately precedes the list of ports used by the BIOS.

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

Figure 3.13 (1 of 2)

NOTE: In Figure 3.13, the numbers prefixed with K refer to kits available for the NCR DECISION MATE V. A switchable RS-232C interface (K801) which can use any of the IFSEL codes, is also available. When using this kit, you should bear in mind that both hardware and software strapping are required.

This applies also to K215, K803, K804, and K806, as these kits are switchable in the same way. You should therefore ensure that your software can use all the IFSEL codes. The IFSEL codes given in Figure 3.13 for these kits constitute default suggestions.

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
	D I A G N O S E R							
1								
2	DMA: R-STATUS W-COMMAND	DMA: W-REQ. REG.	DMA: W-FDC ENABLE	DMA: W-MODE	DMA: CLR POINTER	DMA: R-MASTER CLEAR	DMA: CLR MASK REG.	DMA: W-ALL MASK BITS
3	IFSEL 2B K804							
4								
5								
6	IFSEL 0B							
7	IFSEL 1B							
8								
9								
A								
B	IFSEL 3B K600							
C	IFSEL 4B K803							
D								
E								
F								

Figure 3.13 (2 of 2)

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

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

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 3.14 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 3.14

COLOR

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 absolute memory address F345.

Foreground and background colors are determined by the six most significant bits of the attribute byte (see Figure 3.15). Bit one set activates video blinking.

If you are using one of the special versions of the operating system for either Arabic or the Dead-Key facility, you should also refer to the information immediately following the BIOS link-list at the end of Appendix B.

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 3.15

GRAPHICS

The operating system software provides you with full access to the character set of your NCR DECISION MATE V. The parameters used in the generation of the CRT display are contained in a 32 KB RAM (96 KB 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 0A1H instruction). Data and other parameters following a particular command are received at port A0. Status information can be read at port A0 (IN 0A0H instruction), and data from the GDC can be read via port A1.

This section deals with the aspects of programming the GDC which relate to its environment in your NCR DECISION MATE V.

Following this, you will find a sample programming session consisting of graphic producing routines which you may wish to adapt and expand for your own applications.

THE GRAPHICS DISPLAY CONTROLLER

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

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

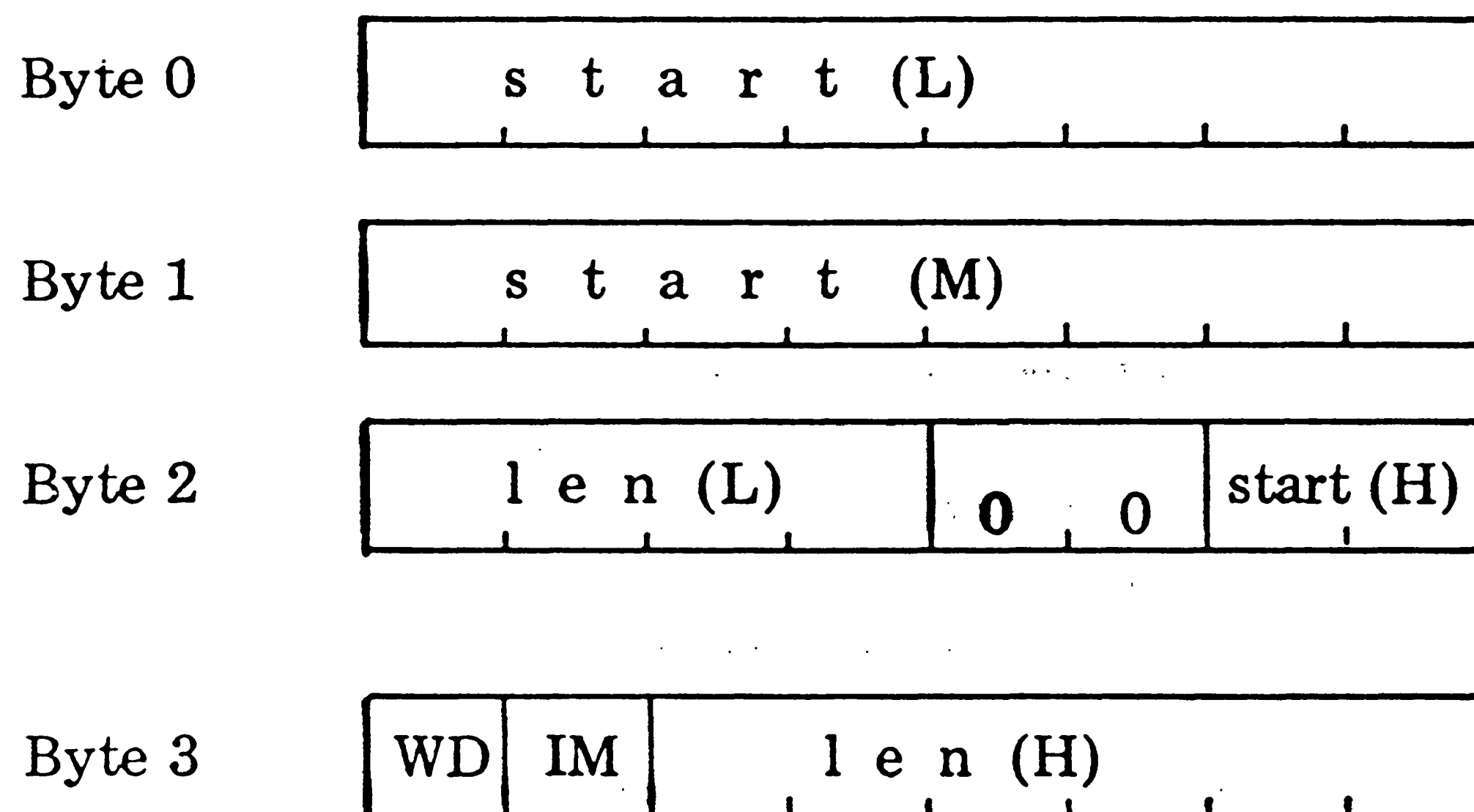
The GDC has two basic modes of operation, namely the Character Mode and the Mixed (Graphics and Character) Mode. The power-up initialization procedure automatically sets the Mixed Mode, as this results in the most efficient non-graphic screen writing in the NCR DECISION MATE V hardware environment. To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

Z-80 assembly language. You may also wish to refer to the manufacturer's description of the PD7220-1 integrated circuit.

The Parameter RAM

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

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



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

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

Bytes 8-15: this area can be used for storing a bit-mapped graphic character in an 8 x 8 pixel format. Upon execution of the appropriate drawing instruction, this area of the parameter RAM is scanned from the least significant bit of byte 15 towards its most significant bit. Scanning then continues from the most significant

bit of byte 14 towards its least significant bit, and so on. If the area to be filled by the parameter RAM is greater than the 8-pixel square, a further subset of the RAM is transmitted to the CRT. If the screen area to be filled is smaller than the 8-pixel square, only a subset of the parameter RAM will appear. Later in this section, you can read how to determine the area on the CRT to be filled, and how to create a slanting (italics) effect.

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

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

GDC Status Information

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

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

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

Bit 2: when set, this bit indicates that the FIFO buffer is empty. It is not necessary, nor desirable, 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).

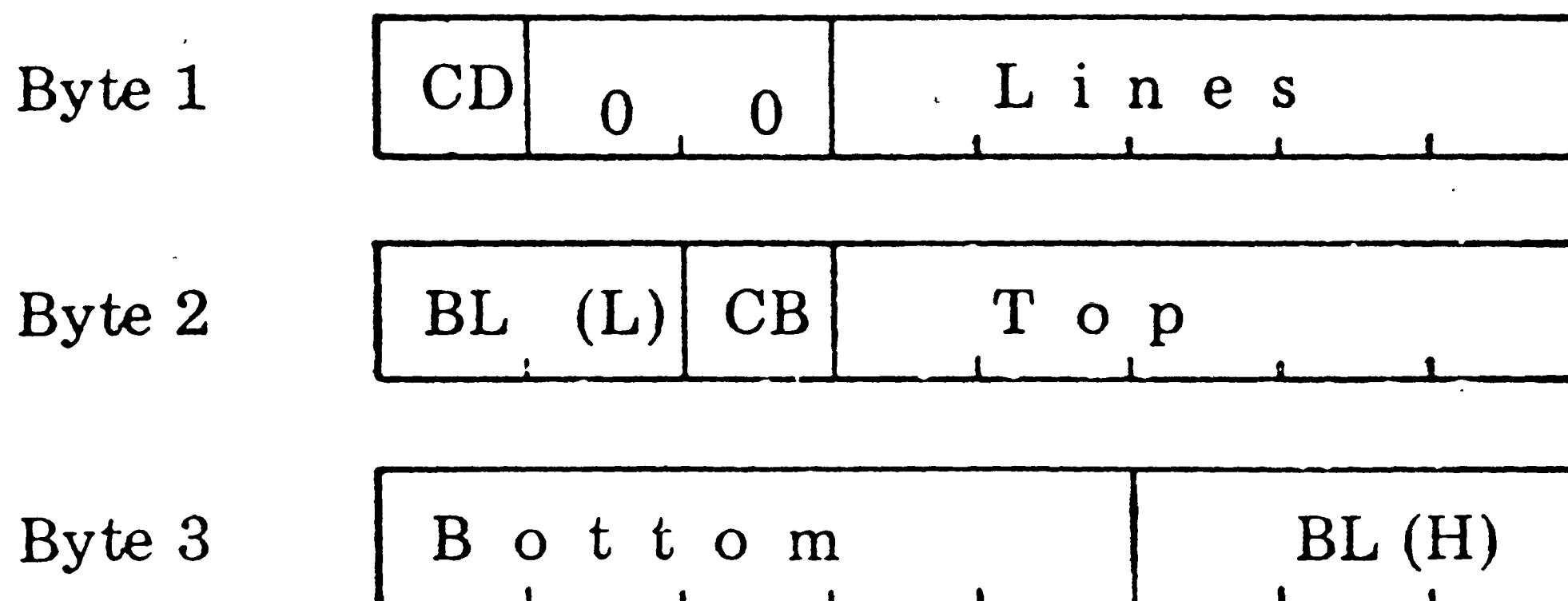
The output parameters are the same as those for the reset command. However, Sync does not reset the GDC or activate idle mode.

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.



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.

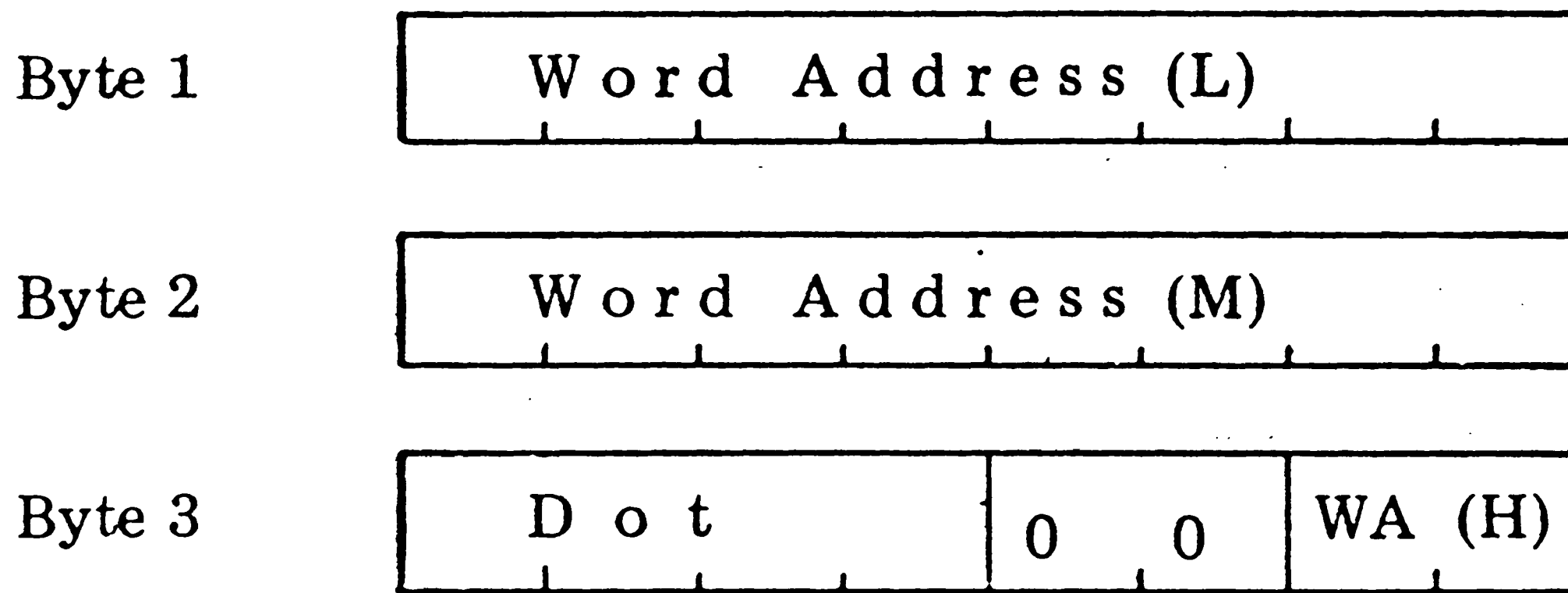
Start Display — Command byte: 6BH, no parameters.

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.

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

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

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

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

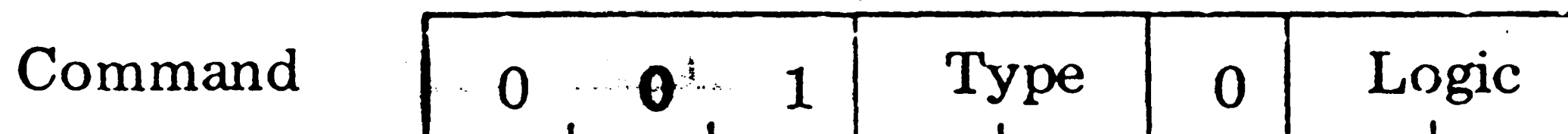
Pitch — Command byte: 47H.

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

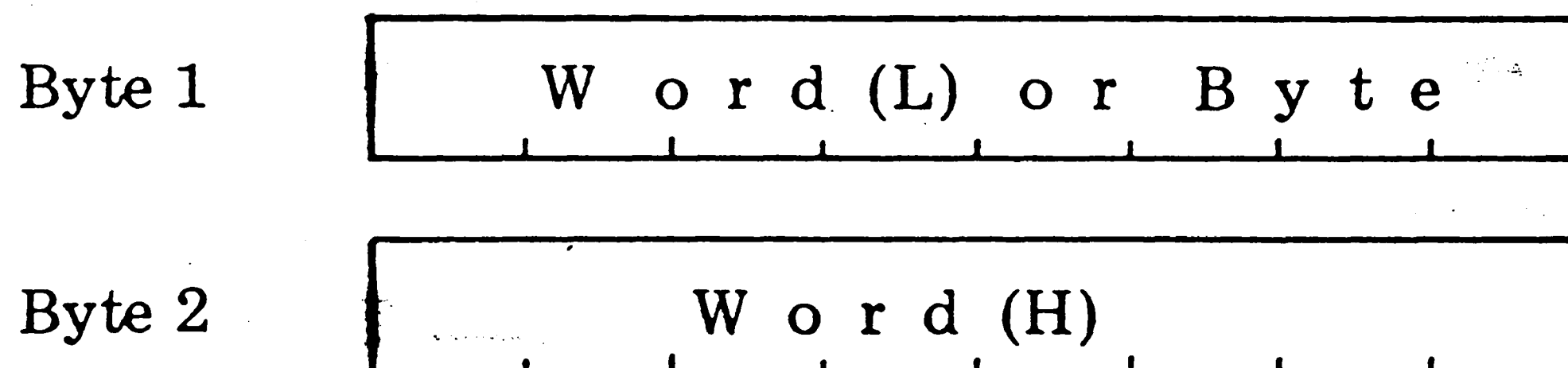
Write Data — This command is an instruction to the GDC to write one word or byte of data into display memory. Following

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

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



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:



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

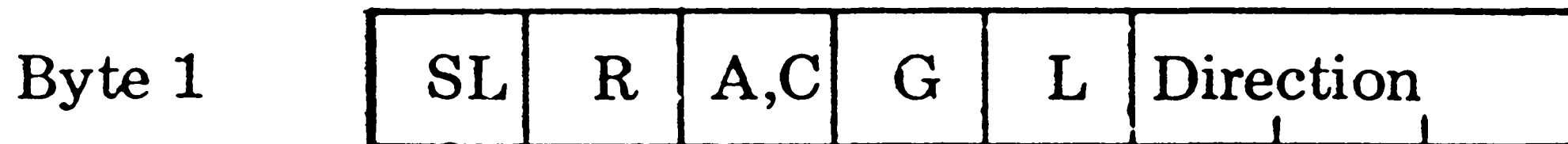
The Write Data command must be preceded by a Figure command (only the first three bytes are required, see Figure).

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.

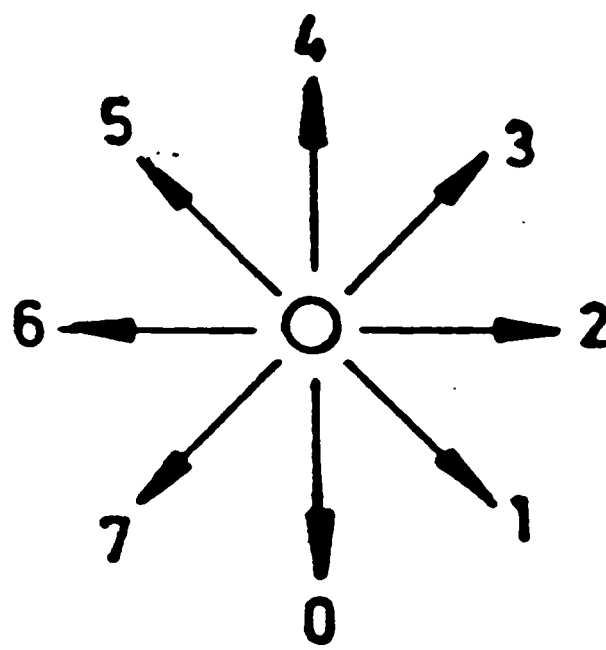
Figure – This command, using as many as 11 parameter bytes, is used for specifying whether individual dot or figure drawing is to take place, and in the latter case, it specifies the figure to be drawn. Beyond this, it is also used for determining the direction of activity for any screen writing. DMA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

Command byte: 4CH.

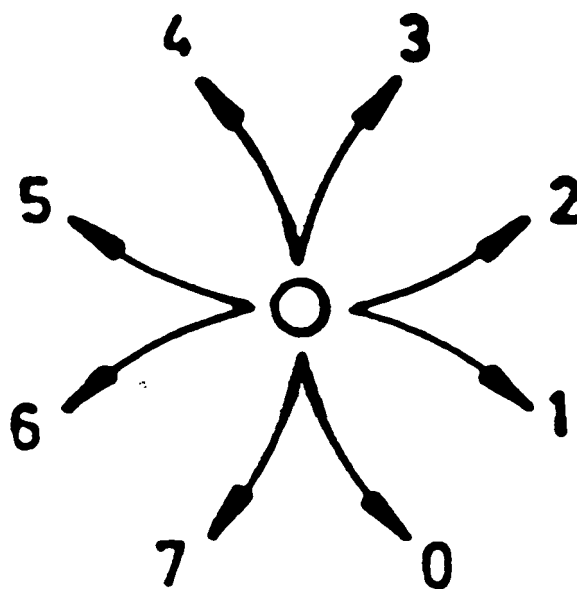


The significance of the individual bits of byte 1 is as follows. SL = slanted graphics character, R = rectangle drawing, A,C = arc or circle drawing, G = graphics character, L = line drawing. None of these bits set denotes individual pixel drawing, character screen writing or reading, or a DMA transfer.

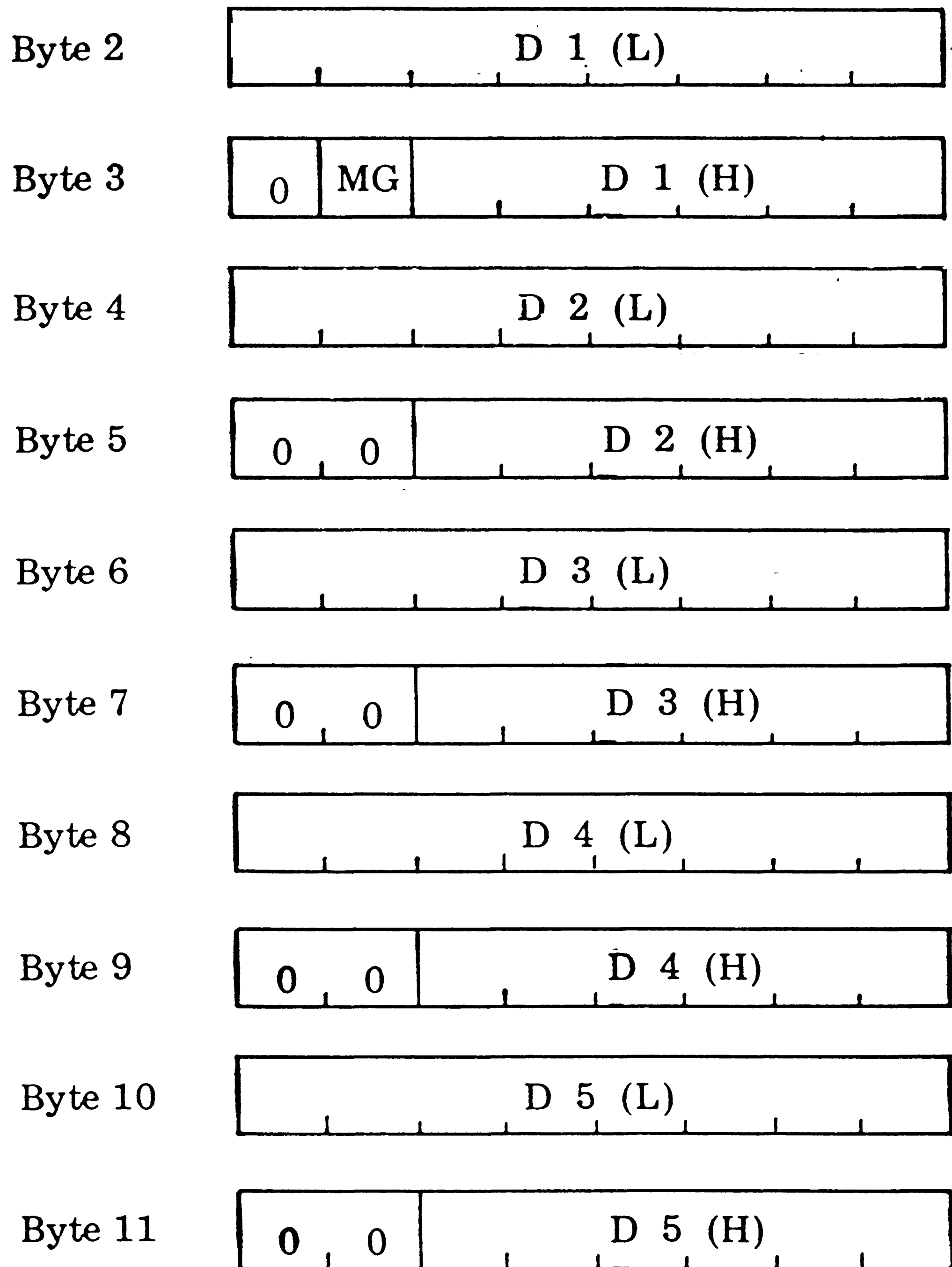
Direction refers to a 3-bit value for the direction of drawing, emanating from the last pixel drawn.



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



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



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.

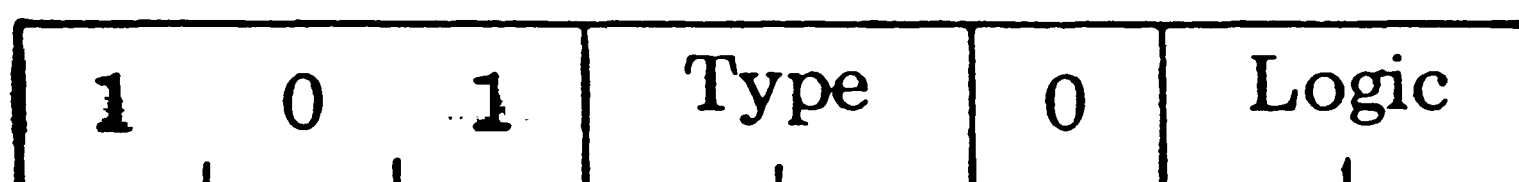
Draw — Command byte: 6CH, no parameters:

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:



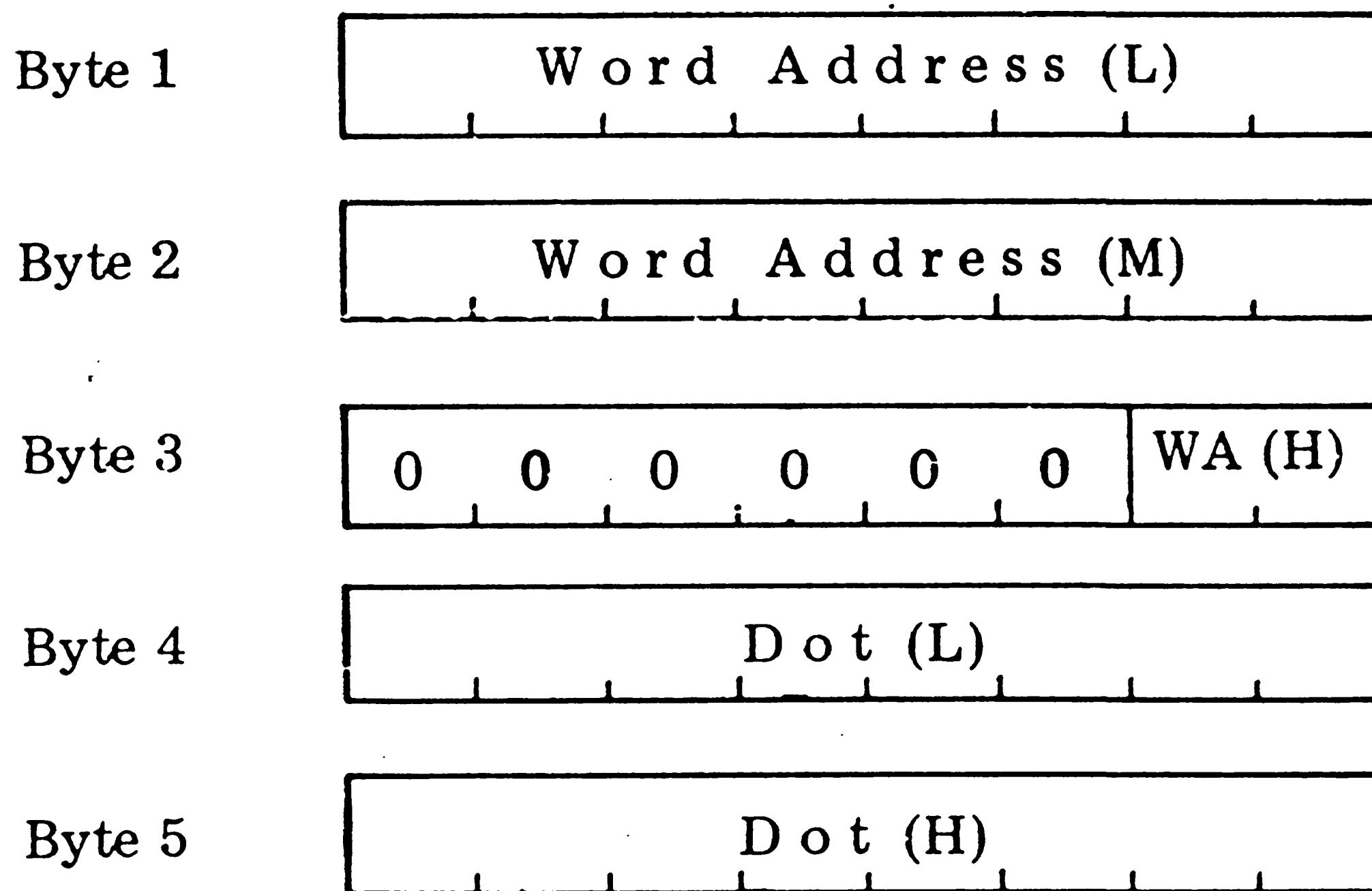
A zero value for Type denotes a word read (low then high). Value 2 indicates low byte of word only, value 3 high byte only. Value 1 is not valid. The Logic value (see Write Data) determines the state in which the graphic display memory will be after reading. Assuming that you wish only to read data and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

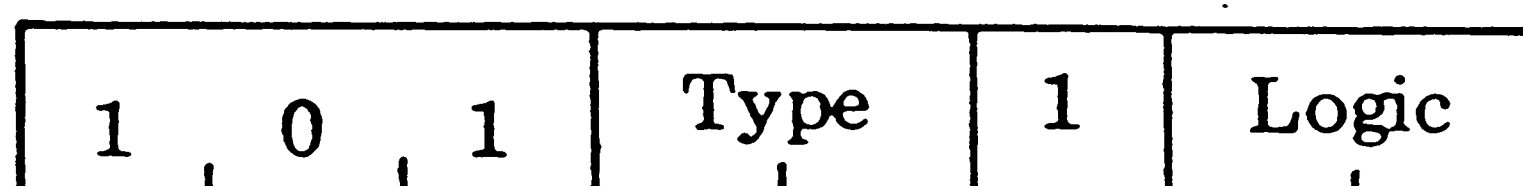
Read Current Address of Cursor – Command byte: 0E0H.

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

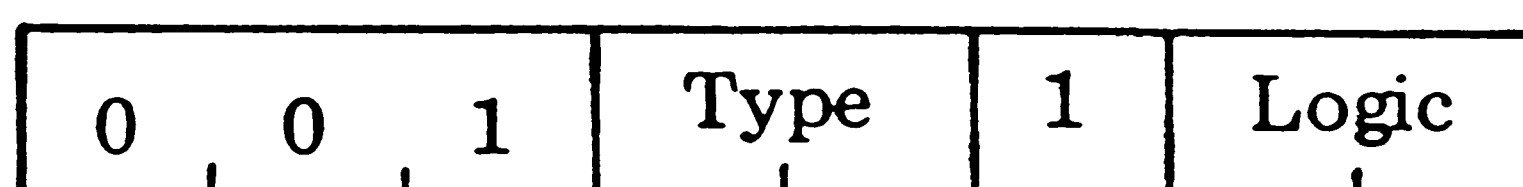


Note that the dot position is not represented by a binary value in 4 bits, but as one set bit among 15 zero bits.

DMA Transfer – Command byte for read request:



Command byte for write request:



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

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

GDC Status Considerations

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

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

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

SOME GDC PROGRAMMING EXAMPLES

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

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

The stage by stage program construction in this section introduces each DB, DW, or DS at the time of discussion of the first routine which makes use of that particular storage definition. Remember to include the ORG 100H directive at the beginning of the program.

```
0100                                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 A. Transmission takes place only when there is no drawing in progress and the FIFO buffer is capable of receiving at least one byte.

```

0102 F5      OUTC:   PUSH PSW
0103 DBA0    OUTC1:  IN DA0H
0105 E60A                    ANI 0AH
0107 C20301                    JNZ OUTC1
010A F1                        POP PSW
010B D3A1                        OUT DA1H
010D C9                        RET

```

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

```

010E DBA0    OUTP:   IN DA0H
0110 E60A                    ANI 0AH
0112 C20E01                    JNZ OUTP
0115 7E      OUTP1:  MOV A,M
0116 D3A0                    OUT DA0H
0118 23                        INX H
0119 0D                        DCR C
011A C21501                    JNZ OUTP1
011D C9                        RET

```

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

```

011E 00      PRAMS:   DB 0
011F 08                        DB 8
0120 0000005900PRAMS1: DB 0,0,0,59H,0,0,0,59H,0FFH,0FFH,0FFH,0FFH,0FFH,
                        0FFH,0FFH,0FFH
0130 000000    PRAMS2: DB 0,0,0
0133 FFFF      PRAMS3: DB 0FFH,0FFH
0135 02FF7F0800PRAMS4: DB 2,0FFH,7FH,8,0,8,0,0FFH,3FH,0FFH,3FH
0140 FFFF      PRAMS5: DB 0FFH,0FFH
0142 21      WRLOGIC: DB 21H      ; complement

```

GINIT is the routine which transmits these parameters:

```

0143 211E01   GINIT:  LXI H,FRAMS
0146 3E0C     MVI A,0CH   ;bit 0 blanks screen
0148 CD0201   CALL OUTC
014B 3E46     MVI A,46H   ;set zoom to zero
014D CD0201   CALL OUTC
0150 0E01     MVI C,1
0152 CD0E01   CALL OUTP
0155 3E48     MVI A,48H   ;cursor/char characs
0157 CD0201   CALL OUTC
015A 0E01     MVI C,1     ;sets lines per row to
                                ;zero for graphics.

015C CD0E01   CALL OUTP
015F 3E70     MVI A,70H   ;load entire parameter
                                ;RAM.

0161 CD0201   CALL OUTC
0164 0E10     MVI C,10H
0166 212001   LXI H,FRAMS1
0169 CD0E01   CALL OUTP   ;sets graphics and 400
                                ;pixels vertical.

016C 3E49     MVI A,49H   ;set cursor pos
016E CD0201   CALL OUTC
0171 0E03     MVI C,3
0173 213001   LXI H,FRAMS2
0176 CD0E01   CALL OUTP   ;first pixel for drawing.
0179 3E4A     MVI A,4AH   ;set mask
017B CD0201   CALL OUTC
017E 0E02     MVI C,2
0180 213301   LXI H,FRAMS3
0183 CD0E01   CALL OUTP
0186 3E4C     MVI A,4CH   ;fig parameters
0188 CD0201   CALL OUTC
018B 0E08     MVI C,08H
018D 213501   LXI H,FRAMS4
0190 CD0E01   CALL OUTP   ;no geom. figs
                                ;direction east.

0193 3E22     MVI A,22H   ;write data word high
                                ;then low, reset to zero.

0195 CD0201   CALL OUTC
0198 0E02     MVI C,2
019A 214001   LXI H,FRAMS5
019D CD0E01   CALL OUTP

```

```

01A0 3E21          MVI A,21H      ;write data, this time
                          ;complement.

01A2 324201       STA WRLOGIC
01A5 CD0201       CALL OUTC
01A8 3E00          MVI A,0DH      ; re-enable screen
01AA CD0201       CALL OUTC
01AD CDB601       WAIT:  CALL GETKEY
01B0 FE24          CPI '$'
01B2 C2ADD1       JNZ WAIT
01B5 C9           RET

```

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

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

```

01B6 C5          GETKEY:  PUSH B
01B7 D5          PUSH D
01B8 0E06        MVI C,6
01BA 1EFF        MVI E,0FFH
01BC C00500      CALL 0005

```



```

01BF D1          POP D
01C0 C1          POP B
01C1 C9          RET

```

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

```

01C2 8F00      EXPRAMS:DB 8FH,0
01C4 0090000100EXPRAMS1: DB 0,90H,0,1,0,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
                                0FFH,0FFH
01D1 21C201    GEXIT:  LXI H,EXPRAMS ;exit routine
01D4 3E4B      MVI A,4BH
01D6 CD0201    CALL OUTC
01D9 0E01      MVI C,1
01DB CD0E01    CALL OUTP
01DE 3E46      MVI A,46H
01E0 CD0201    CALL OUTC
01E3 0E01      MVI C,1
01E5 CD0E01    CALL OUTP
01E8 3E70      MVI A,70H
01EA CD0201    CALL OUTC
01ED 0E00      MVI C,00H
01EF CD0E01    CALL OUTP
01F2 1E1A      MVI E,1AH
01F4 0E02      MVI C,2
01F6 CD0500    CALL 0005
01F9 C9          RET

```

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

As the next stage, we can reserve an area for cursor position (CURPRAMS) and create a routine, CURSET, for transmitting that position to the GDC. CURPRAMS contains in 2 bytes (lower location = less significant byte) the word position, the third byte (highest location) must contain in its four uppermost bits the dot

address within that word (see Position Cursor). The values used here in the DB directives will place the cursor 131,584 pixels from the beginning of display memory (no special significance to this value), that is, approximately halfway along the 206th line of the 400 line display.

```

01FA 20          CURPRAMS: DB 20H
01FB 20          DB 20H
01FC 00          DB 0
01FD 3E49       CURSET: MVI A,49H
01FF CD0201     CALL OUTC
0202 21FA01     LXI H,CURPRAMS
0205 0E03       MVI C,3
0207 CD0E01     CALL OUTP

```

Now reserve an area for storing figure drawing parameters:

```

020A C9          RET
020B          FIGPRAMS: DS 11

```

Enter the routine for transmitting these parameters to the GDC

```

0216 3E4C       FIGSET: MVI A,4CH
0218 CD0201     CALL OUTC
021B 210B02     LXI H,FIGPRAMS
021E 0E08       MVI C,08H
0220 CD0E01     CALL OUTP
0223 C9          RET

```

and the command which sets drawing in progress.

```

0224 3E6C       FIGDRAW: MVI A,6CH
0226 CD0201     CALL OUTC
0229 C9          RET

```

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

```

022A 4003403000FIGPRAM1: DB 40H,3,40H,30H,0,30H,0,0FFH,3FH,30H,0

```

The routines described hitherto can now be used in a program to draw a square. First, the actual parameters in FIGPRAM1 are

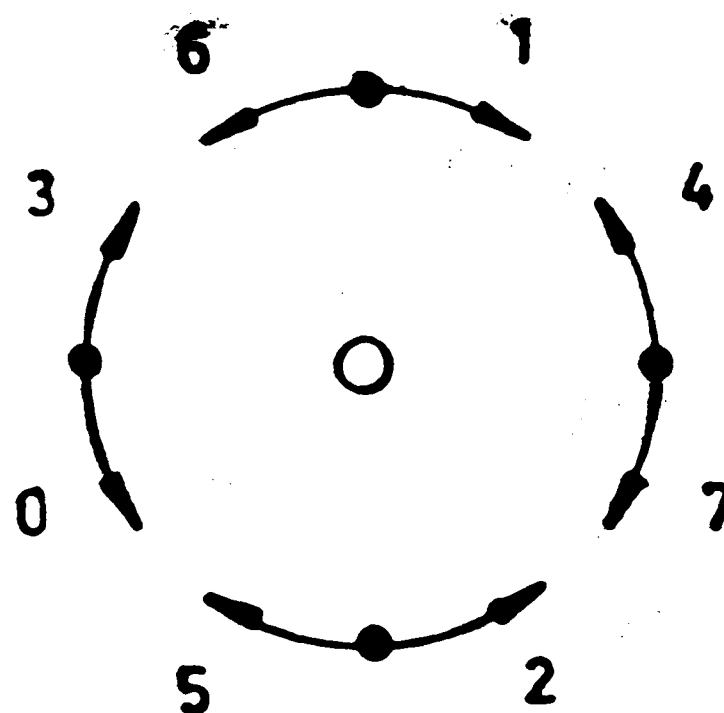
copied to the 11-byte FIGPRAMS area, as this is where the FIGSET routine expects to find them. Then the GDC is set up for graphics. Enter \$, whereupon the cursor is set and the figure drawn. The figure will remain on the screen until you enter x. After the initial run, you may wish to experiment with the values in CURPRAMS and FIGPRAM1.

```

0235 212A02      LXI H,FIGPRAM1
0238 110B02      LXI D,FIGPRAMS
023B 0E0B        MVI C,0BH
023D 7E          NEXTPR1: MOV A,H
023E 12          STAX D
023F 23          INX H
0240 13          INX D
0241 0D          DCR C
0242 023D02      JNZ NEXTPR1
0245 0D4301      CALL GINIT
0248 0DFD01      CALL CURSET
024B 0D1602      CALL FIGSET
024E 0D2402      CALL FIGDRAW
0251 0D8601      WAIT2:  CALL GETKEY
0254 FE78        CPI 'x'
0256 025102      JNZ WAIT2
0259 0DD101      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:

```

025C 40BE      MIDDLE  DW 0BE40H
025E 01        MIDDLEH DB 1
025F 32        RADIUS  DB 50
0260 0000      NORTH   DW 0

```

```

0262 00      NORTHH  DB 0
0263 0000    SOUTH   DW 0
0265 00      SOUTHH  DB 0
0266 0000    EAST    DW 0
0268 00      EASTH   DB 0
0269 0000    WEST    DW 0
026B 00      WESTH   DB 0
026C 0000    PIXEL   DW 0
026E 00      PIXELH  DB 0
026F 00      CURSL   DB 0
0270 00      CURSH   DB 0
0271 00      DUTPOS  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.

```

0272 E5      COMPASS:  PUSH H
0273 D5                      PUSH D
0274 C5                      PUSH B
0275 3A5E02                    LDA MIDDLEH
0278 326202                    STA NORTHH
027B 326502                    STA SOUTHH
027E 326802                    STA EASTH
0281 326B02                    STA WESTH
0284 01B002                    LXI B,280H    ;pitch
0287 215F02                    LXI H,RADIUS
028A 5E                        MOV E,M
028B 2A5C02    CNORTH:      LHLD MIDDLE
028E 53                        MOV D,E
028F 37      NDCR:          STC
0290 3F                        CMC
0291 ED42                    DB 0EDH,42H    ;Z-80 SBC HL,BC
0293 3A6202                    LDA NORTHH
0296 DE00                    SBI 0
0298 326202                    STA NORTHH
029B 15                        DCR D
029C C28F02                    JNZ NDCR

```

029F	226002		SHLD NORTH
02A2	2A5C02	CSOUTH:	LHLD MIDDLE
02A5	53		MOV D,E
02A6	37	SOCR:	STC
02A7	3F		CNC
02A8	ED4A		DB 0EDH,4AH ;Z-80 ADC HL,BC
02AA	3A6502		LDA SOUTHH
02AD	CE00		ACI 0
02AF	326502		STA SOUTHH
02B2	15		DCR D
02B3	C2A602		JNZ SOCR
02B6	226302		SHLD SOUTH
02B9	2A5C02	CEAST:	LHLD MIDDLE
02BC	53		MOV D,E
02BD	010100		LXI B,1
02C0	37	EDCR:	STC
02C1	3F		CNC
02C2	ED4A		DB 0EDH,4AH
02C4	3A6802		LDA EASTH
02C7	CE00		ACI 0
02C9	326802		STA EASTH
02CC	15		DCR D
02CD	C2C002		JNZ EDCR
02D0	226602		SHLD EAST
02D3	2A5C02	CWEST:	LHLD MIDDLE
02D6	53		MOV D,E
02D7	37	WDCR:	STC
02D8	3F		CNC
02D9	ED42		DB 0EDH,42H ;Z-80 ADC HL,BC
02DB	3A6B02		LDA WESTH
02DE	DE00		SBI 0
02E0	326B02		STA WESTH
02E3	15		DCR D
02E4	C2D702		JNZ WDCR
02E7	226902		SHLD WEST
02EA	C1		POP B
02EB	D1		POP D
02EC	E1		POP H
02ED	C9		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.

02EE E5	WORDAD:	PUSH H
02EF D5		PUSH D
02F0 C5		PUSH B
02F1 216C02		LXI H,PIXEL
02F4 7E		MOV A,M
02F5 07		RLC
02F6 07		RLC
02F7 07		RLC
02F8 07		RLC
02F9 47		MOV B,A
02FA E6F0		ANI 0FH
02FC 327102		STA DOTPOS
02FF 78		MOV A,B
0300 E60F		ANI 0FH
0302 326F02		STA CURSL
0305 216C02		LXI H,PIXEL
0308 23		INX H
0309 7E		MOV A,M
030A 07		RLC
030B 07		RLC
030C 07		RLC
030D 07		RLC
030E 47		MOV B,A
030F E6F0		ANI 0FH
0311 216F02		LXI H,CURSL
0314 B6		ORA M
0315 326F02		STA CURSL
0318 78		MOV A,B
0319 E60F		ANI 0FH
031B 327002		STA CURSH
031E 216E02		LXI H,PIXELH
0321 7E		MOV A,M
0322 CB27		DB 0CBH,27H ;Z-80 SLA A
0324 CB27		DB 0CBH,27H
0326 CB27		DB 0CBH,27H
0328 CB27		DB 0CBH,27H
032A 217002		LXI H,CURSH
032D B6		ORA M
032E 77		MOV M,A

```

032F C1          POP B
0330 D1          POP D
0331 E1          POP H
0332 C9          RET

```

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

```

0333 E5          CURTRANSF: PUSH H
0334 2A6F02      LHLD CURSL
0337 22FA01      SHLD CURPRAMS
033A 3A7102      LDA DOTPOS
033D 21FA01      LXI H,CURPRAMS
0340 23          INX H
0341 23          INX H
0342 77          MOV M,A
0343 E1          POP H
0344 C9          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:

```

0345 CD4301      CALL GINIT
0348 CD7202      CALL COMPASS
034B 2A6002      LHLD NORTH
034E 226002      SHLD PIXEL
0351 3A6202      LDA NORTHH
0354 326E02      STA PIXELH
0357 CDDE02      CALL WORDAD
035A CD3303      CALL CURTRANSF
035D CDF001      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.

```

0360 210B02      LXI H,FIGPRAMS
0363 3621        MVI M,21H      ;type of drawing = arc,
                   ;direction = 1.

0365 23         INX H
0366 3623        MVI M,23H      ;rsin 45 for radius
                   ;50 pixels.

0368 23         INX H
0369 3640        MVI M,40H      ;graphics drawing flag
036B 23         INX H
036C 3631        MVI M,31H      ;one less than radius
036E 23         INX H
036F 3600        MVI M,0       ;upper bits zero
0371 23         INX H
0372 3662        MVI M,62H      ;2 * (radius-1)
0374 23         INX H
0375 3600        MVI M,0       ;upper bits zero
0377 23         INX H
0378 36FF        MVI M,0FFH     ;04
037A 23         INX H
037B 363F        MVI M,3FH      ;04 upper bits
037D 23         INX H
037E 3600        MVI M,0       ;05
0380 23         INX H
0381 3600        MVI M,0

;

0383 CD1602      CALL FIGSET
0386 CD2402      CALL FIGDRAW

```

Then follows the Direction 6 arc:

```

0389 CDFD01      CALL CURSET
038C 210B02      LXI H,FIGPRAMS
038F 3626        MVI M,26H
0391 CD1602      CALL FIGSET
0394 CD2402      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.

```

0397 2A6302      _LHD SOUTH
039A 226C02      SHLD PIXEL
039D 3A6502      LDA SOUTHH
03A0 326E02      STA PIXELH
03A3 CDEE02      CALL WORDAD
03A6 CD3303      CALL CURTRANSF
03A9 CDFD01      CALL CURSET
03AC 210B02      LXI H,FIGPRANS
03AF 3622        MVI M,22H
03B1 CD1602      CALL FIGSET
03B4 CD2402      CALL FIGDRAW
03B7 CDFD01      CALL CURSET
03BA 210B02      LXI H,FIGPRANS
03BD 3625        MVI M,25H
03BF CD1602      CALL FIGSET
03C2 CD2402      CALL FIGDRAW

```

;

```

03C5 2A6602      LHLD EAST
03C8 226C02      SHLD PIXEL
03CB 3A6802      LDA EASTH
03CE 326E02      STA PIXELH
03D1 CDEE02      CALL WORDAD
03D4 CD3303      CALL CURTRANSF
03D7 CDFD01      CALL CURSET
03DA 210B02      LXI H,FIGPRANS
03DD 3624        MVI M,24H
03DF CD1602      CALL FIGSET
03E2 CD2402      CALL FIGDRAW
03E5 CDFD01      CALL CURSET
03E8 210B02      LXI H,FIGPRANS
03EB 3627        MVI M,27H
03ED CD1602      CALL FIGSET
03F0 CD2402      CALL FIGDRAW

```

;

```

03F3 2A6902      LHLD WEST
03F6 226C02      SHLD PIXEL
03F9 3A6802      LDA WESTH
03FC 326E02      STA PIXELH

```

```

03FF CDEE02      CALL WORDAD
0402 CD3303      CALL CURTRANSF
0405 CDF001      CALL CURSET
0408 210802      LXI H,FIGPRAMS
040B 3620        MVI M,20H
040D CD1602      CALL FIGSET
0410 CD2402      CALL FIGDRAW
0413 CDF001      CALL CURSET
0416 210802      LXI H,FIGPRAMS
0419 3623        MVI M,23H
041B CD1602      CALL FIGSET
041E CD2402      CALL FIGDRAW

```

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

```

0421 CDB601      WAIT3:  CALL GETKEY
0424 FE78        CPI 'x'
0426 C22104      JNZ WAIT3
0429 CDD101      CALL GEXIT

```

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

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

042C	C0B601	CALCUL:	CALL GETKEY
042F	1600		MVI D,0
0431	FE30		CPI '0'
0433	CA6904		JZ OVER
0436	210B02		LXI H,FIGPRAMS
0439	FE35		CPI '5'
043B	CA7C04		JZ ONOFF
043E	FE31		CPI '1'
0440	CA6A04		JZ DIR7
0443	FE32		CPI '2'
0445	CA7104		JZ DIR0
0448	FE33		CPI '3'
044A	CA7004		JZ DIR1
044D	FE34		CPI '4'
044F	CA6B04		JZ DIR6
0452	FE36		CPI '6'
0454	CA6F04		JZ DIR2
0457	FE37		CPI '7'
0459	CA6C04		JZ DIR5
045C	FE38		CPI '8'
045E	CA6D04		JZ DIR4
0461	FE39		CPI '9'
0463	CA6E04		JZ DIR3
0466	C32C04		JMP CALCUL
0469	C9	OVER:	RET
046A	14	DIR7:	INR D
046B	14	DIR6:	INR D
046C	14	DIR5:	INR D
046D	14	DIR4:	INR D
046E	14	DIR3:	INR D
046F	14	DIR2:	INR D
0470	14	DIR1:	INR D
0471	00	DIR0:	NOP
0472	72		MOV M,D
0473	CD1602		CALL FIGSET
0476	CD2402		CALL FIGDRAW
0479	C32C04		JMP CALCUL
		;	
047C	3A4201	ONOFF:	LDA WRLOGIC
047F	EE01		XRI 1
0481	324201		STA WRLOGIC
0484	CD0201		CALL OUTC
0487	C32C04		JMP CALCUL

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

```
048A 0000400800FIGPRAM2: DB 0,0,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
```

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.

```

0495 218A04      LXI H,FIGPRAM2
0498 110B02      LXI D,FIGPRAMS
049B 0E0B        MVI C,0BH
049D 7E          NEXTPR2: MOV A,M
049E 12          STAX D
049F 23          INX H
04A0 13          INX D
04A1 0D          DCR C
04A2 C29D04      JNZ NEXTPR2
04A5 CD4301      CALL GINIT
04A8 CDFD01      CALL CURSET
04AB 3E23        MVI A,23H
04AD 324201      STA WRLOGIC
04B0 CD0201      CALL OUTC
04B3 CD2C04      CALL CALCUL
04B6 C0B601      WAIT4:  CALL GETKEY
04B9 FE78        CPI 'x'
04BB C2B604      JNZ WAIT4
04BE 3E21        MVI A,21H
04C0 324201      STA WRLOGIC
04C3 CD0201      CALL OUTC      ;resets to complement
                                ;at any rate.
04C6 C3F660      JMP SAVEIT     ;JMP SAVED
04C9 CD0101      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. In addition, you must not attempt to use the BDOS entry facility (CALL 0005) while the ROM is selected, otherwise program control will be lost.

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

```
2100                ORG 2100H
2100      CHSTORE:  DS 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 A 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 HL. The start address of the character area in the ROM is added to this, thus HL addresses the first of the 16 bytes containing the bit pattern. These bytes are then copied via register A to CHSTORE.

```
2110 05      ASCII:  PUSH B
2111 05                PUSH D
2112 E5                PUSH H
2113 6F                MOV L,A
2114 2600             MVI H,0
2116 29                DAD H
2117 29                DAD H
2118 29                DAD H
2119 29                DAD H
211A 110010           LXI D,1000H
211D 19                DAD D      ;ROM char address in HL
211E 110021           LXI D,CHSTORE
2121 0610             MVI B,10H
2123 0311             OUT 11H
2125 7E      ROMBYTE: MOV A,M
2126 12                STAX D
```



```

2127 23          INX H
2128 13          INX D
2129 05          DCR B
212A C22521     JNZ ROMBYTE
212D 0310       OUT 10H
212F E1         POP H
2130 D1         POP D
2131 C1         POP B
2132 C9         RET

```

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

```

2133 3E37       MVI A,'7'
2135 0D1021     CALL ASCII

```

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

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

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

```

2138 04          ZOOMFACT: DB 4
2139 3E46       ZOOM:    MVI A,46H
213B CD0201     CALL OUTC
213E 213821     LXI H,ZOOMFACT
2141 0E01       MVI C,1
2143 CD0E01     CALL OUTP
2146 C9         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.

```

2147 005A427E3CCHARPATT: DB 0,5AH,42H,7EH,3CH,24H,24H,42H
                                ;random example
214F          CHARMIR: DS 8

```

```

;
2157 214721 SKEN: LXI H,CHARPATT
215A 0600 MVI B,0
215C 0E07 MVI C,7
215E 09 DAD B ;HL points to 8th byte
215F 114F21 LXI D,CHARMIR ;DE points to first byte
2162 0E08 MVI C,8
2164 7E NEXTCH: MOV A,M
2165 CD7021 CALL MIRROR ;to cancel mirror,
;replace CALL
;instruction with 3 NOPs
2168 12 STAX D
2169 2B DCX H
216A 13 INX D
216B 0D DCR C ;until all 8 bytes
;inverted and mirrored.
216C C26421 JNZ NEXTCH
216F C9 RET
;
;
2170 C5 MIRROR: PUSH B
2171 D5 PUSH D
2172 E5 PUSH H
2173 57 MOV D,A
2174 21AD21 LXI H,TESTVAR
2177 3E82 MVI A,82H
2179 77 MOV M,A
217A 21B221 LXI H,SETVAR
217D 3EB8 MVI A,0B8H
217F 77 MOV M,A
2180 21B521 LXI H,RESVAR
2183 3E78 MVI A,78H
2185 77 MOV M,A
2186 0E08 MVI C,8
2188 3AAD21 NEXTBIT: LDA TESTVAR ;the Z-80 BIT, SET and
218B D608 SUI 8 ;RESET instructions con-
218D 32AD21 STA TESTVAR ;tained as DBs in the
2190 3AB221 LDA SETVAR ;TESTB routine are modi-
2193 C608 ADI 8 ;fied dynamically eight
2195 32B221 STA SETVAR ;times. Each bit of reg.
2198 3AB521 LDA RESVAR ;D is tested, and its
219B C608 ADI 8 ;mirror counterpart in B
219D 32B521 STA RESVAR ;(e.g. bit 6 for bit 1)
21A0 CDAC21 CALL TESTB ;is set to the value of
21A3 0D DCR C ;the tested bit
21A4 C28821 JNZ NEXTBIT

```

```

21A7 78      MOV A,B
21A8 E1      POP H
21A9 D1      POP D
21AA C1      POP B
21AB C9      RET
21AC CB      TESTB:  DB 0CBH
21AD         TESTVAR: DS 1
21AE CAB421  JZ RESB
21B1 CB      SETB:   DB 0CBH
21B2         SETVAR:  DS 1
21B3 C9      RET
21B4 CB      RESB:   DB 0CBH
21B5         RESVAR:  DS 1
21B6 C9      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

```

21B7 1607400700CHFGRAM: DB 16H,7,40H,7,0
                                     ;set slnt with bit 7
                                     ;in byte 1.

```

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

```

21B8 3E78    CHAROUT: MVI A,78H    ;starter in param at
                                     ;param 8.
21BE C00201  CALL OUTC
21C1 214F21  LXI H,CHARMIR
21C4 0E08    MVI C,8
21C6 C00E01  CALL OUTP
21C9 3E4C    MVI A,4CH    ;figset
21CB C00201  CALL OUTC
21CE 21B721  LXI H,CHFGRAM
21D1 0E05    MVI C,5
21D3 C00E01  CALL OUTP

```

```

21D6 3E04          MVI A,4
21D8 323821       STA ZOOMFACT
21DB CD3921       CALL ZOOM
21DE 3E68          MVI A,68H      ;graphic character draw
21E0 CD0201       CALL OUTC
21E3 C9           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.

```

21E4 CD4301       CALL GINIT
21E7 CDFD01       CALL CURSET
21EA 3E37          MVI A,'7'
21EC CD1021       CALL ASCII
21EF 210021       LXI H,CHSTORE
21F2 23           INX H           ;does not copy
21F3 23           INX H           ;entire character
21F4 23           INX H
21F5 114721       LXI D,CHARPATT
21F8 0E08          MVI C,8
21FA 7E           NEXTCOP: MOV A,M
21FB 12           STAX D
21FC 23           INX H
21FD 13           INX D
21FE 0D           DCR C

```

```

21FF C2FA21          JNZ NEXTCOP
2202 CD5721          CALL SKEW
2205 CDBC21    REPEAT:  CALL CHAROUT
2208 CDB601    WAIT5:   CALL GETKEY
220B FE72          CPI 'r'
220D CA0522          JZ REPEAT
2210 FE78          CPI 'x'
2212 C20822          JNZ WAIT5
2215 CDD101          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 even possible in the 64KB memory. However, if the operating system and the debugging utility are to be retained in memory, this leaves little memory space for other programs.

The data areas required:

```

2218 FFFF    PRMSR:  DB 0FFH,0FFH
221A FFFF    RMASK:  DB 0FFH,0FFH
221C 0208400800FIGSR:  DB 2,8,40H,8,0,8,0,0FFH,3FH,0FFH,3FH
2227 02      MASKFIG:  DB 2
2228        SCREEN:  DS 16000
60A8 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 HL register, and that register is then incremented.

```

60AA C5      READSCRN: PUSH B
60AB DE08           MVI C,8
60AD 1602      NEXTWORD: MVI D,2
60AF DBA0      READYCHK: IN 0A0H
60B1 E601           ANI 1
60B3 CAAF60           JZ READYCHK
60B6 DBA1           IN 0A1H
60B8 77           MOV M,A
60B9 23           INX H
60BA 15           DCR D
60BB C2AF60           JNZ READYCHK
60BE 0D           DCR C
60BF C2AD60           JNZ NEXTWORD
60C2 C1           POP B
60C3 C9           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.

```

60C4 3EAD      FIFOCLR: MVI A,0A0H
60C6 C00201           CALL OUTC
60C9 C9           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.

```

60CA C5      SETREAD:  PUSH B
60CB E5                      PUSH H
60CC 3E78      MVI A,78H
60CE CD0201    CALL OUTC      ;set pRAM
60D1 211822    LXI H,PRAMSR
60D4 0E02      MVI C,2
60D6 CD0E01    CALL OUTP
60D9 3E4A      MVI A,4AH
60DB CD0201    CALL OUTC      ;set mask
60DE 211A22    LXI H,RMASK
60E1 0E02      MVI C,2
60E3 CD0E01    CALL OUTP
60E6 3E4C      MVI A,4CH
60E8 CD0201    CALL OUTC      ;set fig
60EB 211C22    LXI H,FIGSR
60EE 0E0B      MVI C,0BH
60F0 CD0E01    CALL OUTP
60F3 E1        POP H
60F4 C1        POP B
60F5 C9        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.

```

60F6 21FA01   SAVEIT:  LXI H,CURPRAMS
60F9 3600           MVI M,0           ;reading to start
60FB 23           INX H             ;at top left
60FC 3600           MVI M,0           ;corner of the
60FE 23           INX H             ;screen
60FF 3600           MVI M,0
6101 CDF001           CALL CURSET
6104 212822           LXI H,SCREEN
6107 01E803           LXI B,03E8H
610A CDCA60   NEXTSCRN: CALL SETREAD
610D CDC460           CALL FIFUCLR
6110 CDAA60           CALL READSCRN
6113 0B           DCX B
6114 3E00           MVI A,0
6116 B8           CMP B
6117 C20A61           JNZ NEXTSCRN
611A B9           CMP C
611B C20A61           JNZ NEXTSCRN
611E CDD101           CALL GEXIT

```

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

```

6121 CD9261           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.

```

6124 218A04   PAINT:  LXI H,FIGPRAM2
6127 110B02           LXI D,FIGPRAMS
612A 0E0B           MVI C,0BH
612C 7E           NEXTPR3: MOV A,M
612D 12           STAX D

```

```

612E 23      INX H
612F 13      INX D
6130 0D      DCR C
6131 C22C61  JNZ NEXTPR3
6134 3E02      MVI A,2      ;Direction East
6136 320802  STA FIGPRAMS
6139 CD4301  CALL GINIT
613C C0F001  CALL CURSET
613F CD1602  CALL FIGSET
6142 112822  LXI D,SCREEN
6145 01803E  LXI B,3E80H
6148 05      NEWBYTE: PUSH B
6149 0E08      MVI C,8
614B 1A      LDAX D
614C 47      MOV B,A
614D CB38      CHECKBIT: DB 0CBH,38H      ;Z-80 SRL B.
614F DA5761  JC PLOT      ;check Carry,
6152 3E22      MVI A,22H    ;set drawing
6154 C35961  JMP LOGICSET ;Logic
6157 3E23      PLOT: MVI A,23H ;accordingly
6159 C00201  LOGICSET: CALL OUTC
615C 3E4C      MVI A,4CH    ;command to set
615E C00201  CALL OUTC    ;Figure parameters
6161 210802  LXI H,FIGPRAMS
6164 05      PUSH B
6165 0E03      MVI C,3      ;set first three
6167 C00E01  CALL OUTP    ;Figure parameters
616A 01      POP B
616B 3E6C      MVI A,6CH    ;Draw command
616D C00201  CALL OUTC
6170 0D      DCR C
6171 C24D61  JNZ CHECKBIT
6174 13      INX D
6175 01      POP B
6176 0B      DCX B
6177 3E00      MVI A,0
6179 B3      CMP B
617A C24861  JNZ NEWBYTE
617D B9      CMP C
617E C24861  JNZ NEWBYTE
6181 3E21      MVI A,21H    ;restore
6183 C00201  CALL OUTC    ;Complement Logic
6186 CDB601  WAIT6: CALL GETKEY
6189 FE78      CPI 'x'

```

```

618B C28661      JNZ WAIT6
618E C0D101      CALL GEXIT
6191 00          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.

```

6192 C0B601      ADJUST:  CALL GETKEY
619C CAA561      JZ ADJUST1      ;pixel inversion
619F FE6D        CPI 'a'
61A1 CABF61      JZ ADJUST2      ;mirror image
619C CAA561      JZ ADJUST1
619F FE6D        CPI 'a'
61A1 CABF61      JZ ADJUST2
61A4 C9          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 8080 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.

```

61A5 112822      ADJUST1: LXI D,SCREEN
61A8 01803E      LXI B,3E80H
61AB 1A          ADJUST11: LDAX D
61AC 2F          CMA
61AD 12          STAX D
61AE 13          INX D
61AF 0B          DCX B
61B0 3E00        MVI A,0
61B2 BE          CMP B
61B3 C2AB61      JNZ ADJUST11
61B6 B9          CMP C
61B7 C2AB61      JNZ ADJUST11
61BA C9          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." In addition, the same is done within each byte, using the MIRROR routine described earlier. Thus, an arrow which previously pointed left, will now point to

the right when the contents of SCREEN are re-written to graphics display memory.

```

61BB 28      LEFTBYTE: DB 40
61BC 29      RGHTBYTE: DB 41
61BD 2722    LINE:     DW SCREEN -1
;
61BF 01C800  ADJUST2: LXI B,200
61C2 212722          LXI H,SCREEN -1
61C5 22BD61          SHLD LINE
61C8 C5        NEXTLINE: PUSH B
61C9 3E28          MVI A,40
61CB 32BB61      STA LEFTBYTE ;starting from center of
61CE 3C          INR A          ;line working outwards;
61CF 32BC61      STA RGHTBYTE ;exchange byte pairs
61D2 012800      LXI B,40      ;within that line
61D5 C5        LINESWOP: PUSH B
61D6 3ABB61      LDA LEFTBYTE
61D9 5F          MOV E,A
61DA 1600        MVI D,0
61DC 2AB061      LHLD LINE
61DF 19          DAD D
61E0 7E          MOV A,M
61E1 CD7021     CALL MIRROR ;mirror byte within itself.
61E4 47          MOV B,A      ;left byte of pair in B
61E5 3ABC61      LDA RGHTBYTE
61E8 5F          MOV E,A
61E9 3C          INR A
61EA 32BC61      STA RGHTBYTE
61ED 1600        MVI D,0
61EF 2AB061      LHLD LINE
61F2 19          DAD D
61F3 7E          MOV A,M
61F4 CD7021     CALL MIRROR ;mirror byte within itself.
61F7 4F          MOV C,A      ;right byte of pair in C.
61F8 70          MOV M,B      ;new right byte taken from B
61F9 2AB061      LHLD LINE
61FC 3ABB61      LDA LEFTBYTE
61FF 5F          MOV E,A
6200 3D          DCR A
6201 32BB61      STA LEFTBYTE
6204 1600        MVI D,0
6206 19          DAD D
6207 71          MOV M,C      ;new left byte taken from C

```

6208 01	POP B
6209 3E00	MVI A,0
620B 0B	DCX B
620C 88	CMF B
620D C2D561	JNZ LINESWOP
6210 B9	CMF C
6211 C2D561	JNZ LINESWOP
6214 2AB061	LHLD LINE
6217 115000	LXI D,80
621A 19	DAD D ;point to next
621B 22B061	SHLD LINE ;line of screen
621E 01	POP B
621F 0B	DCX B
6220 88	CMF B
6221 C2C861	JNZ NEXTLINE
6224 B9	CMF C
6225 C2C861	JNZ NEXTLINE
6228 09	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 re-stored 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:

6229 0D9261	CALL ADJUST
622C 0D4301	CALL GINIT
622F 3E4C	MVI A,4CH ;command to set
6231 0D0201	CALL OUTC ;Figure parameters
6234 212722	LXI H,MASKFIG


```

6237 0E01          MVI C,1          ;only one required
6239 CD0E01       CALL OUTP
623C CDF001       CALL CURSET
623F 212822       LXI H,SCREEN
6242 11A860       LXI D,DUMBYTES
6245 01401F       LXI B,1F40H
6248 C5           NEXTMASK: PUSH B
6249 3E4A         MVI A,4AH        ;command to
624B CD0201       CALL OUTC        ;load Mask
624E 0E02         MVI C,2          ;with word of
6250 CD0E01       CALL OUTP        ;screen contents.
6253 3E23         MVI A,23H        ;write to screen
6255 CD0201       CALL OUTC        ;with "set" Logic...
6258 EB           XCHG          ;(HL now points to
                       ; DUMBYTES)
6259 0E02         MVI C,2          ;... setting all bits
625B CD0E01       CALL OUTP        ;that are set in Mask
625E 2B           DCX H
625F 2B           DCX H
6260 EB           XCHG          ;HL now points to
                       ;next screen word.
                       ;DE to DUMBYTES

6261 C1           POP B
6262 0B           DCX B
6263 3E00         MVI A,0
6265 BB           CMP B
6266 C24862       JNZ NEXTMASK
6269 B9           CMP C
626A C24862       JNZ NEXTMASK
626D CDB601       WAIT7: CALL GETKEY
6270 FE78         CPI 'x'
6272 C26062       JNZ WAIT7
6275 CD0101       CALL GEXIT
6278 00           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 CUR-PRAMS). This program loads the Mask register word by word with the contents of SCREEN. The Write Data command is transmitted to the GDC with all its parameter bits set. This means that the 16-bit pattern contained in the Mask register appears as a horizontal pattern of data on the screen in one write cycle. There is no need to repeat the Figure parameter setting. By altering the

initial cursor position, you can address different parts of the screen.

COLOR GRAPHICS

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

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

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 B

CP/M-80 BIOS PROGRAM

```

003 (MBIOS)

66
67 ;*****
68 ;*
69 ;* module name = ncrbios.asm
70 ;*
71 ;*****
72 DE00 = STBIOS EQU 0DE00H ;START OF BIOS
73 DE0C = NCONOUT EQU STBIOS+12 ;OUTPUT A CHARACTER TO CRT
74 PAGE

004 (MBIOS)

75
76 VERSEQU
77 + ;*****
78 + ;*
79 + ;* SWITCHES 07.06.83 10:00 PM
80 + ;*
81 + ;*****
82 +
83 0000+= CPMRAM EQU 0 ; CP/M RAM BASE
84 0002+= CPMVER EQU 2 ; CP/M VERSION
85 0000+= RELEASEH EQU 0 ; MCR RELEASE HIGH BYTE
86 0004+= RELEASE EQU 4 ; MCR RELEASE LOW BYTE
87 0000+= MCRVERH EQU 2 ; MCR VERSION HIGH BYTE
88 0004+= MCRVER EQU 4 ; MCR VERSION LOW BYTE
89 0001+= DMRR EQU 1 ; D-NUMBER RELEASE
90 0000+= DMBS EQU 0 ; D-NUMBER SUBISSUE
91 0FF7+= FWVER EQU 0FF7H ; LOCATION OF FWVERSION IN FIRMWARE
92 + PAGE

```

005 :MBIOS)

```
93      +
94      +          ENDM
95      STADDEQU
96      + ;*****
97      + ;#
98      + ;#   START ADDRESS EQUATES   27/10/82 10:21 WF
99      + ;#
100     + ;*****
101     +
102     +
103     DE00+=      BIOS   EQU    00E00H      ; START ADDRESS OF BIOS
104     +
105     0E00+=      BD0SSZ EQU    0E00H      ;
106     +
107     +
108     0800+=      CCPSZ  EQU    0800H      ; CCP SIZE
109     +
110     D000+=      BD0S   EQU    BIOS-BD0SSZ ; START OF BD0S
111     C800+=      CCP    EQU    BD0S-CCPSZ  ; START OF CCP
112     +
113     +
114     D006+=      BD0SNTRY EQU    BD0S+6
115     D047+=      BD0STBL EQU    BD0S+47H  ; BD0S FUNCTION TABLE START ADDRESS
116     C803+=      CCPSTB EQU    CCP+3
117     +
118     +
119     +          PAGE
```

006 (MBIOS)

```

120      +
121      +          ENDM
122      +          DISKCEQU
123      +          ;*****
124      +          ;*
125      +          ;*          DISK CONTROLLER EQUATES 30.11.82 11:56 PH
126      +          ;*
127      +          ;*****
128      +
129      +
130      +          DCMD  EQU    00051H      ; DISK COMMAND PORT
131      +          DSTAT EQU    00050H      ; DISK STATUS PORT
132      +          FDCRA  EQU    00051H      ; READ DATA FROM FDC
133      +          ;
134      +          MOTOROFF EQU    01H      ; MOTOR OFF INDICATOR (SYSSTA)
135      +          ;
136      +          ;
137      +          RAMSEL EQU    00010H      ; RAM SELECT
138      +          ROMSEL EQU    00011H      ; ROM SELECT
139      +          SETTC  EQU    00012H      ; SET TC
140      +          SYSSTA EQU    00013H      ; SYSTEM STATUS PORT
141      +          MOTORON EQU    00014H      ; MOTOR ON
142      +          COAD   EQU    00026H      ; FDC DMA CHANNEL
143      +          COTC   EQU    00027H      ; FDC DMA CHANNEL
144      +          DMACOM EQU    00028H      ; DMA COMMAND
145      +          DMAST  EQU    00028H      ; READ DMA STATUS
146      +          DMANO  EQU    00028H
147      +          DMAMB  EQU    0002AH
148      +
149      +          DMAWRT EQU    004H      ; DMA WRITE COMMAND
150      +          DMAREAD EQU    008H      ; DMA READ COMMAND
151      +
152      +          ENFD   EQU    003H      ; ENABLE FDC CHANNEL (CH3)
153      +          DISFD  EQU    007H      ; DISABLE FDC CHANNEL (CH3)
154      +
155      +
156      +          PAGE
    
```

007 (MBIOS)

```
157 +
158 +
159 +
160 +
161 +
162 +
163 +
164 +
165 +
166 +
167 0003+= IOBYTE EQU 3 ; I/O BYTE
168 0004+= CURDISK EQU 4 ; CURRENT DISK NUMBER
169 000D+= CR EQU 0DH ; CARRIAGE RETURN
170 000A+= LF EQU 0AH ; LINE FEED
171 0008+= BS EQU 08H ; BACK SPACE
172 000C+= FF EQU 0CH ; FORM FEED
173 000C+= TOF EQU 0CH ; TOP OF FORM
174 001B+= ESC EQU 1BH ; ESCAPE
175 001A+= CLRSCRN EQU 01AH ; CLEAR SCREEN
176 +
PAGE
```

008 (MBIOS)

```
177 +
178 +
179 +
PAGE
```


009 (MBIOS)

```

180
181 ;
182 ;
183 0000 032100 BEGIN: JMP BMOVE ; JUMP TO MOVE ROUTINE
184 0003 4350402032 DB 'CPM 2.2' ; DISK IDENTIFICATION
185 000A 4E43522046 DB 'NCR F3' ; DISK FORMAT 0005
186 0010 323034 DB CPMVER+'0'.RELEASE+'0'.RELEASE+'0'
187 0013 3034 DB NCRVER+'0',NCRVER+'0' ; INTERNAL RELEASE NUMBER
188 0015 3030303030SNR: DB '0','0','0','0','0' ; SERIAL NUMBER
189 001A 286329204E DB '(c) NCR' ; COPYRIGHT
190 ;
191 0021 310070 BMOVE: LXI SP,7000H ; TEMPORARILY STACK
192 0024 21000E LXI H,BIOS ; BIOS START ADDRESS
193 0027 110022 LXI D,2200H ; START OF BIOS IN MEMORY
194 002A 010018 LXI B,1800H ; LENGTH OF BIOS
195 002D 000600 CALL MOVER ; MOVE BIOS TO THE RIGHT ADDRESS
196 ;
197 ;
198 ; DISPLAY FIRMWARE VERSION
199 ;
200 ;
201 0030 21F70F LXI H,FWVER
202 0033 7E MOV A,M
203 0034 FE08 CPI 08H
204 0036 024A00 JNZ RANTEST ; JUMP IF OLD FIRMWARE
205 0039 3E46 FW1: MVI A,'F'
206 003B 324501 STA FWMESS1 ; ACTIVATE FWVERSION DISPLAY
207 003E 11F80F LXI D,FWVER+1
208 0041 215701 LXI H,FWMESS2
209 0044 010800 LXI B,B
210 0047 000600 CALL MOVER ; MOVE FIRMWARE VERSION
211
212
213
214 PAGE

```

```

010 (MBIOS)

215
216 ;
217 ;
218 ; RANTEST
219 ;
220 ;
221 RANTEST:
222 004A D310 OUT RAMSEL ; SWITCH TO LOW RAM (0-FFFH)
223 RTEST1:
224 004C 210000 LXI H,C
225 RTEST2:
226 004F 3655 MVI M,055H
227 0051 23 INX H
228 0052 36AA MVI M,0AAH
229 0054 23 INX H
230 0055 7C MOV A,H
231 0056 FE20 CPI 20H
232 0058 C24F00 JNZ RTEST2 ; FILL MEMORY WITH 55AA55AA
233 005B 2B RTEST3: DCX H
234 005C 7C MOV A,H
235 005D FEFF CPI OFFH
236 005F CA7200 JZ RTEST4 ; JUMP IF TEST OK
237 0062 7E MOV A,M
238 0063 FEAA CPI 0AAH
239 0065 C29300 JNZ RERROR ; RAM - TEST ERROR
240 0068 2B DCX H
241 0069 7E MOV A,M
242 006A FE55 CPI 055H
243 006C CA5B00 JZ RTEST3 ; GET NEXT BYTE
244 006F C39300 JMP RERROR ; RAM - TEST ERROR
245 0072 23 RTEST4: INX H
246 0073 36AA RTEST5: MVI M,0AAH
247 0075 23 INX H
248 0076 3655 MVI M,055H
249 0078 23 INX H
250 0079 7C MOV A,H
251 007A FE20 CPI 20H
252 007C C27300 JNZ RTEST5 ; FILL MEMORY WITH AA55AA55
253 007F 2B RTEST6: DCX H
254 0080 7C MOV A,H
255 0081 FEFF CPI OFFH
256 0083 CA6201 JZ SIGNON ; JUMP TO COLD BOOT ROUTINE
257 0086 7E MOV A,M
258 0087 FE55 CPI 055H
259 0089 C29300 JNZ RERROR ; RAM - TEST ERROR
260 008C 2B DCX H
261 008D 7E MOV A,M
262 008E FEAA CPI 0AAH
263 0090 CA7F00 JZ RTEST6 ; GET NEXT BYTE
264 0093 219A00 RERROR: LXI H,RAMERROR
265 0096 C07701 CALL DISPERR ; DISPLAY ERROR MESSAGE
266 0099 76 HLT ; NO MORE RESPONSE
267 ;
268 ;
269 RAMERROR:
270 009A 1A DB CLRSCN

```

011 (MBIOS)

```

271 0098 18303820    DB    ESC, '=', 24+20H, 0+20H    ;SET CURSOR TO ERROR LINE
272 009F 4552524F52  DB    'ERROR ENCOUNTERED AT RAM TEST (0-1FFF)'
273 00C5 24          DB    '$'
274
275
276
277                                PAGE

```

012 (MBIOS)

```

278
279 ;
280 ;
281 ;    MOVE BC-BYTES FROM A(DE) -> A(HL)
282 ;
283 ;
284 MOVER:
285 00C6 1A          LDAX  D
286 00C7 77          MOV   M,A
287 00C8 23          INX  H
288 00C9 13          INX  D
289 00CA 08          DCX  B
290 00CB 78          MOV  A,B
291 00CC B1          ORA  C
292 00CD C8          RZ           ; RETURN IF ALL MOVED
293 00CE C3C600     JMP   MOVER
294                                PAGE

```

013 (MBIOS)

```

295
296 ;*****
297 ;*                                     *
298 ;* DISPLAY SIGNON MESSAGE           *
299 ;*                                     *
300 ;*****
301 ;
302 ;
303 ;
304 ;
305 ;
306 SIGMSG:
307 0001 1A          DB      CLKSCN
308 0002 000A       DB      CR,LF
309 0004 43502F4D20 DB      'CP/M (R) 2.2 for MCR DECISION MATE V'
310 00F8 000A       DB      CR,LF
311 00FA 4430303620 DB      'D006-0051-',DNBRR+'0',DNBRS+'0','00'
312 0108 000A       DB      CR,LF
313 010A 436F707972 DB      'Copyright (c) 1982, DIGITAL RESEARCH'
314 012E 000A       DB      CR,LF
315 0130 5365726961 DB      'Serial Number '
316 013E          SIGMSG1: DB      5          ; SERIAL NUMBER
317 0143 000A       DB      CR,LF
318 0145 2469726D77FWMESS1:DB      '$irmware Version: '
319 0157 2020202020FWMESS2: DB      '
320 015F 000A24     DB      CR,LF,'$'
321
322 SIGNON:
323 0162 111500     LXI      D,SHR
324 0165 213E01     LXI      H,SIGMSG1
325 0168 010500     LXI      B,5
326 0168 CDC600     CALL     MOVER          ; MOVE SERIAL NUMBER
327 016E 21D100     LXI      H,SIGMSG
328 0171 CD7701     CALL     DISPERR       ; DISPLAY MESSAGE
329 0174 C3000E     JMP      STBIOS        ; JUMP TO COLD BOOT ROUTINE
330
331 PAGE

```

014 (MBIOS)

```

332
333 DISPERR:
334 0177 3E01       MVI      A,01
335 0179 320300     STA      IOBYTE        ; SET TEMPORARY I/O BYTE FOR CRT
336 DISP1:
337 017C 7E         MOV      A,M
338 017D FE24       CPI      '$'
339 017F C8         RZ
340 0180 4F         MOV      C,A
341 0181 E5         PUSH     H
342 0182 C0C0DE     CALL     MCONOUT       ;DISPLAY CHARACTER
343 0185 E1         POP      H
344 0186 23         INX     H
345 0187 C37C01     JMP      DISP1        ;NEXT CHARACTER
346 018A          END

```

		015	(MBIOS)						
BDOS	D000	110	111	114	115				
BDOSMTRY	D006	114							
BDOSSZ	DE00	105	110						
BDOSTBL	D047	115							
BEGIN	D000	183							
BIOS	DE00	103	110	192					
BMOVE	0021	183	191						
BS	0008	171							
CCP	C800	111	116						
CCPSTB	C803	116							
CCPSZ	0800	108	111						
CLRSCH	001A	175	270	307					
COAD	0026	142							
COTC	0027	143							
CPM RAM	0000	83							
CPMVER	0002	84	186						
CR	0000	169	308	310	312	314	317	320	
CURDISK	0004	168							
DCOMD	0051	130							
DISFD	0007	153							
DISP1	017C	336	345						
DISPERR	0177	265	328	333					
DMACOM	0028	144							
DMAMB	002A	147							
DMAMO	0028	146							
DMAREAD	0008	150							
DMAST	0028	145							
DMAWRT	0004	149							
DWBRR	0001	89	311						
DWBRS	0000	90	311						
DSTAT	0050	131							
EMFD	0003	152							
ESC	0018	174	271						
FDCRA	0051	132							
FF	000C	172							
FW1	0039	205							
FWMESS1	0145	206	318						
FWMESS2	0157	208	319						
FWVER	0FF7	91	201	207					
IOBYTE	0003	167	335						
LF	000A	170	308	310	312	314	317	320	
MOTOROFF	0001	134							
MOTORON	0014	141							
MOVER	00C6	195	210	284	293	326			
WCONOUT	DE0C	73	342						
WCRVER	0004	88	187						
WCRVERH	0000	87	187						
RAMERROR	009A	264	269						
RAMSEL	0010	137	222						
RAMTEST	004A	204	221						
RELEASE	0004	86	186						
RELEASEH	0000	85	186						
RERROR	0093	239	244	259	264				
ROMSEL	0011	138							
RTEST1	004C	223							
RTEST2	004F	225	232						

		016	(M BIOS)
RTEST3	0058	233 4	243
RTEST4	0072	236	245 4
RTEST5	0073	246 4	252
RTEST6	007F	253 4	263
SETTC	0012	139 4	
SIGMSG	0001	306 4	327
SIGMSG1	013E	316 4	324
SIGNOM	0162	256	322 4
SMR	0015	188 4	323
STBIOS	DE00	72 4	73 329
SYSSTA	0013	140 4	
TOF	000C	173 4	
L			

003 (NBIOS)

```

66
67 ;*****
68 ;*****
69 ;*****
70 ;*****      EXTERNAL DEFINITIONS      *****
71 ;*****
72 ;*****
73 ;*****
74 ;
75 ;
76 ;
77 EXTRN  CONST,CONIN,LIST,PUNCH,READER,LISTST,WRCUPO
78 EXTRN  RESET,CINIT,WINIT,DSKSET,READSEC1,DSKERR,CHECKOS
79 EXTRN  TRKSET,SECSET,DMASET,BTRECS,CREAD,PMAADR,RDTRAK
80 EXTRN  CURCOL,CLRLI1,CONOUT,FORMAT,CWRITE,DHOME
81 EXTRN  SEKOSK,SEKTRK,SEKSEC,DMAADR,READOP,URTYPE,SEKHST
82 EXTRN  UNACNT,UNADSK,UNATRK,UNASEC,READID,DSKCMD
83 EXTRN  RSFLAG,HSTACT,HSTDSK,HSTTRK,HSTSEC,HSTWRT
84 EXTRN  DISKID,ERFLAG,DTRKSET,SAVESEC
85 EXTRN  FREAD,FWRITE
86
87
88 PAGE
    
```

004 (NBIOS)

```

89
90 ;*****
91 ;*****
92 ;*****
93 ;*****      ENTRY DEFINITIONS      *****
94 ;*****
95 ;*****
96 ;*****
97 ;
98 ;
99 ;
100 PUBLIC CRTTBL,OFST,CONFIGFL,KEYTBL,FUNTBL,MOVE,FNTBLMD
101 PUBLIC COMPDEHL,MOVEL,FMERRM,DISPERR,MULT,SUBHLDE
102 PUBLIC RSTC,RETRYC,IOBSET,STBIOS,CHPRE,DSKTYP,MRDYRX
103 PUBLIC CLOSE,NOTRDYM,WRPRCTM,FATERRM,IOERRM,DISPERR
104 PUBLIC WNOTE,M1RS232,M2RS232,PVRS232,DSKGET,FIXINIT
105 PUBLIC DSKMSC,DSKSLC,DSKSID,DSKTRK,FLUSH,CLRLIM,NBRFLEX
106
107 PAGE
    
```

005 (NBIOS)

```

108
109 VERSEQU
110 0000+= CPMRAM EQU 0 ; CP/M RAM BASE
111 0002+= CPMVER EQU 2 ; CP/M VERSION
112 0000+= RELEASEH EQU 0 ; NCR RELEASE HIGH BYTE
113 0004+= RELEASE EQU 4 ; NCR RELEASE LOW BYTE
114 0000+= NCRVERH EQU 0 ; NCR VERSION HIGH BYTE
115 0004+= NCRVER EQU 4 ; NCR VERSION LOW BYTE
116 0001+= DNBR EQU 1 ; D-NUMBER RELEASE
117 0000+= DNBR EQU 0 ; D-NUMBER SUBISSUE
118 0FF7+= FWVER EQU 00FF7H ; LOCATION OF FWVERSION IN FIRMWARE

```

006 (NBIOS)

```

119 STADSEQU
120 0E00+= BIOS EQU 0E00H ; START ADDRESS OF BIOS
121 0E00+= BDOSZ EQU 0E00H ;
122 0800+= CCPSZ EQU 0800H ; CCP SIZE
123 0000+= BDOS EQU BIOS-BDOSZ ; START OF BDOS
124 C800+= CCP EQU BDOS-CCPSZ ; START OF CCP
125 0006+= BDOSNTRY EQU BDOS+6
126 0047+= BOOSTBL EQU BDOS+47H ; BDOS FUNCTION TABLE START ADDRESS
127 C803+= CCPSTB EQU CCP+3

```

007 (NBIOS)

128		DISKCEQU			
129	0051+=	DCOMD EQU	00051H		; DISK COMMAND PORT
130	0050+=	DSTAT EQU	00050H		; DISK STATUS PORT
131	0051+=	FDCRA EQU	00051H		; READ DATA FROM FDC
132	0001+=	MOTOROFF	EQU	01H	; MOTOR OFF INDICATOR (SYSSTA)
133	0010+=	RAMSEL EQU	(; RAM SELECT
134	0011+=	ROMSEL EQU	(; ROM SELECT
135	0012+=	SETTC EQU	(; SET TC
136	0013+=	SYSSTA EQU	(; SYSTEM STATUS PORT
137	0014+=	MOTORON EQU	(; MOTOR ON
138	0026+=	COAD EQU	(; FDC DMA CHANNEL
139	0027+=	COTC EQU	(; FDC DMA CHANNEL
140	0028+=	DMACOM EQU	(; DMA COMMAND
141	0028+=	DMAST EQU	(; READ DMA STATUS
142	0028+=	DMAMO EQU	(
143	002A+=	DMAMB EQU	(
144	0004+=	DMAWRT EQU	(; DMA WRITE COMMAND
145	0008+=	DMAREAD EQU	(; DMA READ COMMAND
146	0003+=	ENFD EQU	(; ENABLE FDC CHANNEL (CH3)
147	0007+=	DISFO EQU	(; DISABLE FDC CHANNEL (CH3)

008 (NBIOS)

148		CNTLEQU			
149	0003+=	I0BYTE EQU	3		; I/O BYTE
150	0004+=	CURDISK EQU	4		; CURRENT DISK NUMBER
151	0000+=	CR EQU	0DH		; CARRIGE RETURN
152	000A+=	LF EQU	0AH		; LINE FEED
153	0008+=	BS EQU	08H		; BACK SPACE
154	000C+=	FF EQU	0CH		; FORM FEED
155	000C+=	TOF EQU	0CH		; TOP OF FORM
156	001B+=	ESC EQU	1BH		; ESCAPE
157	001A+=	CLRSCN EQU	01AH		; CLEAR SCREEN

009 (MBIOS)

```

158 KEYEQU
159 0040+= KEYBASE EQU 00040H ; BASE ADDRESS OF KEYBOARD
160 0040+= WDKEY EQU KEYBASE ; WRITE DATA TO KEYBOARD
161 0040+= RDKEY EQU KEYBASE ; READ DATA FROM KEYBOARD
162 0041+= RSKEY EQU KEYBASE+1 ; READ KEYBOARD STATUS
163 0041+= KBELL EQU KEYBASE+1 ; RING BELL
164 0041+= KCOUNT EQU KEYBASE+1 ; COUNTRY CODE PORT
165 0002+= INPBFF EQU 02H ; INPUTBUFFER FULL
166 0001+= OUTPBFF EQU 01H ; OUTPUTBUFFER FULL
167 0080+= LGDAT EQU 80H ; LANGUAGE CODE READY FLAG
168 0001+= KBDAT EQU 01H ; DATA READY BIT
169 0001+= COUNTRY EQU 01H ; COMMAND TO GET COUNTRY CODE
170 0006+= MUSIC EQU 06H ; MUSIC COMMAND
171 DISKBFF
172 F600+= HSTBUF EQU 0F600H ;LENGTH 400H
173 FA00+= DIRBUF EQU HSTBUF+400H ;LENGTH 128
174 FA80+= ALV0 EQU DIRBUF+128 ;LENGTH 32
175 FAA0+= CSV0 EQU ALV0+32 ;LENGTH 64
176 FAE0+= ALV1 EQU CSV0+64 ;LENGTH 81
177 FB31+= CSV1 EQU ALV1+81 ;LENGTH 64
178 FB71+= ALV2 EQU CSV1+64 ;LENGTH 81
179 0000+= CSV2 EQU 0 ;NOT USED FOR HARD DISK
180 FBC2+= ALV3 EQU ALV2+81 ;LENGTH 81
181 0000+= CSV3 EQU 0 ;NOT USED FOR HARD DISK
182 FC13+= ALV4 EQU ALV3+81 ;LENGTH 81
183 0000+= CSV4 EQU 0 ;NOT USED FOR HARD DISK
184 FC64+= ALV5 EQU ALV4+81 ;LENGTH 81
185 0000+= CSV5 EQU 0 ;NOT USED FOR HARD DISK
186 FC85+= ALV6 EQU ALV5+81 ;LENGTH 81
187 0000+= CSV6 EQU 0 ;NOT USED FOR HARD DISK
188 FDC6+= ALV7 EQU ALV6+81 ;LENGTH 81
189 0000+= CSV7 EQU 0 ;NOT USED FOR HARD DISK
190 FDS7+= ALV8 EQU ALV7+81 ;LENGTH 81
191 0000+= CSV8 EQU 0 ;NOT USED FOR HARD DISK
192 FDA8+= ALV9 EQU ALV8+81 ;LENGTH 81
193 0000+= CSV9 EQU 0 ;NOT USED FOR HARD DISK
194 FDF9+= LIMBUF EQU ALV9+81 ;LENGTH 160
195 FE99+= DP80 EQU LIMBUF+160 ;LENGTH 15
196 FEA8+= DP81 EQU DP80+15 ;LENGTH 15
197 FEB7+= DP82 EQU DP81+15 ;LENGTH 15
198 FEC6+= DP83 EQU DP82+15 ;LENGTH 15
199 FED5+= DP84 EQU DP83+15 ;LENGTH 15
200 FEE4+= DP85 EQU DP84+15 ;LENGTH 15
201 FEF3+= DP86 EQU DP85+15 ;LENGTH 15
202 FF02+= DP87 EQU DP86+15 ;LENGTH 15
203 FF11+= DP88 EQU DP87+15 ;LENGTH 15
204 FF20+= DP89 EQU DP88+15 ;LENGTH 15
205 PAGE

```

010 (MBIOS)

```

206
207 ;*****
208 ;*
209 ;*          BIOS BEGINS HERE
210 ;*
211 ;*****
212
213
214
215 0000 C35800 STBIOS: JMP  CBOOT   ; COLD BOOT      0E5B
216 0003 C36F00 WBOOT:  JMP  WBOOT   ; WARM BOOT     0E6F
217 0006 C30090          JMP  CONST   ; GET CONSOLE STATUS  EC21
218 0009 C30000          JMP  CONIN   ; GET CONSOLE CHARACTER EC2F
219 000C C30000          JMP  COMOUT  ; OUTPUT A CHARACTER TO CRT EC3D
220 000F C30000          JMP  LIST   ; OUTPUT TO LIST DEVICE  EC6D
221 0012 C30000          JMP  PUNCH  ; OUTPUT TO PUNCH DEVICE EC4B
222 0015 C30000          JMP  READER ; GET CHARACTER FROM READER
223 0018 C3A001          JMP  HOME   ; HOME DISK (SET TRACK TO 0)
224 001B C35801          JMP  SELDSK ; SELECT DISK
225 001E C3A801          JMP  SETTRK ; SET TRACK
226 0021 C3B101          JMP  SETSEC ; SET SECTOR
227 0024 C38701          JMP  SETDMA ; SET DMA ADDRESS
228 0027 C3A405          JMP  READ   ; READ SECTOR
229 002A C38805          JMP  WRITE  ; WRITE SECTOR
230 002D C30000          JMP  LISTST ; GET STATUS OF THE LIST DEVICE
231 0030 C38D01          JMP  SECTRM ; TRANSLATE SECTOR NUMBER
232 0033 C33401          JMP  SPECFUN ; EXECUTE SPECIAL BIOS FUNCTIONS
233 0036 C35108          JMP  SELTYP ; RETURN PARAMETERS FOR TYPSELECTION
234
235 0039 C33808          DISPERR: JMP  DISPERO ; DISPLAY ERROR MESSAGE
236
237 003C C30000          CLRLIN:  JMP  CLRLI1  ; CLEAR ERROR LINE
238
PAGE

```

```

239
240 ;*****
241 ;*
242 ;*   INITIALIZE BDOS
243 ;*
244 ;*****
245
246 IMITBDOS:
247 003F 2161D0      LXI   H,BDOSBTL+13*2 ; HL - A(BDOS FUNCTION 13 TABLE ENTRY)
248 0042 E5         PUSH  H           ; SAVE PARAMETER
249 0043 5E         MOV   E,M           ; ACTUAL ADDR --) HL
250 0044 23         INX   H
251 0045 56         MOV   D,M
252 0046 EB         XCHG
253 0047 225900     SHLD  BDOSRA       ; STORE RESTART ADDRESS
254 004A 115200     LXI   D,RSTBDOS
255 004D E1         POP   H
256 004E 73         MOV   M,E
257 004F 23         INX   H
258 0050 72         MOV   M,D
259 0051 C9         RET
260
261
262 ;*****
263 ;*
264 ;*   BDOS RESET FUNCTION
265 ;*
266 ;*****
267
268 RSTBDOS:
269 0052 C00000     CALL  RESET        ; WARM RESET
270 0055 2A5900     LHLD  BDOSRA       ; GET BDOS RESET FUNCTION ADDR
271 0058 E9         PCHL
272
273 0059 0000     BDOSRA: DW 00      ; BDOS RESET FUNCTION ADDR
274
275 PAGE

```


012 (MBIOS)

```

276
277 ;*****
278 ;*                                     *
279 ;*      CBOOT  COLD BOOT                *
280 ;*                                     *
281 ;*****
282
283
284 CBOOT:
285 0058 F3          DI          SP,100H
286 005C 310001     LXI         SP,100H
287 005F C00000     CALL        CINIT          ; COLD INIT
288 0062 C00000     CALL        RESET          ; RESET DISKS
289 0065 CDA900     CALL        LOADSYS        ; LOAD CCP AND BDOS
290 0068 AF         XRA         A
291 0069 320400     STA         CURDISK        ; SET DRIVE A = CURRENT DRIVE
292 006C C37D00     JMP         MCRDMV
293
294
295
296 ;*****
297 ;*                                     *
298 ;*      WBOOT  WARM BOOT                *
299 ;*                                     *
300 ;*****
301
302
303 WBOOT:
304 006F 310001     LXI         SP,100H
305 0072 C00000     CALL        RESET          ; RESET DISKS
306 0075 CDA900     CALL        LOADSYS        ; LOAD CCP AND BDOS
307 0078 C00000     CALL        WINIT          ; WARM INIT
308 007B 3EFF       MVI         A,OFFH        ; SET FLAG FOR WARM BOOT
309 PAGE

```

```

310
311 ;*****
312 ;*
313 ;* MCRDMV INIT PAGE ZERO
314 ;*
315 ;*****
316
317
318 MCRDMV:
319 007D B7      ORA   A
320 007E F5      PUSH  PSW      ; IF COLD START THEN Z = ON
321 007F C03F00  CALL  INITBDOS ; INITIALIZE BDOS RESET FUNCTION
322
323 0082 3EC3    MVI   A,CC3H      ; "JMP" INSTRUCTION
324 0084 320000  STA   CPMRAM      ; WBOOT "JMP"
325 0087 320500  STA   CPMRAM+5    ; BDOS "JMP"
326 008A 2103DE  LXI   H,BIOS+3
327 008D 220100  SHLD  CPMRAM+1    ; INIT WARM BOOT "JMP" ADDRESS
328 0090 2106D0  LXI   H,BDOSMTRY
329 0093 220600  SHLD  CPMRAM+6    ; INIT BDOS "JMP" ADDRESS
330 0096 3A0400  LDA   CURDISK
331 0099 4F      MOV   C,A          ; USER ⇄ = X, DISK UNIT = Y ,XXXXYYYY
332 009A F1      POP   PSW          ; RESTORE FLAGS
333 009B 3A0E02  LDA   MODEFL      ; GET MODE FLAG
334 009E 0F      RRC
335 009F CAA300  JZ    G01          ; JUMP IF COLD BOOT
336 00A2 0F      RRC
337 00A3 D203C8  G01: JNC  CCPSTB    ; JUMP IF NO AUTO LOAD
338 00A6 C300C8  JMP  CCP          ; JUMP IF AUTO LOAD
339
340 PAGE

```

```

397
398 ;
399 ;
400 ; DISPLAY BOOT ERROR
401 ;
402 MOOSDK:
403 010B 211E04 LXI H,MOOSERM
404 010E C31401 JMP BOOTER1
405
406
407 ;
408 0111 210E04 BOOTER: LXI H,BOOTERM
409 BOOTER1:
410 0114 E5 PUSH H
411 0115 210018 LXI H,01800H
412 0118 220000 SHLD CURCOL
413 011B E1 POP H
414 011C C03900 CALL DISPERR
415 011F C00000 BOOTER2:CALL COMIN ; GET A CHARACTER
416 0122 FE00 CPI CR
417 0124 C21F01 JNZ BOOTER2 ; JUMP IF NOT CARRIGE RETURN
418 0127 3E18 MVI A,24
419 0129 C03C00 CALL CLRLIN ; CLEAR LINE
420 012C 0E1A MVI C,CLRSCH
421 012E C00000 CALL COMOUT ; CLEAR SCREEN
422 0131 C3A900 JMP LOADSYS ; RETRY
423 ;
424 ;
425 ;
426
427 PAGE

```

014 (NBIOS)

```

341
342 ;*****
343 ;#
344 ;# LOAD CCP AND BDOS
345 ;# START TRACK/SECTOR = 1/6
346 ;# LENGTH = 1600H
347 ;#
348 ;*****
349 ;
350 ;
351 LOADSYS:
352 00A9 1E00 MVI E,0 ; DISK UNIT 0 DISK PARAMETER
353 00AB CD0000 CALL DSKSET ; SELECT DISK UNIT
354 00AE 3AC002 LDA RETRYC ; SET UP RETRY COUNTER
355 00B1 47 MOV B,A ;
356
357 00B2 C5 PUSH B ; SAVE RETRY COUNTER
358 00B3 CD0000 CALL READSEC1 ; READ SECTOR 1
359 00B6 C1 POP B ; RESTORE RETRY COUNTER
360 00B7 CAC600 JZ LOADB ; JUMP IF GOOD STATUS
361 00BA 0E01 MVI C,1 ; ENRTY FOR ERROR ROUTINE
362 00BC CD0000 CALL DSKERR ; DETECT DISK ERRORS
363 00BF 05 DCR B ;
364 00C0 C2B200 JNZ LOADA ; JUMP IF RETRY
365 00C3 C31101 JMP BOOTER ; ELSE OVERRIDE
366
367 00C6 CD0000 CALL CHECKOS ;CHECK IF OS DISK LOADED
368 00C9 C20B01 JNZ NOOSDSK ;RETRY IF NO OS DISK LOADED
369 00CC 1E01 MVI E,1
370 00CE CD0000 CALL TRKSET ; SET TRACK
371 00D1 1E06 MVI E,6
372 00D3 CD0000 CALL SECSET ; SET SECTOR
373 00D6 1100C8 LXI D,CCP
374 00D9 CD0000 CALL DMASET ; SET DMA ADDRESS
375 00DC 3E03 MVI A,3
376 00DE 320000 STA BTRECS
377 00E1 CD0000 LOAD1: CALL CREAD ; READ SECTOR
378 00E4 C21101 JNZ BOOTER ; JUMP IF ERROR
379 00E7 2A0000 LHLD PMAADR
380 00EA 110002 LXI D,HSTSIZD
381 00ED 19 DAD D
382 00EE 220000 SHLD PMAADR ; SET NEW DMA ADDRESS
383 00F1 210000 LXI H,BTRECS
384 00F4 35 DCR H
385 00F5 CAFF00 JZ LOAD2 ; JUMP IF 3 SECTOR'S READ
386 00F8 210000 LXI H,HSTSEC
387 00FB 34 INR H ; NEXT SECTOR +
388 00FC C3E100 JMP LOAD1 ; READ NEXT SECTOR
389 ;
390 LOAD2:
391 00FF 1E02 MVI E,2
392 0101 CD0000 CALL TRKSET ; SET TRACK +
393 0104 CD0000 CALL R0TRAK ; READ TRACK
394 0107 C21101 JNZ BOOTER ; JUMP IF ERROR
395 010A C9 RET
396 PAGE

```

D16 (NBIOS)

```

428
429 ;
430 ;
431 ;*****
432 ;#          BIOS SPECIAL FUNCTION DISPATCHER          #
433 ;#          PARAMETER PASSING ROUTINES                #
434 ;#
435 ;*****
436 ;
437 ;
438 SPECFUN:
439 0134 79      MOV    A,C
440 0135 87      ADD    A
441 0136 214101 LXI    H,SPECTBL ; SPECIAL FUNCTION TABLE ADDRESS
442 0139 CD4CD3 CALL  OFST
443 013C 7E      MOV    A,M
444 013D 23      INX    H
445 013E 66      MOV    H,M
446 013F 6F      MOV    L,A
447 0140 E9      PCHL          ; TRANSFER TO REQUESTED FUNCTION
448
449 SPECTBL:
450 0141 5200    DW    RSTBDOS ; RESET DISK STUFF
451 0143 0000    DW    FORMATT ; FORMAT TRACK
452 0145 0000    DW    ROTRAK ; READ TRACK (ONLY FLEX DISK).
453 0147 0000    DW    DMASET ; SET DMA ADDRESS
454 0149 0000    DW    DSKSET ; SELECT DISK
455 014B 0000    DW    TRKSET ; SET TRACK
456 014D 0000    DW    SECSET ; SET SECTOR
457 014F 0000    DW    CREAD ; READ ONE PHYSICAL SECTOR
458 0151 0000    DW    CWRITE ; WRITE ONE PHYSICAL SECTOR
459 0153 0000    DW    DHOME ; HOME DISK
460 0155 0000    DW    DTRKSET ; SET TRACK TWO BYTES
461 0157 0000    DW    FREAD ; READ SECTOR WITH ERROR RETURNED
462 0159 0000    DW    FWRITE ; WRITE SECTOR WITH ERROR RETURNED
463 PAGE

```

017 (NBIOS)

```

464
465 ;*****
466 ;#
467 ;#          SELECT DISK  C = DRIVENUMBER
468 ;#
469 ;*****
470
471
472 SELDSK:
473 0158 79          MOV    A,C          ; GET DRIVE #
474 015C 210A02     LXI    H,MBRFLEX    ; A(# OF DRIVES IN THE SYSTEM)
475 015F BE          CMP    M
476 0160 D27D01     JNC    HRDSK          ; JUMP TO CHECK IF HARD DISK ACCESS
477 0163 320000     STA    SEKDSK         ; SAVE SELECTED DISK
478 0166 C00607     CALL   MOUNT          ; MOUNT DISK
479 0169 C29801     JNZ    SELCTER
480 016C 3A0000     GETDPH: LDA    SEKDSK
481 016F 87          ADD    A              ; * 2
482 0170 87          ADD    A              ; * 4
483 0171 87          ADD    A              ; * 8
484 0172 87          ADD    A              ; * 16
485 0173 210405     LXI    H,DPBASE
486 0176 C04C08     CALL   OFST          ; OFFSET --> HL
487 0179 3A0000     LDA    SEKDSK         ; GET SELECTED DISK
488 017C C9          RET
489
490 HRDSK:
491 017D 211402     LXI    H,MUMHDSK     ; TOTAL DISK NUMBER
492 0180 BE          CMP    M
493 0181 D29801     JNC    SELCTER       ; DRIVE DOESN'T EXIST
494 0184 320000     STA    SEKDSK        ; SAVE SELECTED DISK
495 0187 C06708     CALL   CKSEL         ; CHECK IF DISK SELECTED
496 018A CA6C01     JZ     GETDPH        ; JUMP IF DISK SELECTED
497 018D 3E03       MVI    A,3
498 018F C06207     CALL   FIXINIT       ; INIT DISK
499 0192 C00000     CALL   DHOME        ; RESTORE DISK TO SET STEP RATE
500 0195 C36C01     JMP    GETDPH
501
502 SELCTER:
503 0198 210000     LXI    H,0           ; RETURN ERROR
504 019B AF          XRA    A
505 019C 320400     STA    CURDISK       ; SELECT DRIVE A
506 019F C9          RET
507
508
509
510 PAGE

```


018 (NBIOS)

```

511
512 ;*****
513 ;#
514 ;# HOME HOME THE DRIVE
515 ;#
516 ;*****
517
518 HOME:
519 01A0 3A0000 LDA HSTWRT ; CHECK FOR PENDING WRITE
520 01A3 B7 ORA A
521 01A4 C2AA01 JNZ HOMED
522 01A7 320000 STA HSTACT ; CLEAR HOST ACTIVE FLAG
523 01AA C9 HOMED: RET
524
525
526 ;*****
527 ;#
528 ;# SETTRK SET TRACK
529 ;# [BC] - TRACK NUMBER
530 ;#
531 ;*****
532
533 SETTRK:
534 01AB 69 MOV L,C
535 01AC 60 MOV H,B
536 01AD 220000 SHLD SEKTRK ; SET TRACK
537 01B0 C9 RET
538
539
540 ;*****
541 ;#
542 ;# SETSEC SET SECTOR
543 ;# [C] - SECTOR NUMBER
544 ;#
545 ;*****
546
547 SETSEC:
548 01B1 79 MOV A,C
549 01B2 30 DCR A ; START SECTOR = 0
550 01B3 320000 STA SEKSEC ; SET SECTOR
551 01B6 C9 RET
552
553
554 ;*****
555 ;#
556 ;# SETDMA SET DMA ADDRESS
557 ;# [BC] - DMA ADDRESS
558 ;#
559 ;*****
560
561 SETDMA:
562 01B7 60 MOV H,B
563 01B8 69 MOV L,C
564 01B9 220000 SHLD DMAADR ; SET DMA ADDRESS
565 01BC C9 RET
566 PAGE

```

019 (NBIOS)

```
567
568 ;*****
569 ;*
570 ;*   SECTRAM   SECTOR TRANSLATE
571 ;*
572 ;*****
573 ;
574 ;TRANSLATE SECTORNUMBER C USING TABLE AT DE RETURN RESULT IN HL
575 ; IF DE = 0 ONLY DECREMENT SECTOR NUMBER
576
577 SECTRAM:
578 01B0 69      MOV    L,C          ; SECTOR NO. TO L
579 018E 2600    MVI    H,0          ; CLEAR H
580 01C0 23      INX    H          ;
581 01C1 C9      RET
582
583
584
585
586
587
588 PAGE
```

020 (NBIOS)

```

589
590 ;*****
591 ;*
592 ;*          MISC. SUBROUTINES
593 ;*
594 ;*****
595
596
597 ;*****
598 ;*
599 ;*      MULT          MULTIPLY [A] BY [HL]
600 ;*          RETURNS PRODUCT IN[HL]
601 ;*
602 ;*****
603
604
605 MULT:
606 01C2 C5          PUSH    B
607 01C3 D5          PUSH    D
608 01C4 EB          XCHG
609 01C5 210000      LXI     H,0
610 01C8 010008      LXI     B,800H      ; B IS LOOP COUNTER, C IS OVERFLOW INDICATOR
611 01CB 29          MUL1:  DAD    H
612 01CC D20001      JMC    MUL2
613 01CF 0C          INR     C          ; BUMP OVERFLOW
614 01D0 07          MUL2:  RLC
615 01D1 D20901      JMC    MUL3          ; ROTATE NEXT BIT OF MULTIPLICAND INTO CARRY
616 01D4 19          DAD    D
617 01D5 D20901      JMC    MUL3
618 01D8 0C          INR     C
619 01D9 05          MUL3:  DCR    B
620 01DA C2C801      JNZ    MUL1
621 01DD 79          MOV    A,C          ; SEE IF OVERFLOW
622 01DE B7          ORA    A          ; TEST ZERO FLAG AND RESET CARRY
623 01DF D1          POP    D
624 01E0 C1          POP    B
625 01E1 C8          RZ
626 01E2 37          STC
627 01E3 C9          RET          ; CARRY INDICATES OVERFLOW
628
629
630
631 PAGE

```

021 (NBIOS)

```

632
633
634 ;*****
635 ;*
636 ;* MOVE MOVE [BC] BYTES FROM [HL] TO [DE]
637 ;*
638 ;*****
639
640
641 01E4 78 MOVE: MOV A,B
642 01E5 B1 ORA C
643 01E6 C8 RZ ; RETURN IF ALL MOVED
644 01E7 ED DB 0EDH ; Z80 LDIR COMMAND
645 01E8 B0 DB 0B0H ; (DE) (- (HL))
646 01E9 C9 RET
647
648
649
650
651 ;*****
652 ;*
653 ;* MOVEL MOVE HL - DE
654 ;*
655 ;*****
656
657
658 MOVEL:
659 01EA 78 MOV A,B
660 01EB B1 ORA C
661 01EC C8 RZ ; RETURN IF END
662 01ED ED DB 0EDH ; Z80 LDOR COMMAND
663 01EE B8 DB 0B8H ; (DE) (- (HL)) DCX H DCX D DCX B
664 01EF C9 RET
665
666 PAGE

```

022 (NBIO5)

```

667
668 ;*****
669 ;*
670 ;* SUBHLDE HL = HL - DE
671 ;*
672 ;*****
673
674
675 SUBHLDE:
676 01F0 B7 ORA A ;CLEAR CARRY
677 01F1 ED DB 0EDH ; Z80 SBC HL,DE COMMAND
678 01F2 52 DB 052H ; SUBTRACT WITH CARRY
679 01F3 C9 RET
680
681
682
683 ;*****
684 ;*
685 ;* COMPDEHL COMPARE DE WITH HL
686 ;*
687 ;* CARRY ZERO
688 ;* DE < HL 0 0
689 ;* DE = HL 0 1
690 ;* DE > HL 1 0
691 ;*
692 ;*****
693
694
695 COMPDEHL:
696 01F4 7C MOV A,H
697 01F5 BA CMP D
698 01F6 D8 RC
699 01F7 C0 RNZ
700 01F8 7D MOV A,L
701 01F9 B8 CMP E
702 01FA C9 RET
703 PAGE

```

```

704
705
706
707
708
709 ;*****
710 ;*****
711 ;*****
712 ;***** CONFIG AREA *****
713 ;*****
714 ;*****
715 ;*****
716
717 0005 = FILLER EQU 200H-($-ST&IOS)
718
719 01FB DS FILLER
720
721
722 0200 0A02 MMAREA: DW SPAREA ; SPECIAL AREA (CONFIG BYTES)
723 0202 1502 MFMTBL: DW FUNTBL ; START ADDRESS OF FUNCTION TABLE
724 0204 1603 MCRTTBL: DW CRITBL ; START OF CRT TABLE
725 0206 EE03 MKEYTBL: DW KEYTBL ; START OF KEYTABLE
726 0208 0E04 MMESS: DW ERRORMSG ; ERROR MESSAGES
727 SPAREA:
728
729
730 020A 02 NBRFLEX: DB 2 ; # OF FLEX DRIVES
731 020B 81 IOBSET: DB 10$00$00$01B ; INITIAL IOBYTE
732 020C 05 RETRYC: DB 5 ; RETRY COUNTER
733 020D 05 RSTC: DB 5 ; RESTORE COUNTER
734
735 020E 03 MODEFL: DB 3 ; 0 - NO AUTO LOAD
736 ; 1 - AUTO LOAD ON COLD BOOT
737 ; 2 - AUTO LOAD ON WARM BOOT
738 ; 3 - AUTO LOAD ON COLD AND WARM BOOT IF
739 ; CCP BUFFER LENGTH > 0
740 ;
741 020F 00 CONFIGFL: DB 0 ; CONFIGURE FLAG IF SET THEN IGNORE FNCT
742
743 0210 79 M1RS232: DB 79H ; 1 STOP BIT , EVEN PARITY , PARITY ENABLED
744 ; 7 BIT CHARACTER , ASYNCHRON
745 0211 3E M2RS232: DB 3EH ; INTERNAL CLOCKS , 9600 BAUD
746
747 0212 01 CONFVER: DB 01H ; VERSION NUMBER OF CONFIG
748 ; REL 02.03.03 HAS VER 0 ADDRESSED BY
749 ; PVR232 WHICH IS UNUSED IN 02.03.03
750
751 0213 00 PVR232: DB 00H ; PROTOKOL VEKTOR
752
753 0214 02 NUMHDSK: DB 02 ; TOTAL NUMBER OF DRIVES HARD + FLEX DISKS
754
755 PAGE

```


024 (NBIO5)

```

756
757 ;*****
758 ;*****
759 ;*****
760 ;***** FUNCTION TABLE *****
761 ;*****
762 ;*****
763 ;*****
764 ;
765 ;
766 ;
767
768
769 0215 = FUNTBL EQU $
770
771 0215 08 F1: DB LEN1
772 0216 4449522041 DB 'DIR A:',CR
773 0008 = LEN1 EQU $-F1
774
775
776 021D 08 F2: DB LEN2
777 021E 464F524041 DB 'FORMAT',CR
778 0008 = LEN2 EQU $-F2
779
780
781 0225 08 F3: DB LEN3
782 0226 5359534745 DB 'SYSGEN',CR
783 0008 = LEN3 EQU $-F3
784
785
786 022D 11 F4: DB LEN4
787 022E 5049502042 DB 'PIP B:=A:*.#(V)',CR
788 0011 = LEN4 EQU $-F4
789
790
791 023E 08 F5: DB LEN5
792 023F 4449522042 DB 'DIR B:',CR
793 0008 = LEN5 EQU $-F5
794
795
796 0246 08 F6: DB LEN6
797 0247 434F4E4649 DB 'CONFIG',CR
798 0008 = LEN6 EQU $-F6
799
800 PAGE

```

025 (NB10S)

```

801
802 024E 08      F7:   08    LEN7
803 024F 4449534349  08    'DISCIT',CR
804 0008 =      LEN7   EQU    $-F7
805
806 0256 11      F8:   08    LEN8
807 0257 5049502041  08    'PIF A:=B:*.#[V]',CR
808 0011 =      LEN8   EQU    $-F8
809
810
811 0267 0A      F9:   08    LEN9
812 0268 4558434841  08    'EXCHANGE',CR
813 000A =      LEN9   EQU    $-F9
814
815
816 0271 09      F10:  08    LEN10
817 0272 5354415420  08    'STAT A:',CR
818 0009 =      LEN10  EQU    $-F10
819
820
821 027A 0C      F11:  08 LEN11
822 027B 5354415420  08    'STAT B:=*.*',CR
823 000C =      LEN11  EQU    $-F11
824
825
826 0286 04      F12:  08    LEN12
827 0287 463132    08    'F12'
828 0004 =      LEN12  EQU    $-F12
829
830
831 028A 04      F13:  08    LEN13
832 028B 463133    08    'F13'
833 0004 =      LEN13  EQU    $-F13
834
835
836 028E 08      F14:  08    LEN14
837 028F 4C41444445  08    'LADDER',CR
838 0008 =      LEN14  EQU    $-F14
839
840
841 0296 09      F15:  08    LEN15
842 0297 4341544348  08    'CATCHUM',CR
843 0009 =      LEN15  EQU    $-F15
844
845
846 029F 07      F16:  08    LEN16
847 02A0 4445404F35  08    'DEMOS',CR
848 0007 =      LEN16  EQU    $-F16
849
850                                PAGE

```

026 (NB105)

```

851
852 02A6 07      F17:  DB      LEN17
853 02A7 434C4F434B  DB      'CLOCK',CR
854 0007 =      LEN17 EQU      $-F17
855
856 02AD 07      F18:  DB      LEN18
857 02AE 4055534943  DB      'MUSIC',CR
858 0007 =      LEN18 EQU      $-F18
859
860
861 02B4 07      F19:  DB      LEN19
862 02B5 5645474153  DB      'VEGAS',CR
863 0007 =      LEN19 EQU      $-F19
864
865
866 02B8 04      F20:  DB      LEN20
867 02B9 463230      DB      'F20'
868 0004 =      LEN20 EQU      $-F20
869
870 02BF 00      F21:  DB      0                ; LAST FUNCTION
871
872 0055 =      FFILLER EQU      256-($-FUNTBL)    ; TOTAL FUNTBL HAS 256 BYTES
873 02C0      DS      FFILLER
874 0315 00      FNTBLND:DB      0                ; END OF FUNCTION TABLE
875
876                                PAGE

```

```

877
878 ;*****
879 ;#
880 ;# LANGUAGE TRANSLATION TABLE
881 ;#
882 ;*****
883
884
885
886
887
888 CRTTBL: ; START OF TRANSLATION TABLE
889 ;
890 ;
891 ;
892 0316 03 LVAR0: DB VAR0L
893 0317 8A2E US: DB 8AH,2EH
894 0003 = VAR0L EQU $-LVAR0
895
896 0319 07 LVAR1: DB VAR1L
897 031A 5E0E23038AUK: DB 5EH,0EH,23H,03H,8AH,2EH
898 0007 = VAR1L EQU $-LVAR1
899
900 0320 15 LVAR2: DB VAR2L
901 0321 5B005C085DFRANCE: DB 5BH,0DH,5CH,08H,5DH,1CH,4DH,0AH,7BH,14H,7CH,1AH
902 032D 7D0B7E0F23 DB 7DH,0BH,7EH,0FH,23H,03H,27H,0CH
903 0015 = VAR2L EQU $-LVAR2
904
905 0335 13 LVAR3: DB VAR3L
906 0336 5B005C065DGERMANY: DB 5BH,0DH,5CH,06H,5DH,09H,4DH,1CH,7BH,10H,7CH,16H
907 0342 7D197E1E27 DB 7DH,19H,7EH,1EH,27H,0CH
908 0013 = VAR3L EQU $-LVAR3
909
910 0348 13 LVAR4: DB VAR4L
911 0349 5B005C065DSWEDEN: DB 5BH,0DH,5CH,06H,5DH,02H,24H,13H,7BH,10H,7CH,16H
912 0355 7D127E0F27 DB 7DH,12H,7EH,0FH,27H,0CH
913 0013 = VAR4L EQU $-LVAR4
914
915 0358 13 LVAR5: DB VAR5L
916 035C 5B015C075DDANSK: DB 5BH,01H,5CH,07H,5DH,02H,23H,03H,7BH,11H,7CH,17H
917 0368 7D127E0F27 DB 7DH,12H,7EH,0FH,27H,0CH
918 0013 = VAR5L EQU $-LVAR5
919
920 036E 0D LVAR6: DB VAR6L
921 036F 5B1F5C055DKSPAIN: DB 5BH,1FH,5CH,05H,5DH,1DH,27H,0CH,7CH,15H,23H,03H
922 000D = VAR6L EQU $-LVAR6
923
924 037B 17 LVAR7: DB VAR7L
925 037C 5B005C085DITALY: DB 5BH,0DH,5CH,08H,5DH,14H,23H,03H,4DH,1CH,7BH,0AH
926 0388 7C187D0B7E DB 7CH,18H,7DH,0BH,7EH,18H,6DH,1AH,27H,0CH
927 0017 = VAR7L EQU $-LVAR7
928
929 0392 15 LVAR8: DB VAR8L
930 0393 2303270C40SWISS12: DB 23H,03H,27H,0CH,4DH,08H,5BH,0AH,5CH,14H,5DH,0BH
931 039F 7B107C167D DB 7BH,10H,7CH,16H,7DH,19H,7EH,0FH
932 0015 = VAR8L EQU $-LVAR8

```

028 (NB10S)

933
934 03A7 01 LVAR9: DB VAR9L
935 CANADA1:
936 0001 = VAR9L EQU \$-LVAR9
937
938 03A8 0F LVAR10: DB VAR10L
939 03A9 270C400A5CCANADA2:DB 27H,0CH,40H,0AH,5CH,08H,7BH,14H,7CH,9FH,7DH,0BH
940 03B5 7E0F DB 7EH,0FH
941 000F = VAR10L EQU \$-LVAR10
942
943 03B7 11 LVAR11: DB VAR11L
944 03B8 270C5B835CSAFRICA:DB 27H,0CH,5BH,83H,5CH,84H,5DH,82H,7BH,93H
945 03C2 7C947D927E DB 7CH,94H,7DH,92H,7EH,0FH
946 0011 = VAR11L EQU \$-LVAR11
947
948 03C8 11 LVAR12: DB VAR12L
949 63C9 2303270C5BPORTUG: DB 23H,03H,27H,0CH,5BH,80H,5CH,81H,5DH,85H,7BH,90H
950 03D5 7C917D08 DB 7CH,91H,7DH,08H
951 0011 = VAR12L EQU \$-LVAR12
952
953 03D9 15 LVAR13: DB VAR13L
954 03DA 408C5B885CYUGOSL: DB 40H,8CH,5BH,8BH,5CH,88H,5DH,89H,5EH,8AH,60H,9CH
955 03E6 7B9B7C927D DB 7BH,9BH,7CH,9BH,7DH,99H,7EH,9AH
956 0015 = VAR13L EQU \$-LVAR13
957
958 PAGE

```

959
960 ;*****
961 ;*
962 ;*      KEYBOARD TRANSLATION TABLE      *
963 ;*      80H-9FH                          *
964 ;*****
965
966
967 KEYTBL:
968
969 03EE 80      08      80H      ; 80H
970 03EF 17      08      17H      ; 81H  HOME
971 03F0 13      08      13H      ; 82H  CURSOR LEFT
972 03F1 18      08      18H      ; 83H  CURSOR DOWN
973 03F2 05      08      05H      ; 84H  CURSOR UP
974 03F3 04      08      04H      ; 85H  CURSOR RIGTH
975 03F4 18      08      18H      ; 86H  CLR
976 03F5 87      08      87H      ; 87H
977 03F6 00      08      00H      ; 88H  CARRIGE RETURN
978 03F7 89      08      89H      ; 89H
979 03F8 2C      08      2CH      ; 8AH  COMMA
980 03F9 08      08      08H      ; 8BH  BACKSPACE
981 03FA 8C      08      8CH      ; 8CH
982 03FB 8D      08      8DH      ; 8DH
983 03FC 8E      08      8EH      ; 8EH
984 03FD 8F      08      8FH      ; 8FH
985
986 03FE 90      08      90H      ; 90H
987 03FF 17      08      17H      ; 91H  HOME
988 0400 13      08      13H      ; 92H  ←
989 0401 18      08      18H      ; 93H  ↓
990 0402 05      08      05H      ; 94H  ↑
991 0403 04      08      04H      ; 95H  →
992 0404 18      08      18H      ; 96H  CLR
993 0405 97      08      97H      ; 97H
994 0406 00      08      00H      ; 98H  CARRIGE RETURN
995 0407 99      08      99H      ; 99H
996 0408 2C      08      2CH      ; 9AH  COMMA
997 0409 08      08      08H      ; 9BH  BACKSPACE
998 040A 9C      08      9CH      ; 9CH
999 040B 9D      08      9DH      ; 9DH
1000 040C 9E      08      9EH      ; 9EH
1001 040D 9F      08      9FH      ; 9FH
1002

```

```

1025
1026 ;*****
1027 ;#          DISK DEFINITIONS          #
1028 ;*****
1029
1030 DPBASE:
1031
1032          DISKS  10          ; 10 DISKS ( 2 FLEX , 8 HARD )
1033 0504+=      DPBASE EQU $          ;BASE OF DISK PARAMETER BLOCKS
1034 0504+00000000 DPE0: DW 0000H,0000H ;TRANSLATE TABLE
1035 0508+00000000          DW 0000H,0000H ;SCRATCH AREA
1036 050C+00FA99FE          DW DIRBUF,DPB0 ;DIR BUFF,PARM BLOCK
1037 0510+A0FA80FA          DW CSV0,ALV0 ;CHECK, ALLOC VECTORS
1038 0514+00000000 DPE1: DW 0000H,0000H ;TRANSLATE TABLE
1039 0518+00000000          DW 0000H,0000H ;SCRATCH AREA
1040 051C+00FAA8FE          DW DIRBUF,DPB1 ;DIR BUFF,PARM BLOCK
1041 0520+31FBEDFA          DW CSV1,ALV1 ;CHECK, ALLOC VECTORS
1042 0524+00000000 DPE2: DW 0000H,0000H ;TRANSLATE TABLE
1043 0528+00000000          DW 0000H,0000H ;SCRATCH AREA
1044 052C+00FAB7FE          DW DIRBUF,DPB2 ;DIR BUFF,PARM BLOCK
1045 0530+000071FB          DW CSV2,ALV2 ;CHECK, ALLOC VECTORS
1046 0534+00000000 DPE3: DW 0000H,0000H ;TRANSLATE TABLE
1047 0538+00000000          DW 0000H,0000H ;SCRATCH AREA
1048 053C+00FAC6FE          DW DIRBUF,DPB3 ;DIR BUFF,PARM BLOCK
1049 0540+0000C2FB          DW CSV3,ALV3 ;CHECK, ALLOC VECTORS
1050 0544+00000000 DPE4: DW 0000H,0000H ;TRANSLATE TABLE
1051 0548+00000000          DW 0000H,0000H ;SCRATCH AREA
1052 054C+00FAD5FE          DW DIRBUF,DPB4 ;DIR BUFF,PARM BLOCK
1053 0550+000013FC          DW CSV4,ALV4 ;CHECK, ALLOC VECTORS
1054 0554+00000000 DPE5: DW 0000H,0000H ;TRANSLATE TABLE
1055 0558+00000000          DW 0000H,0000H ;SCRATCH AREA
1056 055C+00FAE4FE          DW DIRBUF,DPB5 ;DIR BUFF,PARM BLOCK
1057 0560+000064FC          DW CSV5,ALV5 ;CHECK, ALLOC VECTORS
1058 0564+00000000 DPE6: DW 0000H,0000H ;TRANSLATE TABLE
1059 0568+00000000          DW 0000H,0000H ;SCRATCH AREA
1060 056C+00FAF3FE          DW DIRBUF,DPB6 ;DIR BUFF,PARM BLOCK
1061 0570+000085FC          DW CSV6,ALV6 ;CHECK, ALLOC VECTORS
1062 0574+00000000 DPE7: DW 0000H,0000H ;TRANSLATE TABLE
1063 0578+00000000          DW 0000H,0000H ;SCRATCH AREA
1064 057C+00FA02FF          DW DIRBUF,DPB7 ;DIR BUFF,PARM BLOCK
1065 0580+000006FD          DW CSV7,ALV7 ;CHECK, ALLOC VECTORS
1066 0584+00000000 DPE8: DW 0000H,0000H ;TRANSLATE TABLE
1067 0588+00000000          DW 0000H,0000H ;SCRATCH AREA
1068 058C+00FA11FF          DW DIRBUF,DPB8 ;DIR BUFF,PARM BLOCK
1069 0590+000057FD          DW CSV8,ALV8 ;CHECK, ALLOC VECTORS
1070 0594+00000000 DPE9: DW 0000H,0000H ;TRANSLATE TABLE
1071 0598+00000000          DW 0000H,0000H ;SCRATCH AREA
1072 059C+00FA20FF          DW DIRBUF,DPB9 ;DIR BUFF,PARM BLOCK
1073 05A0+0000A8FD          DW CSV9,ALV9 ;CHECK, ALLOC VECTORS
1074
1075
1076
1077
1078
1079          PAGE

```


030 (NBIO\$)

```

1003
1004 ;*****
1005 ;*
1006 ;* ERROR MESSAGES
1007 ;*
1008 ;*****
1009
1010
1011 ERRORMSG:
1012
1013 040E 424F4F5420B00TERM:DB 'BOOT ERROR (CR)$'
1014 041E 413A204D4FN00SERM:DB 'A: MOUNT O.S. DISK (CR)$'
1015 0436 203A204E4FH0TRDYM:DB ' : NOT READY (R)$'
1016 0447 203A204E4FN0DYRX: DB ' : NOT READY (R/X)$'
1017 045A 203A205752WRPRCTM:DB ' : WRITE PROTECT (R/O/X)$'
1018 0473 203A20492FIOERRM: DB ' : I/O ERROR (R/O/X)$'
1019 0488 203A204641FATERRM:DB ' : FATAL ERROR (R/O/X)$'
1020 049F 464E435420FNERRM: DB 'FNCT TABLE FULL (CR)$'
1021 04B4 DS 80 ; PATCH AREA
1022
1023
1024 PAGE

```

032 (NBIOS)

```

1080
1081 ;*****
1082 ;#          DISK DEBLOCKING MACROS          #
1083 ;*****
1084
1085 ;          UTILITY MACRO TO COMPUTE SECTOR MASK
1086
1087 SMASK  MACRO  HBLK          ;; COMPUTE LOG2(HBLK) RETURN AX
1088 @Y     SET    HBLK
1089 @X     SET    0
1090       REPT    8
1091       IF     @Y = 1
1092       EXITM
1093       ENDF
1094 ;;     @Y IS NOT 1, SHIFT RIGHT ONE POSITION
1095 @Y     SET    @Y SHR 1
1096 @X     SET    @X + 1
1097       ENDM
1098       ENDM
1099
1100 HOSTDEQU
1101 0800+= BLKSIZ EQU 2048          ;CP/M ALLOCATION SIZE
1102 0200+= HSTSIZD EQU 512          ;HOST DISK SECTOR SIZE
1103 0008+= HSTSPTD EQU 08          ;HOST DISK SECTORS/TRK
1104 0004+= HSTBLKD EQU HSTSIZD/128 ;CP/M SECTS/HOST BUFF
1105 0020+= CPMSPTD EQU HSTBLKD * HSTSPTD ;CP/M SECTORS/TRACK
1106 0003+= SECMSKD EQU HSTBLKD-1   ;SECTOR MASK
1107 0002+= SECSHFD EQU @X          ;LOG2(HSTBLK)
1108
1109
1110
1111 WRITEEEQ
1112 0000+= WRALL EQU 0             ;WRITE TO ALLOCATED
1113 0001+= WRDIR EQU 1            ;WRITE TO DIRECTORY
1114 0002+= WRUAL EQU 2            ;WRITE TO UNALLOCATED

```

```

1115
1116
1117
1118 ;*****
1119 ;*                                     *
1120 ;* READ      LOGICAL READ ROUTINE    *
1121 ;*                                     *
1122 ;*****
1123
1124
1125 READ:                                ; READ THE SELECTED CP/M SECTOR
1126 05A4 AF          XRA      A
1127 05A5 320000     STA      UNACNT
1128 05A8 3E01       MVI      A,1
1129 05AA 320000     STA      READOP      ; READ OPERATION
1130 05AD 320000     sta      rsflag      ;must read data
1131 05B0 3E02       mov      a,wrual
1132 05B2 320000     sta      wrtype      ;treat as unalloc
1133 05B5 C32206     jmp      rwoper      ;to perform the read
1134 ;
1135 ;*****
1136 ;*                                     *
1137 ;* WRITE     LOGICAL WRITE ROUTINE   *
1138 ;*                                     *
1139 ;*****
1140
1141 WRITE:
1142 ;write the selected CP/M sector
1143 05B8 AF          xra      a            ;0 to accumulator
1144 05B9 320000     sta      readop      ;not a read operation
1145 05BC 79         mov      a,c          ;write type in c
1146 05BD 320000     sta      wrtype
1147 05C0 FE02       cpi      wrual      ;write unallocated?
1148 05C2 C2DE05     jnz      chkuna      ;check for unalloc
1149 ;
1150 ; write to unallocated, set parameters
1151 05C5 210000     LXI      H,SEKDSK      ; FETCH BLOCKSIZE / 128
1152 05C8 4E         MOV      C,M
1153 05C9 210108     LXI      H,DSKCNT
1154 05CC C05207     CALL     DSKGET
1155 05CF 79         MOV      A,C          ;next unalloc recs
1156 05D0 320000     sta      unacnt
1157 05D3 210000     LXI      H,sekdsk      ;
1158 05D6 110000     LXI      D,unadsk      ;unadsk = sekdsk
1159 05D9 010400     LXI      B,4          ;unatrk = sectrk
1160 05DC ED         DB      0EDH        ;unasec = seksec
1161 05DD 80         DB      0B0H        ; Z80 LDIR COMMAND
1162
1163
1164 PAGE

```

:034 (NB IOS)

```

1165
1166      chkuna:
1167      ;check for write to unallocated sector
1168 05DE 3A0000      lda      unacnt      ;any unalloc remain?
1169 05E1 B7         ora      a
1170 05E2 CA1A06     jz      alloc      ;skip if not
1171      ;
1172      ; more unallocated records remain
1173 05E5 3D         dcr      a          ;unacnt = unacnt-1
1174 05E6 320000     sta      unacnt
1175      ;
1176 05E9 210000     LXI     H,UNADSK   ; COMPARE DSK, TRK, SEC
1177 05EC 110000     LXI     D,SEKDSK
1178 05EF 0604       MVI     B,4
1179 05F1 CD5D08     CALL    CMPRE
1180 05F4 C21A06     JNZ     ALLOC
1181 05F7 2B         DCX     H
1182
1183      ;
1184      ; match, move to next sector for future ref
1185 05F8 34         inr     a          ;unasec = unasec+1
1186 05F9 E5         PUSH   H          ;end of track?
1187 05FA 210000     LXI     H,UNADSK   ; FETCH CP/M SECTORS PER TRACK
1188 05FD 4E         MOV     C,M
1189 05FE 21C107     LXI     H,DSKSPT
1190 0601 CD5207     CALL    DSKGET
1191 0604 E1         POP     H
1192 0605 7E         MOV     A,M
1193 0606 B9         CMP     C
1194 0607 DA1306     jc     noovf      ;skip if no overflow
1195      ;
1196      ; overflow to next track
1197 060A 3600       mvi     a,0        ;unasec = 0
1198 060C 2A0000     lhld   unatrkl
1199 060F 23         inx     h
1200 0610 220000     shld   unatrkl    ;unatrkl = unatrkl+1
1201      ;
1202      noovf:
1203      ;match found, mark as unnecessary read
1204 0613 AF         xra     a          ;0 to accumulator
1205 0614 320000     sta     rsflag     ;rsflag = 0
1206 0617 C32206     jmp     rwoper     ;to perform the write
1207      ;
1208      alloc:
1209      ;not an unallocated record, requires pre-read
1210 061A AF         xra     a          ;0 to accum
1211 061B 320000     sta     unacnt     ;unacnt = 0
1212 061E 3C         inr     a          ;1 to accum
1213 061F 320000     sta     rsflag     ;rsflag = 1
1214
1215
1216      PAGE

```

```

1217
1218 ;*****
1219 ;*
1220 ;*   Common code for READ and WRITE follows
1221 ;*
1222 ;*****
1223 ruoper:
1224 ;enter here to perform the read/write
1225 0622 AF      xra    a           ;zero to accum
1226 0623 320000 sta    erflag        ;no errors (yet)
1227 0626 210000 LXI    H,SEKDSK      ; FETCH CODED SECTORLENGTH
1228 0629 4E      MOV    C,M
1229 062A 218907 LXI    H,DSKSLC
1230 062D C05207 CALL   DSKGET
1231 0630 0C      INR    C
1232 0631 3A0000 LDA    SEKSEC        ; COMPUTE HOST SECTOR
1233 0634 C33906 JMP    SHIFTE
1234 0637 B7      SHIFT: ORA    A           ; CARRY = 0
1235 0638 1F      RAR
1236 0639 0D      SHIFTE: DCR   C           ; DECREMENT LOOP COUNTER
1237 063A C23706 JNZ    SHIFT
1238 063D 320000 sta    sekfst        ;host sector to seek
1239 ;
1240 ;   active host sector?
1241 0640 210000 lxi    h,hstact      ;host active flag
1242 0643 7E      mov    a,a
1243 0644 3601      movi   a,1           ;always becomes 1
1244 0646 B7      ora    a           ;was it already?
1245 0647 CA6906 jz     filhst        ;fill host if not
1246 ;
1247 ;   host buffer active, same as seek buffer?
1248 064A 210000 LXI    H,HSTDISK     ;COMPARE DSK AND TRK
1249 064D 110000 LXI    D,SEKDSK
1250 0650 0603      MVI    B,3
1251 0652 C05D08 CALL   CMPRE
1252 0655 C26206 JNZ    NOMATCH
1253 ;
1254 ;   same disk, same track, same buffer?
1255 0658 3A0000 lda    sekfst
1256 065B 210000 lxi    h,hstsec      ;sekfst = hstsec?
1257 065E BE      cap    a
1258 065F CA8606 jz     match        ;skip if match
1259 ;
1260 nomatch:
1261 ;proper disk, but not correct sector
1262 0662 3A0000 lda    hstwt         ;host written?
1263 0665 E7      ora    a
1264 0666 C41807 cnz   wrttenst      ;clear host buff
1265 ;
1266 filhst:
1267 ;may have to fill the host buffer
1268 0669 3A0000 lda    sekdst
1269 066C 320000 sta    hstdsk
1270 066F 2A0000 lhld  sektrk
1271 0672 220000 shld  hsttrk
1272 0675 3A0000 lda    sekfst

```

```

036 (NB10S)

1273 0678 320000      sta  hstsec
1274 067B 3A0000      lda  rsflag          ;need to read?
1275 067E B7          ora  a
1276 067F C41A07      cnz  readhst        ;yes, if 1
1277 0682 AF          xra  a              ;0 to accum
1278 0683 320000      sta  hstwrtr        ;no pending write
1279
;
1280      match:
1281          ;copy data to or from buffer
1282 0686 210000      LXI  H,HSTDSK
1283 0689 4E          MOV  C,M
1284 068A 21B007      LXI  H,DSKSMA        ;GET SECTORMASK FROM TABLE
1285 068D C05207      CALL DSKGET
1286 0690 3A0000      lda  seksec         ;mask buffer number
1287 0693 A1          ANA  C              ;least signif bits
1288 0694 6F          mov  l,a            ;ready to shift
1289 0695 2609      mov  h,0            ;double count
1290          rept 7      ;shift left 7
1291          dad  h
1292          endm
1293 0697+29          DAD  H
1294 0698+29          DAD  H
1295 0699+29          DAD  H
1296 069A+29          DAD  H
1297 069B+29          DAD  H
1298 069C+29          DAD  H
1299 069D+29          DAD  H
1300          ; hl has relative host buffer address
1301 069E 1100F6      Lxi  d,hstbuf
1302 06A1 19          dad  d              ;hl = host address
1303 06A2 EB          xchg          ;now in DE
1304 06A3 2A0000      lhld dmaadr        ;get/put CP/M data
1305 06A6 018000      LXI  B,128         ;length of move
1306 06A9 3A0000      lda  readop        ;which way?
1307 06AC B7          ora  a
1308 06AD C2B606      jnz  rmove         ;skip if read
1309
;
1310          ; write operation, mark and switch direction
1311 06B0 3E01          mov  a,1
1312 06B2 320000      sta  hstwrtr        ;hstwrtr = 1
1313 06B5 EB          xchg          ;source/dest swap
1314
;
1315      rmove:
1316          ;C initially 128, DE is source, HL is dest
1317 06B6 EB          XCHG          ;SOURCE / DESTINATION SWAP
1318 06B7 ED          DB  0EDH        ;Z80 LDIR COMMAND (DE) (- (HL))
1319 06B8 B0          DB  0B0H
1320
;
1321          ; CHECK IF DATA CONVERSATION IS REQUIRED
1322
;
1323 06B9 E5          PUSH  H              ;SAVE HL
1324 06BA 210000      LXI  H,HSTDSK
1325 06BD 4E          MOV  C,M              ;GET DISK NUMBER
1326 06BE 21B507      LXI  H,DSKSID
1327 06C1 C05207      CALL DSKGET        ;GET SPECIAL BYTE
1328 06C4 E1          POP   H              ;RESTORE HL

```

```

037 (NBIOS)

1329 06C5 79      MOV    A,C
1330 06C6 07      RLC
1331 06C7 D2D706  JNC    RWM1      ;JUMP IF NO CONVERSION REQ.
1332 06CA 2B      DCX    H          ;CORRECT SOURCE ADDRESS
1333 06CB 1B      DCX    D          ;CORRECT DESTINATION ADDRESS
1334 06CC 0680    MVI    B,128
1335              RWM2:
1336 06CE 7E      MOV    A,M
1337 06CF 2F      CMA                    ;COMPLEMENT IT
1338 06D0 12      STAX   D
1339 06D1 2B      DCX    H
1340 06D2 1B      DCX    D
1341 06D3 05      DCR    B
1342 06D4 C2CE06  JNZ    RWM2      ;MOVE 128 BYTES
1343              RWM1:
1344              ;
1345              ; data has been moved to/from host buffer
1346              ;
1347 06D7 3A0000   LDA    wrtype      ;write type
1348 06DA FE01    CPI    wrdir       ;to directory?
1349 06DC 3A0000   LDA    ERFLAG
1350 06DF C2EE06  JNZ    RWEND      ;END OF THE R/W OPERATION
1351              ;
1352              ; clear host buffer for directory write
1353 06E2 B7      ORA    a           ;errors?
1354 06E3 C0      RNZ                    ;skip if so
1355 06E4 AF      XRA    a           ;0 to accum
1356 06E5 320000   STA    hstwr      ;buffer written
1357 06E8 C01807  CALL   writehst
1358 06EB 3A0000   LDA    erflag
1359 06EE B7      RWEND: ORA    A          ;SET STATUS
1360 06EF C9      RET
1361              ;
1362              ;
1363              PAGE

```

```

038 (NBIOS)

1364
1365
1366 06F0 3A0000   FLUSH: LDA    HSTWRT
1367 06F3 B7      ORA    A
1368 06F4 C8      RZ                    ; RETURN IF NO WRITE PENDING
1369 06F5 CDFB06  CALL   CLOSE         ; CLOSE IF NO WRITE PENDING
1370 06F8 C31807  JMP    WRITEHST
1371
1372 06F8 AF      CLOSE: XRA    A
1373 06FC 320000   STA    UNACNT       ; RESET UNALLOCATED COUNTER
1374 06FF 320000   STA    HSTACT       ; RESET HOST ACTIVE FLAG
1375 0702 320000   STA    HSTWRT       ; RESET HOST WRITE PENDING FLAG
1376 0705 C9      RET
1377              PAGE

```


039 (MBIOS)

```

1378
1379 ;*****
1380 ;#
1381 ;* MOUNT SELECT DRIVE
1382 ;#
1383 ;*****
1384
1385
1386 MOUNT:
1387 0706 CD6708 CALL CKSEL ; CHECK IF DISK SELECTED
1388 0709 C8 RZ ; DRIVE IS ALREADY INITIALIZED
1389 070A CDF006 CALL FLUSH ; WRITE OUT BUFFER IF PENDING
1390 0700 CDFB06 CALL CLOSE ; AND CLOSE IF READ
1391 0710 3A0000 LDA SEKDSK
1392 0713 4F MOV C,A ; DRIVE NO. IN C
1393 0714 CD6F07 CALL FDIINIT ; TRY TO INITIALIZE DISK
1394 0717 C9 RET ; FDIINIT RETURNS ZERO IF O.K.
1395
1396
1397
1398
1399 PAGE
    
```

```

1400
1401
1402
1403
1404 ;*****
1405 ;*
1406 ;*   HOST DISK READ / WRITE
1407 ;*
1408 ;*****
1409
1410
1411 WRITEHST:
1412 0718 AF      XRA   A           ; WRITE HOST
1413 0719 21      DB    21H        ; EXCHANGES THE REQ. "JMP"
1414 READHST:      ; READ HOST
1415 071A 3E01     MVI   A,1
1416 071C 320000  STA   DSKCMD
1417
1418 071F 3A0000   LDA   HSTDSK      ; TEST IF SECTOR TRANSLATE NECESSARY
1419 0722 4F      MOV   C,A
1420 0723 21F007  LXI   H,DSKDBL
1421 0726 CD5207  CALL  DSKGET
1422 0729 79      MOV   A,C           ; GET BYTE
1423 072A B7      ORA   A
1424 072B 3A0000   LDA   HSTSEC
1425 072E 320000  STA   SAVESEC      ; SAVE HSTSEC
1426 0731 CA3E07  JZ    DSKOP2       ; ZERO = 1 -> NO TRANSLATE ( INCREMENT)
1427 0734 211108  LXI   H,XLT        ; LOAD TRANSLATE TABLE ADDRESS
1428 0737 CD4C08  CALL  UFST         ; ADD SECTORNO TO STARTADDRESS
1429 073A 7E      MOV   A,M           ; GET TRANSLATED SECTOR
1430 073B C33F07  JMP   DSKOP3
1431 073E 3C      DSKOP2: INR   A           ; INCREMENT SECTOR NO
1432 073F 320000  DSKOP3: STA   HSTSEC      ; STORE SECTORNO
1433 0742 2100F6  LXI   H,HSTBUF
1434 0745 220000  DSKOP1: SHLD  PMAADR      ; SET DMA - ADDRESS
1435 0748 CD0000  CALL  DISKIO       ; CALL DISKROUTINES
1436 074B 3A0000   LDA   SAVESEC      ; FETCH OLD HSTSEC
1437 074E 320000  STA   HSTSEC      ; RESTORE IT
1438 0751 C9      RET
1439 PAGE

```

041 (NBIO5)

```

1440
1441      ;
1442      ;*****
1443      ;*
1444      ;*   DSKGET RETURNS A VALUE FROM THE DISK TYPE TABLE   *
1445      ;*   C:= (DSKxxx + (DSKTYP + DISKNO.))                   *
1446      ;*
1447      ;*   IN:   HL = DSKxxx                                     *
1448      ;*         C = DISKNO.                                    *
1449      ;*   OUT:  C = VALUE FROM TABLE                         *
1450      ;*         B = 0                                          *
1451      ;*   CHANGED REG. BC          UNCHANGED   A DE HL      *
1452      ;*
1453      ;*****
1454
1455      0752 D5      DSKGET: PUSH   D          ; SAVE DE
1456      0753 113108 LXI     D,DSKTYP   ; TYPE TABLE
1457      0756 0600   MVI     B,0        ; EXPAND DISKNO.
1458      0758 EB     XCHG              ;
1459      0759 09     DAD     B          ; HL = DSKTYP + DISKNO
1460      075A 6E     MOV     L,M        ; L = TYP (DISKNO.)
1461      075B 2600   MVI     H,0        ; EXPAND TYPE
1462      075D 19     DAD     D          ; HL = DSKxxx + (DSKTYP + DISKNO.)
1463      075E 4E     MOV     C,M        ; C = RESULT
1464      075F EB     XCHG              ; DSKxxx IN HL AGAIN
1465      0760 D1     POP     D          ; RESTORE DE
1466      0761 C9     RET
1467
1468      PAGE

```

```

1469
1470
1471 ;*****
1472 ;#
1473 ;# FIXINIT INITIALIZES DRIVE WITH FIXED TYPE
1474 ;# STORES TYPE IN DSKTYP TABLE AND EXCHANGES PARAMETERS IN
1475 ;# DPEX AND DPBX TABLES
1476 ;#
1477 ;# IN: C = DISKNO.
1478 ;# A = TYP
1479 ;#
1480 ;# CHANGED REG. A BC DE HL UNCHANGED
1481 ;#
1482 ;*****
1483 ;
1484 0762 210000 FIXINIT:LXI H,HSTDSK
1485 0765 71 MOV M,C ; SAVE DISKNO.
1486 0766 0600 MVI B,0
1487 0768 213108 LXI H,DSKTYP
1488 076E 09 DAD B
1489 076C C38107 JMP EXTINI ; FILL UP TABLES
1490
1491
1492 ;*****
1493 ;#
1494 ;# FDINIT INITIALIZES DRIVE
1495 ;# STORES TYPE IN DSKTYP TABLE AND EXCHANGES PARAMETERS IN
1496 ;# DPEX AND DPBX TABLES
1497 ;#
1498 ;# IN: C = DISKNO.
1499 ;# OUT ZERO = O.K
1500 ;# NOT ZERO = ERROR
1501 ;# CHANGED REG. A BC DE HL UNCHANGED
1502 ;#
1503 ;*****
1504
1505 076F 210000 FDINIT: LXI H,HSTDSK ; SAVE DRIVENO.
1506 0772 71 MOV M,C
1507 0773 0600 MVI B,0 ; EXTEND DRIVENO
1508 0775 213108 LXI H,DSKTYP ; STARTING ADDRESS OF DISK TYPE TABLE
1509 0778 09 DAD B ;
1510 0779 E5 PUSH H ; SAVE ENTRYPOINT IN TABLE
1511 077A C00000 CALL READID ; READ FIRST SECTOR AND MCR FORMAT ID
1512 077D C2AF07 JNZ FIERR
1513 0780 E1 POP H ; RESTORE ENTRYPOINT TO TABLE
1514 0781 E5 EXTINI: PUSH H ; AND SAVE AGAIN
1515 0782 77 MOV M,A ; STORE TYPE IN TABLE
1516 0783 3A0000 LDA HSTDSK
1517 0786 2199FE LXI H,DPB0 ; LOAD STARTING ADDRESS OF FIRST DPB TABLE
1518 0789 110F00 LXI D,DPB1-DPB0 ; LOAD LENGTH OF TABLE
1519 078C 3C INR A
1520 078D 3D MODLOP: DCR A ; COMPUTE STARTING ADDRESS OF DPB TABLE
1521 078E CA9507 JZ MODLOE ; FOR DRIVE DINO
1522 0791 19 DAD D
1523 0792 C38D07 JMP MODLOP
1524 0795 EB MODLOE: XCHG ; DE = STARTING ADDRESS OF DPB TABLE

```

043 (MBIOS)

```

1525 0796 E1          POP     H          ; RESTORE TYPADDRESS
1526 0797 4E          MOV     C,M          ; DISKTYPE IN C
1527 0798 0600        MVI     B,0          ; EXPAND TYPE NO.
1528 079A 21C107      LXI     H,DSKSPT      ; FIRST PARAMETER TO BE MOVED
1529 079D 09          DAD     B            ; TYPEDEPENDANT
1530 079E 010400      LXI     B,DSKBLM-DSKBSH ; WIDTH OF TYPE PARAMETER TABLE
1531 07A1 3E0F        MVI     A,DPB1-DPBO  ; LENGTH OF DPBx TABLES
1532 07A3 F5          M0LOOP: PUSH    PSW      ; SETUP DPBx TABLE SAVE BYTECOUNTER
1533 07A4 7E          MOV     A,M          ; GET VALUE
1534 07A5 12          STAX   D            ; AND STORE IT
1535 07A6 13          INX    D            ; NEXT PARAMETER IN DPBx TABLE
1536 07A7 09          DAD     B            ; NEXT PARAMETER IN TYPE PARAMETER TABLE
1537 07A8 F1          POP     PSW          ; RSTORE COUNTER
1538 07A9 3D          DCR     A            ; DECREMENT LOOP COUNTER
1539 07AA C2A307      JNZ     M0LOOP      ; LOOP UNTIL ALL PARAMETERS TRANSFERED
1540                   ;
1541 07AD AF          XRA     A            ; ALL O.K.
1542 07AE C9          RET
1543
1544                   FIERR:
1545 07AF E1          POP     H          ; RESTORE TYPEPOINTER
1546 07B0 3E80        MVI     A,80H        ; ERROR NO TYPE SELECTED
1547 07B2 77          MOV     M,A
1548 07B3 87          ORA     A            ; SET FLAGS
1549 07B4 C9          RET
1550
1551
1552                   PAGE

```

044 (MBIOS)

```

1553
1554
1555                   TYPDEF 0,1,8,0,2048,128,128,3,80,256,1,0,0 ; NON MCR TYP (DS DD FOR INIT)
1556 0000+=          XLTO   EQU     0
1557                   TYPDEF 1,1,8,0,2048,128,128,3,40,512,0,0,0 ; MCR SS DD 1.4 COMP.
1558 0000+=          XLT1   EQU     0
1559                   TYPDEF 2,1,8,0,2048,128,128,3,80,512,1,0,0 ; MCR DS DD
1560 0000+=          XLT2   EQU     0
1561                   TYPDEF 3,1,17,0,8192,512,0,0,610,512,1,0,0 ; SMB-HARD DISK
1562 0000+=          XLT3   EQU     0
1563                   PAGE

```

```

1564
1565
1566
1567
1568
1569
1570
1571 0785 09080909 dskSID: db SIB0, SIB1, SIB2, SIB3 ;SPECIAL BYTE
1572 0789 01020202 dskslc: db slc0, slc1, slc2, slc3 ;Sektorlänge codiert
1573 078D 01030303 dsksmā: db smā0, smā1, smā2, smā3 ;Sektormask für deblock
1574 07C1 10202044 dsksept: db spl0, spl1, spl2, spl3 ;CPM Sektoren pro Track
1575 07C5 00000000 dsksph: db sph0, sph1, sph2, sph3 ;high byte
1576 07C9 04040406 dskbsn: db bsh0, bsh1, bsh2, bsh3 ;block shift bsh
1577 07CD 0F0F0F3F dskblm: db blm0, blm1, blm2, blm3 ;block mask blm
1578 07D1 01000103 dskext: db exm0, exm1, exm2, exm3 ;extnt mask
1579 07D5 4C499987 dskdsl: db dsl0, dsl1, dsl2, dsl3 ;Disksize low in blocks
1580 07D9 0E000002 dskdsh: db dsh0, dsh1, dsh2, dsh3 ;Disksize High
1581 07DD 7F7F7FFF dskmdl: db mdl0, mdl1, mdl2, mdl3 ;maxdir low
1582 07E1 00000001 dskmdh: db mdh0, mdh1, mdh2, mdh3 ;maxdir high
1583 07E5 C0C0C0C0 dskal0: db al00, al01, al02, al03 ;alloc0
1584 07E9 00000000 dskal1: db al10, al11, al12, al13 ;alloc1
1585 07ED 20202000 dskcsl: db csl0, csl1, csl2, csl3 ;check size low
1586 07F1 00000000 dskcsh: db csh0, csh1, csh2, csh3 ;check size high
1587 07F5 03030300 dskofl: db ofl0, ofl1, ofl2, ofl3 ;track offset low
1588 07F9 00000000 dskofh: db ofh0, ofh1, ofh2, ofh3 ;track offset high
1589 07FD 00000000 dskdbl: db DBL0, DBL1, DBL2, DBL3 ;SECTORTRANSLATE YES/NO
1590 0801 10101040 dskcnt: db bl0, bl1, bl2, bl3 ;Blocksize/128
1591 0805 50285062 DSKTRK: DB TRLO, TRL1, TRL2, TRL3 ;NO. OF TRACKS OF DISC LOW BYTE
1592 0809 00000002 DSKTRH: DB TRHO, TRH1, TRH2, TRH3 ;NO. OF TRACKS HIGH BYTE
1593 080D 08080811 DSKMSC: DB MSC0, MSC1, MSC2, MSC3 ;MAXIMUM SECTORNO.
1594

```

E5DC

046 (MBIOS)

```

1595
1596
1597 0020 =      LEN      EQU      32          ; LENGTH OF XLT TABLE
1598 0002 =      VER      EQU      2          ; VERSION OF DEBLOCKING
1599 0220 =      VERLEN   EQU      (VER*256)+LEN ; IDENTIFICATION WORD
1600
1601 0811        ;
1602            ;
1603 0831 FF     DSKTYP: DB      OFFH        ;DISKTYP TABLE
1604 0832 FF           DB      OFFH        ;TYPE INITIALLY NOT SELECTED (FF)
1605 0833 FF           DB      OFFH
1606 0834 FF           DB      OFFH
1607 0835 FF           DB      OFFH
1608 0836 FF           DB      OFFH
1609 0837 FF           DB      OFFH
1610 0838 FF           DB      OFFH
1611 0839 FF           DB      OFFH
1612 083A FF           DB      OFFH
1613            ;
1614            ;
1615            ;
1616            ;
1617            ;
1618            ;
1619            ;
1620            ;
1621            ;
1622            ;
1623            ;
1624            ;
1625 0838 C00000   DISPER0:      CALL      WRCUPO ;SET CURSOR
1626 083E 7E       DISPER1:MOV      A,M
1627 083F FE24     CPI      '$'
1628 0841 C8       RZ              ; RETURN IF END OF MESSAGE
1629 0842 4F       MOV      C,A
1630 0843 E5       PUSH     H
1631 0844 C00000   CALL      CONOUT ; DISPLAY CHARACTER
1632 0847 E1       POP      M
1633 0848 23       INX      H
1634 0849 C33E08   JMP      DISPER1 ; NEXT CHARACTER
1635
1636            PAGE
    
```

D F C 2

047 (NB10S)

```

1637
1638 ;*****
1639 ;*
1640 ;* OFST ADD [A] TO [HL] RESULT IN [HL]
1641 ;*
1642 ;*****
1643
1644 084C 85 OFST: ADC L
1645 084D 6F MOV L,A
1646 084E 00 RNC
1647 084F 24 INC H
1648 0850 C9 RET
1649
1650
1651 ;*****
1652 ;*
1653 ;* SELTYP RETURNS PARAMETERS FOR XCHG UTILITY
1654 ;*
1655 ;*****
1656
1657 0851 3E04 SELTYP: MVI A,DSKBLM-DSKBSH ; LOAD WIDTH OF TYP-TABLE
1658 0853 012002 LXI B,VERLEN ; LOAD VERSIONNO. AND TABLELENGTH
1659 0856 111108 LXI D,XLT ; LOAD STARTING ADDRESS OF XLT-TABLE
1660 0859 21B507 LXI H,TYP TAB ; LOAD STARTING ADDRESS OF TYP-TABLE
1661 085C C9 RET
1662
1663 ;*****
1664 ;*
1665 ;* CMPEL COMPARE STRING AT [DE] WITH STRING AT [HL]
1666 ;* B BYTES
1667 ;*
1668 ;*****
1669
1670
1671 085D 1A CMPEL: LDAX D ; COMPARES (HL) WITH (DE) B BYTES
1672 085E 8E CMP M ; RETURNS ZERO IF EQUAL OTHERWISE NON ZERO
1673 085F C0 RNZ ; RETURN IF NOT EQUAL
1674 0860 23 INX H
1675 0861 13 INX D
1676 0862 05 DCR B
1677 0863 C8 RZ ; RETURN IF EQUAL
1678 0864 C35D08 JMP CMPEL
1679
1680
1681 PAGE

```

048 (NEIOS)

```

1682
1683
1684 ;*****
1685 ;*
1686 ;* CKSEL CHECK IF DISK SELECTED *
1687 ;*
1688 ;*****
1689
1690 CKSEL:
1691 0867 0600 MVI B,0 ; DISKNO IN C EXPAND TO 16 BIT
1692 0869 213108 LXI H,DSKTYP ; BASEADDRESS OF DISK TYPE TABLE
1693 086C 09 DAD B ; COMPUTE ENTRYPOINT FOR DRIVE
1694 086D 7E MOV A,M ; GET TYPE
1695 086E E680 ANI 80H ; IF BIT 8 IS SET DRIVE IS NOT INITIALIZED
1696 0870 C9 RET
1697 0871 END

```

049 (MBIOS)

ALLOC	061A	1170	1180	1208							
ALV0	FA80	174	175	1037							
ALV1	FAE0	176	177	1041							
ALV2	FB71	178	180	1045							
ALV3	FBC2	180	182	1049							
ALV4	FC13	182	184	1053							
ALV5	FC64	184	186	1057							
ALV6	FC85	186	188	1061							
ALV7	FD06	188	190	1065							
ALV8	FD57	190	192	1069							
ALV9	FDA8	192	194	1073							
BDOS	0000	123	124	125	126						
BDOSNTRY	D006	125	328								
BDOSRA	0059	253	270	273							
BDOSSZ	0E00	121	123								
BDOSTBL	D047	126	247								
BIOS	DE00	120	123	326							
BLKSIZ	0800	1101									
BOOTER	0111	365	378	394	408						
BOOTER1	0114	404	409								
BOOTER2	011F	415	417								
BOOTERM	040E	408	1013								
BS	0008	153									
BTRECS	0000	79	376	383							
CANADA1	03A8	935									
CANADA2	03A9	939									
CBOOT	0058	215	284								
CCP	C800	124	127	338	373						
CCPSTB	C803	127	337								
CCPSZ	0800	122	124								
CHECKOS	0000	78	367								
CHKUMA	050E	1148	1166								
CIMIT	0000	78	287								
CKSEL	0867	495	1387	1690							
CLOSE	06FB	103	1369	1372	1390						
CLRLI1	0000	80	237								
CLRLIN	003C	105	236	419							
CLRSCN	001A	157	420								
CMPRE	0850	102	1179	1251	1671	1678					
COAD	0026	138									
COMPDEHL	01F4	101	695								
CONFIGFL	020F	100	741								
CONFVER	0212	747									
CONIM	0000	77	218	415							
CONOUT	0000	80	219	421	1631						
CONST	0000	77	217								
COTC	0027	139									
COUNTRY	0001	169									
CPMARM	0000	110	324	325	327	329					
CPMSPTD	0020	1105									
CPMVER	0002	111									
CR	0000	151	416	772	777	782	787	792	797	803	807
		812	817	822	837	842	847	853	857	862	
CREAD	0000	79	377	457							
CRTTBL	0316	100	724	888							
CSVD	FAA0	175	176	1037							

	050	(NB10S)										
CSV1	F831	177	172	1041								
CSV2	0000	179	1045									
CSV3	0000	181	1049									
CSV4	0000	183	1053									
CSV5	0000	185	1057									
CSV6	0000	187	1061									
CSV7	0000	189	1065									
CSV8	0000	191	1069									
CSV9	0000	193	1073									
CURCOL	0000	80	412									
CURDISK	0004	150	291	330	505							
CWRITE	0000	80	458									
DANSK	035C	916										
DCOMD	0051	129										
DHOME	0000	80	459	499								
DIRBUF	FA00	173	174	1036	1040	1044	1048	1052	1056	1060	1064	
		1068	1072									
DISFD	0007	147										
DISKIO	0000	84	1435									
DISPER0	0838	235	1624									
DISPER1	083E	1626	1634									
DISPERR	0039	101	103	234	414							
DMAADR	0000	81	564	1304								
DMACOM	0028	140										
DMAMB	002A	143										
DMAMO	0028	142										
DMAREAD	0008	145										
DMASET	0000	79	374	453								
DMAST	0028	141										
DMAWRT	0004	144										
DMBRR	0001	116										
DMBRS	0000	117										
DPB0	FE99	195	196	1036	1517	1518	1531					
DPB1	FEA8	196	197	1040	1518	1531						
DPB2	FEB7	197	198	1044								
DPB3	FEC6	198	199	1048								
DPB4	FED5	199	200	1052								
DPB5	FEE4	200	201	1056								
DPB6	FEF3	201	202	1060								
DPB7	FF02	202	203	1064								
DPB8	FF11	203	204	1068								
DPB9	FF20	204	1072									
DPBASE	0504	485	1030	1033								
DPE0	0504	1034										
DPE1	0514	1038										
DPE2	0524	1042										
DPE3	0534	1046										
DPE4	0544	1050										
DPE5	0554	1054										
DPE6	0564	1058										
DPE7	0574	1062										
DPE8	0584	1066										
DPE9	0594	1070										
DSKALO	07E5	1583										
DSKAL1	07E9	1584										
DSKBLM	07CD	1530	1577	1657								

DSKESH	0709	1530	15764	165					
DSKCMD	0000	22	1415						
DSKCNT	0801	1153	15904						
DSKCSH	07F1	15864							
DSKCSL	07E0	15854							
DSKDBL	07FD	1420	15894						
DSKDSH	0709	15804							
DSKDSL	0705	15794							
DSKERR	0000	76	362						
DSKEXT	0701	15784							
DSKGET	0752	104	1154	1190	1230	1285	1327	1421	14554
DSKMDH	07E1	15824							
DSKMDL	0700	15814							
DSKMSC	0800	105	15934						
DSKOFH	07F9	15884							
DSKOFFL	07F5	15874							
DSKOP1	0745	14344							
DSKOP2	073E	1426	14314						
DSKOP3	073F	1430	14324						
DSKSET	0000	78	353	454					
DSKSID	0785	105	1326	15714					
DSKSLC	0789	105	1229	15724					
DSKSMA	0780	1284	15734						
DSKSPH	0705	15754							
DSKSPT	0701	1189	1528	15744					
DSKTRH	0809	15924							
DSKTRK	0805	105	15914						
DSKTYP	0831	102	1456	1487	1508	16034	1692		
DSTAT	0050	1304							
DTRKSET	0000	84	460						
ENFD	0003	1464							
ERFLAG	0000	84	1226	1349	1358				
ERRORMSG	040E	726	10114						
ESC	0018	1564							
EXTINI	0781	1489	15144						
F1	0215	7714	773						
F10	0271	8164	818						
F11	027A	8214	823						
F12	0286	8264	828						
F13	028A	8314	833						
F14	028E	8364	838						
F15	0296	8414	843						
F16	029F	8464	848						
F17	02A6	8524	854						
F18	02AD	8564	858						
F19	02B4	8614	863						
F2	0210	7764	778						
F20	02B8	8664	868						
F21	02BF	8704							
F3	0225	7814	783						
F4	0220	7864	788						
F5	023E	7914	793						
F6	0246	7964	798						
F7	024E	8024	804						
F8	0256	8064	808						
F9	0267	8114	813						

	052	(NB105)								
FATERRM	0498	103	1019							
FOORA	0051	131								
FOINIT	076F	1393	1505							
FF	000C	154								
FFILLER	0055	872	873							
FIERR	07AF	1512	1544							
FILHST	0669	1245	1266							
FILLER	0005	717	719							
FIXINIT	0762	104	498	1484						
FLUSH	06F0	105	1366	1389						
FNERRM	049F	101	1020							
FNTBLND	0315	100	874							
FORMATT	0090	80	451							
FRANCE	0321	901								
FREAD	0000	85	461							
FUNTBL	0215	100	723	769	872					
FWRITE	0000	85	462							
FWVER	0FF7	118								
GERMANY	0336	906								
GETDPH	016C	480	496	500						
GO1	00A3	335	337							
HOME	01A0	223	518							
HOMED	01AA	521	523							
HRDDSK	0170	476	490							
HSTACT	0000	83	522	1241	1374					
HSTBLKD	0004	1104	1105	1106						
HSTBUF	F600	172	173	1301	1433					
HSTDSK	0000	83	1248	1269	1282	1324	1418	1484	1505	1516
HSTSEC	0000	83	386	1256	1273	1424	1432	1437		
HSTSZD	0200	380	1102	1104						
HSTSPTD	0008	1103	1105							
HSTRK	0000	83	1271							
HSTWRT	0000	83	519	1262	1278	1312	1356	1366	1375	
INITBDOS	003F	246	321							
INPBFF	0002	165								
IOBSET	0208	102	731							
IOBYTE	0003	149								
IOERRM	0473	103	1018							
ITALY	037C	925								
KBDAT	0001	168								
KBELL	0041	163								
KCOUNT	0041	164								
KEYBASE	0040	159	160	161	162	163	164			
KEYTBL	03EE	100	725	967						
KSPAIN	036F	921								
LEN	0020	1597	1599	1601						
LEN1	0008	771	773							
LEN10	0009	816	818							
LEN11	000C	821	823							
LEN12	0004	826	828							
LEN13	0004	831	833							
LEN14	0008	836	838							
LEN15	0009	841	843							
LEN16	0007	846	848							
LEN17	0007	852	854							
LEN18	0007	856	858							

LEN19	0007	861	863+
LEN2	0008	776	778+
LEN20	0004	866	868+
LEN3	0008	781	783+
LEN4	0011	786	788+
LEN5	0008	791	793+
LEN6	0008	796	798+
LEN7	0008	802	804+
LEN8	0011	806	808+
LEN9	000A	811	813+
LF	000A	152+	
LGDAT	0020	167+	
LIMBUF	FDF9	194+	195
LIST	0000	77	220
LISTST	0000	77	230
LOAD1	00E1	377+	388
LOAD2	00FF	385	390+
LOADA	00B2	356+	364
LOADB	00C6	360	366+
LOADSYS	00A9	289	306 351+ 422
LVAR0	0316	892+	894
LVAR1	0319	896+	898
LVAR10	03A8	938+	941
LVAR11	0387	943+	946
LVAR12	03C8	948+	951
LVAR13	03D9	953+	956
LVAR2	0320	900+	903
LVAR3	0335	905+	908
LVAR4	0348	910+	913
LVAR5	0358	915+	918
LVAR6	036E	920+	922
LVAR7	0378	924+	927
LVAR8	0392	929+	932
LVAR9	03A7	934+	936
M1RS232	0210	104	743+
M2RS232	0211	104	745+
MATCH	0686	1258	1280+
MCRTTBL	0204	724+	
MFNTBL	0202	723+	
MKEYTBL	0206	725+	
MMAREA	0200	722+	
MMESS	0208	726+	
MODEFL	020E	333	735+
MODLOE	0795	1521	1524+
MODLOP	0780	1520+	1523
MOLOOP	07A3	1532+	1539
MOTOROFF	0001	132+	
MOTORON	0014	137+	
MOUNT	0706	478	1386+
MOVE	01E4	100	641+
NOVEL	01EA	101	658+
MUL1	01CB	611+	620
MUL2	01D0	612	614+
MUL3	01D9	615	617 619+
MULT	01C2	101	605+
MUSICC	0006	170+	

		054	(NB IOS)																		
NBRFLEX	020A	105	474	730																	
NCRDMV	007D	292	318																		
NCRVER	0004	115																			
NCRVERH	0000	114																			
NOMATCH	0662	1252	1260																		
NOOSDSK	0108	368	402																		
NOOSERM	041E	403	1014																		
NOGVF	0613	1194	1202																		
NOTROYM	0436	103	1015																		
NRDYRX	0447	102	1016																		
NUMHDSK	0214	491	753																		
OFST	084C	100	442	486	1428	1644															
OUTPEFF	0001	166																			
PMAADR	0000	79	379	382	1434																
PORTUG	03C9	949																			
PUNCH	0000	77	221																		
PVRS232	0213	104	751																		
RAMSEL	0010	133																			
RDKEY	0040	161																			
RDTRAK	0000	79	393	452																	
READ	05A4	228	1125																		
READER	0000	77	222																		
READHST	071A	1276	1414																		
READID	0000	82	1511																		
READOP	0000	81	1129	1144	1306																
READSEC1	0000	78	358																		
RELEASE	0004	113																			
RELEASEH	0000	112																			
RESET	0000	78	269	288	305																
RETRYC	020C	102	354	732																	
ROMSEL	0011	134																			
RSFLAG	0000	83	1130	1205	1213	1274															
RSKEY	0041	162																			
RSTBDOS	0052	254	268	450																	
RSTC	0200	102	733																		
RWEND	06EE	1350	1359																		
RWM1	06D7	1331	1343																		
RWM2	06CE	1335	1342																		
RWMOVE	0686	1308	1315																		
RWOPER	0622	1133	1206	1223																	
SAFRICA	03B8	944																			
SAVESEC	0000	84	1425	1436																	
SECMSKD	0003	1106																			
SECSET	0000	79	372	456																	
SECSHFD	0002	1107																			
SECTRAH	0180	231	577																		
SEKDSK	0000	81	477	480	487	494	1151	1157	1177	1227	1249										
		1268	1391																		
SEKHST	0000	81	1238	1255	1272																
SEKSEC	0000	81	550	1232	1286																
SEKTRK	0000	81	536	1270																	
SELCTER	0198	479	493	502																	
SELDISK	0158	224	472																		
SELTYP	0851	233	1657																		
SETDMA	0187	227	561																		
SETSEC	0181	226	547																		

055 (NB10S)

SETTC	0012	1354						
SETTRK	0148	225	5334					
SHIFT	0637	12344	1237					
SHIFTE	0639	1233	12364					
SFAREA	020A	722	7274					
SPECFUN	0134	232	4384					
SPECTBL	0141	441	4494					
STBIOS	0000	102	2154	717				
SUBHLDE	01F0	101	6754					
SWEDEN	0349	9114						
SWISS12	0393	9304						
SYSSTA	0013	1364						
TOF	0000	1554						
TRKSET	0000	79	370	392	455			
TYPTAB	0785	15664	1660					
UK	0316	8974						
UNACNT	0000	82	1127	1156	1168	1174	1211	1373
UNADSK	0000	82	1158	1176	1187			
UNASEC	0000	82						
UNATRK	0000	82	1198	1200				
US	0317	8934						
VAR0L	0003	892	8944					
VAR10L	000F	938	9414					
VAR11L	0011	943	9464					
VAR12L	0011	948	9514					
VAR13L	0015	953	9564					
VAR1L	0007	896	8984					
VAR2L	0015	908	9034					
VAR3L	0013	905	9084					
VAR4L	0013	910	9134					
VAR5L	0013	915	9184					
VAR6L	0000	920	9224					
VAR7L	0017	924	9274					
VAR8L	0015	929	9324					
VAR9L	0001	934	9364					
VER	0002	15984	1599					
VERLEN	0220	15994	1658					
WBOOT	006F	216	3034					
WBOOT	0003	104	2164					
WKEY	0040	1604						
WINIT	0000	78	307					
WRALL	0000	11124						
WRCUPO	0000	77	1625					
WRDIR	0001	11134	1348					
WRITE	0588	229	11414					
WRITEHST	0718	1264	1357	1370	14114			
WRPRCTM	045A	103	10174					
WRTYPE	0000	81	1132	1146	1347			
WRUAL	0002	11144	1131	1147				
XLT	0811	1427	16014	1659				
XLTO	0000	15564						
XLT1	0000	15584						
XLT2	0000	15604						
XLT3	0000	15624						
YUGOSL	030A	9544						

003 (0BIOS)

```

64
65 ;*****
66 ;*****
67 ;*****
68 ;***** EXTERNAL DEFINITIONS *****
69 ;*****
70 ;*****
71 ;*****
72 ;
73 ;
74 ;
75 EXTRN RSTC,RETRYC,FIXINIT,FLUSH,CHPRE,DSKTYP
76 EXTRN MOVE,CLOSE,CURCOL,TMPCOL,DISPERR
77 EXTRN NOTRDYM,WRPRCTM,FATERM,IQERRM,HRDYRX
78 EXTRN CCHIN,CLRLIN,WBOTE,DSKGET,DSKTRK,DSKSID,DSKSLC,DSKMSC
79 EXTRN WINDRE,WINDFM,WINDRW,NORFLEX
80 ;
81
82 PAGE

```

004 (0BIOS)

```

83
84 ;*****
85 ;*****
86 ;*****
87 ;***** ENTRY DEFINITIONS *****
88 ;*****
89 ;*****
90 ;*****
91 ;
92 ;
93 ;
94 PUBLIC PMAADR,ERRBUF,BTRECS,SAVESEC,DTRKSET
95 PUBLIC DMAADR,READOP,WRTYPE,ERFLAG
96 PUBLIC SEKDSK,SEKTRK,SEKSEC,SEKHST
97 PUBLIC UMACHT,UMADSK,UMATRK,UMASEC
98 PUBLIC RSFLAG,HSTACT,HSTDSK,HSTWRT,HSTSEC,HSTTRK,DSKCMD
99 PUBLIC RESET,DSKSET,READSEC1,DSKERR,CHECKOS
100 PUBLIC TRKSET,SECSET,DMASET,CREAD,ROTRAK,FORMAT,CWRITE
101 PUBLIC DHOME,READID,DISKID,COMBLO,ERRBUF
102 PUBLIC FREAD,FWRITE
103 PAGE

```

```

104
105 ;+++++
106 ;+
107 ;+*
108 ;+* PHYSICAL ROUTINES FOR MCR DMS FLEX CONTROLLER
109 ;+*
110 ;+
111 ;+++++
112 DISKCEQU
113 + ;+
114 + ;+
115 + ;* DISK CONTROLLER EQUATES 30.11.82 11:56 PH
116 + ;+
117 + ;+
118 +
119 +
120 0051+= DCMD EQU 00051H ; DISK COMMAND PORT
121 0050+= DSTAT EQU 00050H ; DISK STATUS PORT
122 0051+= FDCRA EQU 00051H ; READ DATA FROM FDC
123 + ;
124 0001+= MOTOROFF EQU 01H ; MOTOR OFF INDICATOR (SYSSTA)
125 + ;
126 + ;
127 0010+= RAMSEL EQU 00010H ; RAM SELECT
128 0011+= ROMSEL EQU 00011H ; ROM SELECT
129 0012+= SETTC EQU 00012H ; SET TC
130 0013+= SYSSTA EQU 00013H ; SYSTEM STATUS PORT
131 0014+= MOTORDN EQU 00014H ; MOTOR ON
132 0026+= COAD EQU 00026H ; FDC DMA CHANNEL
133 0027+= COTC EQU 00027H ; FDC DMA CHANNEL
134 0028+= DMACOM EQU 00028H ; DMA COMMAND
135 0028+= DMAST EQU 00028H ; READ DMA STATUS
136 0028+= DMAMC EQU 00028H
137 002A+= DMAMB EQU 0002AH
138 +
139 0004+= DMAWRT EQU 004H ; DMA WRITE COMMAND
140 0008+= DMAREAD EQU 008H ; DMA READ COMMAND
141 +
142 0003+= ENFD EQU 003H ; ENABLE FDC CHANNEL (CH3)
143 0007+= DISFD EQU 007H ; DISABLE FDC CHANNEL (CH3)
144 +
145 +
146 + PAGE

```

006 (0BIOS)

```

147      +
148      +           ENDM
149      DISKBFF
150      +
151      +           ;*****
152      +           ;*
153      +           ;*           DISK BUFFER MACRO 11.03.83 17.15 PH           *
154      +           ;*
155      +           ;*****
156      +
157      +
158      F600+=      HSTBUF EQU      0F600H           ;LENGTH 400H
159      FA00+=      DIRBUF EQU      PSTBUF+400H      ;LENGTH 128
160      FA80+=      ALV0  EQU      DIRBUF+128        ;LENGTH 32
161      FAA0+=      CSV0  EQU      ALV0+32           ;LENGTH 64
162      FAE0+=      ALV1  EQU      CSV0+64           ;LENGTH 81
163      FB31+=      CSV1  EQU      ALV1+81           ;LENGTH 64
164      FB71+=      ALV2  EQU      CSV1+64           ;LENGTH 81
165      0000+=      CSV2  EQU      0                 ;NOT USED FOR HARD DISK
166      FBC2+=      ALV3  EQU      ALV2+81           ;LENGTH 81
167      0000+=      CSV3  EQU      0                 ;NOT USED FOR HARD DISK
168      FC13+=      ALV4  EQU      ALV3+81           ;LENGTH 81
169      0000+=      CSV4  EQU      0                 ;NOT USED FOR HARD DISK
170      FC64+=      ALV5  EQU      ALV4+81           ;LENGTH 81
171      0000+=      CSV5  EQU      0                 ;NOT USED FOR HARD DISK
172      FCB5+=      ALV6  EQU      ALV5+81           ;LENGTH 81
173      0000+=      CSV6  EQU      0                 ;NOT USED FOR HARD DISK
174      FD06+=      ALV7  EQU      ALV6+81           ;LENGTH 81
175      0000+=      CSV7  EQU      0                 ;NOT USED FOR HARD DISK
176      FD57+=      ALV8  EQU      ALV7+81           ;LENGTH 81
177      0000+=      CSV8  EQU      0                 ;NOT USED FOR HARD DISK
178      FDA8+=      ALV9  EQU      ALV8+81           ;LENGTH 81
179      0000+=      CSV9  EQU      0                 ;NOT USED FOR HARD DISK
180      FDF9+=      LIMBUF EQU      ALV9+81           ;LENGTH 160
181      FE99+=      DPB0  EQU      LIMBUF+160        ;LENGTH 15
182      FEA8+=      DPB1  EQU      DPB0+15           ;LENGTH 15
183      FEB7+=      DPB2  EQU      DPB1+15           ;LENGTH 15
184      FEC6+=      DPB3  EQU      DPB2+15           ;LENGTH 15
185      FED5+=      DPB4  EQU      DPB3+15           ;LENGTH 15
186      FEE4+=      DPB5  EQU      DPB4+15           ;LENGTH 15
187      FEF3+=      DPB6  EQU      DPB5+15           ;LENGTH 15
188      FF02+=      DPB7  EQU      DPB6+15           ;LENGTH 15
189      FF11+=      DPB8  EQU      DPB7+15           ;LENGTH 15
190      FF20+=      DPB9  EQU      DPB8+15           ;LENGTH 15
191      +           ENDM

```

007 (OBIOS)

```
192          CMTLEQU
193          + ;*****
194          + ;#
195          + ;#   CONTROL CHARACTER EQUATES 27/10/82 10:22 WF
196          + ;#
197          + ;*****
198          +
199          +
200 0003+=   IOBYTE EQU   3           ; I/O BYTE
201 0004+=   CURDISK EQU  4           ; CURRENT DISK NUMBER
202 0000+=   CR      EQU   0DH        ; CARRIGE RETURN
203 000A+=   LF      EQU   0AH        ; LINE FEED
204 0008+=   BS      EQU   08H        ; BACK SPACE
205 000C+=   FF      EQU   0CH        ; FORM FEED
206 000C+=   TOF     EQU   0CH        ; TOP OF FORM
207 0018+=   ESC     EQU   1BH        ; ESCAPE
208 001A+=   CLRSCN EQU   01AH       ; CLEAR SCREEN
209          + PAGE
```

```

008 (0B.IOS)

210 +
211 +          ENDM
212 +          ;*****
213 +          ;#          DISK DEBLOCKING MACROS          #
214 +          ;*****
215
216 +          ;          UTILITY MACRO TO COMPUTE SECTOR MASK
217
218 +          SMASK  MACRO  HBLK          ;COMPUTE LOG2(HBLK) RETURN @X
219 +          @Y    SET   HBLK
220 +          @X    SET   0
221 +          REPT   8
222 +          IF     @Y = 1
223 +          EXITM
224 +          ENDF
225 +          ;          @Y IS NOT 1, SHIFT RIGHT ONE POSITION
226 +          @Y    SET   @Y SHR 1
227 +          @X    SET   @X + 1
228 +          ENDM
229 +          ENDM
230
231 +          HOSTDEQU
232 +          ;*****
233 +          ;#          CP/M TO HOST DISK CONSTANTS 27/10/82 10:23 WF          #
234 +          ;#
235 +          ;*****
236 +
237 +          0800+=    BLKSIZ  EQU    2048          ;CP/M ALLOCATION SIZE
238 +
239 +          0200+=    HSTSIZD EQU    512          ;HOST DISK SECTOR SIZE
240 +          0008+=    HSTSPTD EQU    08          ;HOST DISK SECTORS/TRK
241 +          0004+=    HSTBLKD EQU    HSTSIZD/128 ;CP/M SECTS/HOST BUFF
242 +          0020+=    CPMSPTD EQU    HSTBLKD * HSTSPTD ;CP/M SECTORS/TRACK
243 +          0003+=    SECMSKD EQU    HSTBLKD-1   ;SECTOR MASK
244 +          +          SMASK   HSTBLKD   ;COMPUTE SECTOR MASK
245 +          0004+=    @Y     SET   HSTBLKD
246 +          0000+=    @X     SET   0
247 +          +          REPT   8
248 +          +          IF     @Y = 1
249 +          +          EXITM
250 +          +          ENDF
251 +          +          ;          @Y IS NOT 1, SHIFT RIGHT ONE POSITION
252 +          +          @Y    SET   @Y SHR 1
253 +          +          @X    SET   @X + 1
254 +          +          ENDM
255 +          +          IF     @Y = 1
256 +          +          EXITM
257 +          +          ENDF
258 +          +          ;          @Y IS NOT 1, SHIFT RIGHT ONE POSITION
259 +          0002+=    @Y    SET   @Y SHR 1
260 +          0001+=    @X    SET   @X + 1
261 +          +          IF     @Y = 1
262 +          +          EXITM
263 +          +          ENDF
264 +          +          ;          @Y IS NOT 1, SHIFT RIGHT ONE POSITION
265 +          0001+=    @Y    SET   @Y SHR 1

```



```

009 (OBIO5)

266 0002+*  @X   SET   @X + 1
267      +           IF   @Y = 1
268      +           EXITM
269      +           ENDM
270 0002+=  SECSHFD EQU   @X           ;LOG2(HSTBLK)
271      +
272      +           ENDM
273      +           DISKCOEQ
274      +           ;*****
275      +           ;#
276      +           ;#           DISK CONTROL BITS 24.01.83 12:00 PH
277      +           ;#
278      +           ;*****
279      +
280      +
281 0004+=  SIDE1  EQU   04H           ; SIDE SELECT BIT
282 0000+=  UNITLO EQU   00H           ; LO UNIT BIT
283 0002+=  UNITHI EQU   02H           ; HI UNIT BIT
284 0008+=  FDCEOT EQU   08           ; MAX NUMBER OF SECTORS / TRACK
285 0008+=  FDCIN  EQU   08H           ; FDC INT BIT IN MAIN STATUS
286 0002+=  DDIND  EQU   02H           ; DENSITY INDICATOR (2-512 BYTES)
287 0040+=  DDEN   EQU   040H          ; DOUBLE DENSITY BIT
288 0018+=  GPL    EQU   18H           ; GAP LENGTH
289      +
290      +           ENDM
291      +           DISKCKEQ
292      +           ;*****
293      +           ;#
294      +           ;#           DISK CONTROLLER COMMANDS 27/10/82 10:26 WF
295      +           ;*****
296      +
297      +
298 0002+=  READTRK EQU   02H           ; READ A TRACK
299 0005+=  WRITDAT EQU   05H           ; WRITE DATA COMMAND
300 0006+=  READDAT EQU   06H           ; READ DATA COMMAND
301 0007+=  RESTORE EQU   07H           ; RESTORE COMMAND
302 0008+=  FDCSIS EQU   08H           ; SENSE INTERRUPT STATUS
303 000A+=  IDREAD EQU   0AH           ; READ ID
304 000D+=  WRITFMT EQU   0DH           ; FORMAT A TRACK
305 000F+=  SEEKTRK EQU   0FH           ; SEEK A TRACK
306      +
307      +           PAGE

```

```

010 (0E10S)

308 +
309 +          ENDM
310 ;*****
311 ;*
312 ;*   DISKIO       READ / WRITE SECTOR
313 ;*
314 ;*****
315
316 DISKIO:
317 0000 3A0000      LDA   RSTC       ; GET RESTORE COUNT
318 0003 4F          MOV   C,A        ; --> C
319 DI01:
320 0004 3A0000      LDA   RETRYC      ; GET RETRY COUNT
321 0007 47          MOV   B,A        ; --> B
322 DI02:
323 0008 C0D500      CALL  DDIO       ; "READ"
324 0008 C8          RZ              ; RETURN IF NO ERROR OCCURED
325 000C CD9003      CALL  DSKERR      ; DETECT DISK ERRORS
326 000F 05          DCR   B          ; RETRIES - 1
327 0010 C20800      JNZ   DI02       ; RETRY
328
329 0013 00          DCR   C          ; RESTORE - 1
330 0014 CA1D00      JZ    RWERR      ; REPORT ERROR OR OVERRIDE
331 0017 CDC001      CALL  RECAL      ; "RECALIBRATE"
332 001A C30400      JMP   DI01       ; RETRY
333
334
335 RWERR:
336 001D 3EFF        MVI   A,OFFH      ; ERROR CODE
337 001F 329904      STA   ERFLAG
338 0022 B7          ORA   A
339 0023 C9          RET
340
341 PAGE

```

```

342
343 ;*****
344 ;#
345 ;# CWRITE WRITE SECTOR #
346 ;# #
347 ;*****
348
349 CWRITE:
350 0024 AF XRA A ; CODE FOR WRITE SECTOR
351 0025 328004 STA DSKCMD ; SAVE IT
352 0028 C30000 JMP DISKIO ;
353
354 ;*****
355 ;#
356 ;# CREAD READ SECTOR #
357 ;# #
358 ;*****
359
360 0028 3E01 CREAD: MVI A,01H ; CODE FOR READ SECTOR
361 002D 328004 STA DSKCMD ; SAVE COMMAND
362 0030 C30000 JMP DISKIO ;
363
364
365 ;*****
366 ;#
367 ;# FWRITE WRITE SECTOR RETURN ERROR #
368 ;# #
369 ;*****
370
371 0033 AF FWRITE: XRA A
372 0034 328004 STA DSKCMD ; SAVE WRITE COMMAND
373 0037 C30500 JMP DDIO
374
375
376 ;*****
377 ;#
378 ;# FREAD READ SECTOR RETURN ERROR #
379 ;# #
380 ;*****
381
382 003A 3E01 FREAD: MVI A,01H
383 003C 328004 STA DSKCMD ; SAVE READ COMMAND
384 003F C30500 JMP DDIO
385
386 PAGE

```

012 (OBIOS)

```

387
388 ;*****
389 ;*                                     *
390 ;*   FORMATT WRITE OUT A TRACK       *
391 ;*           [D] PATTERN             *
392 ;*                                     *
393 ;*****
394
395
396
397 FORMATT:
398 0042 0E0D      MVI    C,WRITFMT      ; FIRST PUT IN THE WRITE TRACK COMMAND
399 0044 3A6F04    LDA    HSTDSK
400 0047 210000    LXI    H,NBRFLEX
401 004A BE       CMP    M
402 004B D20000    JNC    WINDFM
403 004E D5       PUSH   D              ; SAVE PATTERN
404 004F CD5501    CALL   SETUPH              ; SET UP FUNCTION AND DRIVE #
405 0052 3602     MVI    M,DDIND        ; DENSITY INDICATOR
406 0054 23       INX    H
407 0055 3608     MVI    M,FDCE0T
408 0057 23       INX    H
409 0058 3650     MVI    M,50H          ; GPL = IBM FORMAT
410 005A 23       INX    H
411 005B D1       POP    D
412 005C 72       MOV    M,D          ; PATTERN
413 005D CD2E03    CALL   SEEK1              ; SEEK TRACK
414 0060 0608     MVI    B,DMAREAD
415 0062 211F00    LXI    H,HSTSPTD*4-1      ; DMA LENGTH
416 0065 CD4F02    CALL   DMA1              ; INIT DMA
417 0068 0606     MVI    B,6              ; # OF CONTROLL BYTES
418 006A CDCA02    CALL   FDCWAIT           ; SEND BYTES
419 006D C8       RZ              ;RETURN IF GOOD STATUS
420 006E 010101    LXI    B,0101H          ;NO RETRY
421 0071 CD9003    CALL   DSKERR            ;DETECT DISK ERRORS
422 0074 D5       DCR    B              ;
423 0075 C24200    JNZ    FORMATT          ;JUMP IF RETRY
424 0078 C31D00    JMP    RWERR            ;ELSE OVERRIDE
425
426
427 PAGE

```

```

428
429
430 ;*****
431 ;*                                     *
432 ;*      RDTRAK  READ A TRACK          *
433 ;*                                     *
434 ;*****
435
436
437
438 RDTRAK:
439 007B 3E01      MVI    A,1
440 007D 327204    STA    HSTSEC      ; SET SECTOR = 1
441 0080 0E46      MVI    C,046H      ; FIRST PUT IN THE READ TRACK COMMAND
442 0082 C0F800    CALL   FDCSET      ; SET FUNCTION AND DRIVE
443 0085 0604      MVI    B,DMAWRT
444 0087 21FF0F    LXI    H,HSTSIZE*HSTSPTD-1  ; SIZE OF A TRACK -1 = DMA LENGTH
445 008A CDAF02    CALL   DMA1        ; INITIALIZE DMA
446 008D 0609      MVI    B,9          ; # OF CONTROL BYTES
447 008F C0CA02    CALL   FDCWAIT     ; SEND OUT THE COMMAND AND WAIT FOR COMPLETION
448 0092 C29900    JNZ    CRDTR1     ; JUMP IF ABNORMAL TERMINATION
449 0095 23        INX    H          ; ERROR BUFFER + 1 = ST 1
450 0096 7E        MOV    A,M        ; GET STATUS 1
451 0097 87        ORA    A
452 0098 C8        RZ          ; RETURN IF GOOD STATUS
453
454 CRDTR1:
455 0099 010101    LXI    B,0101H    ; NO RETRY
456 009C C09003    CALL   DSKERR     ; DETECT DISK ERRORS
457 009F 05        DCR    B          ;
458 00A0 C27B00    JNZ    RDTRAK     ; JUMP IF RETRY
459 00A3 C31D00    JMP    RWERR      ; ELSE OVERRIDE
460
461 PAGE

```

014 (0BIOS)

```

462
463 ;*****
464 ;*
465 ;*   DMASET  SET PHYSICAL DMA ADDRESS
466 ;*       [DE] - DMA-ADDRESS
467 ;*
468 ;*****
469
470
471
472 00A6 EB   DMASET: XCHG
473 00A7 229104 SHLD  PMAADR   ; SET DMA ADDRESS
474 00AA C9   RET
475
476
477
478 ;*****
479 ;*
480 ;*   DSKSET  SET PHYSICAL DISK
481 ;*       [E] - DISK NUMBER
482 ;*
483 ;*
484 ;*****
485
486
487
488 00AB D5   DSKSET: PUSH  D       ; SAVE DISKNO.
489 00AC C00000 CALL  FLUSH
490 00AF C00000 CALL  CLOSE
491 00B2 C1   POP    B       ; RESTORE INTO C
492 00B3 79   MOV    A,C
493 00B4 210000 LXI   H,NBRFLEX
494 00B7 BE   CMP    M
495 00B8 D28E00 JNC   DSKSET1
496 00B8 3E02 MVI   A,2       ; FLEX DISK
497 00B0 21   DB     21H
498 00BE 3E03 DSKSET1: MVI  A,3       ; FIX DISK
499 00C0 C00000 CALL  FIXINIT   ; INITIALIZE AS DS DD MCR DISK
500 00C3 C9   RET
501
502
503
504 ;*****
505 ;*
506 ;*   TRKSET  SET PHYSICAL TRACK
507 ;*       [E] - TRACK NUMBER
508 ;*
509 ;*****
510
511
512
513 00C4 68   TRKSET: MOV   L,E
514 00C5 2600 MVI   H,0
515 00C7 227004 SHLD  HSTRK   ; SET TRACK
516 00CA C9   RET
517

```

015 (0BIOS)

518
519

PAGE

```
520
521 ;*****
522 ;*
523 ;* SECSET SET PHYSICAL SECTOR
524 ;* [E] - SECTOR NUMBER
525 ;*
526 ;*****
527
528
529
530 00C8 78 SECSET: MOV A,E
531 00CC 327204 STA HSTSEC ; SET SECTOR
532 00CF C9 RET
533
534 ;*****
535 ;*
536 ;* PSDRK SET PHYSICAL TRACK
537 ;* [DE] - TRACK NUMBER
538 ;*
539 ;*****
540
541 DTRKSET:
542 00D0 EB XCHG
543 00D1 227004 SHLD HSTRK ; SET TRACK
544 00D4 C9 RET
545
546
547 PAGE
```


017 (OBIO5)

```

548
549 ;*****
550 ;*
551 ;* DDIO PHYSICAL READ AND WRITE *
552 ;*
553 ;*****
554
555 0005 C5 DDIO: PUSH B ; SAVE [B]
556 0006 3A8004 LDA DSKCMD ; LOAD COMMAND ( READ OR WRITE )
557 0009 B7 ORA A ;
558 000A C2E200 JNZ DDIOR ; 1 = READ
559 000D 0E05 MVI C,WRITDAT ; WRITE DATA COMMAND BYTE
560 000F C3E400 JMP DDIO1 ;
561 00E2 0E06 DDIOR: MVI C,READDAT ; READ DATA COMMAND BYTE
562 DDIO1:
563 00E4 3A6F04 LDA HSTDSK
564 00E7 210000 LXI H,NBRFLEX
565 00EA BE CMP M
566 00EB D20000 JNC WINDRW
567 00EE CDF800 CALL FDCSET ; SET UP COMMAND STRING AND DMA
568 00F1 0609 MVI B,9 ; B = NUMBER OF BYTES TO SEND TO FDC
569 00F3 CDCA02 CALL FDCWAIT ; SEND COMMAND STRING
570 00F6 C1 POP B ; RESTORE [B]
571 00F7 C9 RET ; RETURN STATUS
572
573
574
575
576 PAGE

```

```

577
578 ;*****
579 ;*
580 ;* FDCSET SETUP CONTROLL STRING FOR FDC
581 ;* SEEK TRACK AND ACTIVATE DMA
582 ;*
583 ;*****
584
585
586 FDCSET:
587 00F8 C5 PUSH B ; SAVE FUNCTION
588 00F9 C0501 CALL SETUPH ; SET FUNCTION AND DRIVE
589 00FC E5 PUSH H ; SAVE POINTER
590 00FD C07601 CALL TRKTRN ; TRANSLATE TRACK NO.
591 0100 E1 POP H ; RESTORE POINTER
592 0101 77 MOV M,A ; SET TRACK
593 0102 23 INX H
594 0103 C06801 CALL FORMSIDE ; RETURNS HEAD IN BIT 3
595 0106 0F RRC
596 0107 0F RRC
597 0108 77 MOV M,A ; SET HEAD
598 0109 23 INX H
599 010A 3A7204 LDA HSTSEC
600 010D E67F ANI 0111111B ; MASK OF SIDE SELECT BIT
601 010F 77 MOV M,A ; SET SECTOR
602 0110 23 INX H
603 0111 3A6F04 LDA HSTDSK ; GET DISKNO.
604 0114 4F MOV C,A ;
605 0115 C5 PUSH B ; SAVE DISKNO.
606 0116 E5 PUSH H ; SAVE TABLE ADDRESS
607 0117 210000 LXI H,DSKSLC ; CODED SECTORLENGTH
608 011A C00000 CALL DSKGET ; FETCH VALUE
609 011D E1 POP H ; RESTORE TABLEADDRESS
610 011E 71 MOV M,C ; AND WRITE TO TABLE
611 011F 23 INX H
612 0120 01 POP B ; RESTORE DISKNO.
613 0121 C5 PUSH B ; SAVE CODED SECTORLENGTH
614 0122 48 MOV C,E ; MOVE DISKNO. TO C
615 0123 E5 PUSH H ; SAVE TABLE ADDRESS
616 0124 210000 LXI H,DSKMSC ; MAXIMUM SECTOR NUMBER
617 0127 C00000 CALL DSKGET ; GET VALUE INTO C
618 012A E1 POP H ; RESTORE TABLE ADDRESS
619 012B 71 MOV M,C ; AND WRITE TO TABLE
620 012C 23 INX H
621 012D 3618 MVI M,GPL ; GPL = IBM FORMAT
622 012F 23 INX H
623 0130 36FF MVI M,OFFH ; DTL
624 0132 C02E03 CALL SEEK1 ; SEEK TRACK
625 0135 01 POP B ; RESTORE CODED SECTORLENGTH
626 0136 C1 POP B ; RESTORE FUNCTION
627 0137 79 MOV A,C
628 0138 FE46 CPI 046H
629 013A C8 RZ ; RETURN IF READ TRACK
630 013B 1F RAR
631 013C 0608 MVI B,DMAREAD ; DMA READ FUNCTION
632 013E 0A4301 JC FDCSET1 ; JUMP IF WRITE FUNCTION

```

019 (0BIOS)

```
633 0141 0604      MVI    B,DMAWRT      ; DMA WRITE FUNCTION (READ)
634                FDCSET1:
635 0143 78        MOV    A,E           ; A=CODED SECTORL. 0=128 BYTES .. 3=1024 BYTES
636 0144 218000    LXI    H,128        ; CALCULATE SECTORLENGTH FROM CODED SL
637 0147 B7        ORA    A           ; SET FLAGS
638 0148 CA5001    JZ     FDCSET3
639                FDCSET2:
640 014B 27        DAD    H           ; DOUBLE HL
641 014C 3D        DCR    A
642 014D C24801    JNZ    FDCSET2      ;
643                FDCSET3:
644 015C 2B        DCX    H           ; DMA LENGTH - 1
645 0151 CDAF02    CALL   DMA1         ; ACTIVATE DMA
646 0154 C9        RET
647                PAGE
```

```

648
649 ;*****
650 ;*
651 ;* SETUPH SET FUNCTION COMBINED WITH DENSITY
652 ;* AND DRIVE # COMBINED WITH HEAD INTO
653 ;* COMMAND BLOCK
654 ;* [C] FUNCTION
655 ;*****
656 ;
657 ;
658 0155 217304 SETUPH: LXI H,COMBLO ; COMMAND BLOCK FOR FDC
659 0158 3E40 MVI A,DDEN
660 015A 81 ORA C
661 015B 77 MOV M,A ; SET FUNCTION
662 015C 23 INX H
663 015D CD6801 CALL FORMSIDE ;COMPUTE SIDE BIT
664 0160 47 MOV B,A
665 0161 3A6F04 LDA HSTDSK
666 0164 80 ORA B
667 0165 77 MOV M,A ; SET DRIVE
668 0166 23 INX H
669 0167 C9 RET
670
671
672
673 ;*****
674 ;*
675 ;* FORMSIDE RETURNS SIDE BIT IN [A]
676 ;*
677 ;*****
678
679 FORMSIDE:
680 0168 E5 PUSH H
681 0169 CD9101 CALL TYPBYT ; GET TYPBYTE TO A
682 016C E601 ANI 01H ; TEST BIT 0
683 016E CA7401 JZ FMSID1 ; DONE IF SINGLE SIDED
684 0171 CD9D01 CALL DSTRKTRN ; RETURN SIDE BIT
685 0174 E1 FMSID1: POP H ; RESTORE [HL]
686 0175 C9 RET ; RETURNS SIDE NUMBER IN BIT 0
687
688 PAGE

```

021 (0BIOS)

```

689
690 ;*****
691 ;*
692 ;*   TRKTRN RETURNS PHYSICAL TRACK NUMBER IN [A]
693 ;*
694 ;*****
695
696
697 TRKTRN:
698 0176 CD9101      CALL   TYPBYT      ; GET TYPBYTE
699 0179 E607        ANI     00001118
700 017B CA8D01     JZ      TRKTE      ; SINGLE SIDED DISK
701 017E E606        ANI     00001108
702 0180 C28D01     JNZ     TRKTE      ; D-SIDE BUT SECTORS FOR ONE TRK ON BOTH SIDES
703 0183 CD8E01     CALL   GETTRK2     ; B = MAX TRACK / 2
704 0186 3A7004     LDA     HSTTRK
705 0189 B8         CMP     B
706 018A D8         RC
707 018B 90         SUB     B
708 018C C9         RET
709 018D 3A7004     TRKTE: LDA     HSTTRK
710 0190 C9         RET
711
712
713 0191 3A6F04     TYPBYT: LDA     HSTDISK
714 0194 4F         MOV     C,A
715 0195 210000     LXI    H,DSKSID
716 0198 CD0000     CALL   DSKGET
717 019B 79         MOV     A,C
718 019C C9         RET
719
720
721 PAGE

```

```

722
723 ;*****
724 ;*
725 ;*   DSTRKTRN   RETURN SIDE BIT IN [A]
726 ;*
727 ;*
728 ;*****
729
730
731 DSTRKTRN:
732 019D CD9101   CALL   TYPBYT   ; GET TYPBYTE
733 01A0 E606     ANI    00001108 ;
734 01A2 C2B401   JNZ    DSTK2    ; SECTORS FOR ONE TRACK ON TWO SIDES
735 01A5 CDBE01   CALL   GETTRK2  ; B = MAX TRACK / 2
736 01A8 3A7004   LDA    HSTRK    ; GET TRACK NUMBER
737 01AB B8       CMP    B
738 01AC D2B101   JNC    DSTK1    ; JUMP IF SIDE 1
739 01AF AF       XRA    A
740 01B0 C9       RET
741 01B1 3E04     DSTK1: MVI   A,SIDE1
742 01B3 C9       RET
743
744 01B4 3A7204   DSTK2: LDA    HSTSEC
745 01B7 E680     ANI    080H
746 01B9 C2B101   JNZ    DSTK1    ; ON SIDE 2
747 01BC AF       XRA    A
748 01B0 C9       RET
749
750
751
752 GETTRK2:      ; RETURNS MAX TRACK / 2
753 01BE 3A6F04   LDA    HSTDISK  ; LOAD DISKNO.
754 01C1 4F       MOV    C,A
755 01C2 210000   LXI   H,DSKTRK ;
756 01C5 C00000   CALL   DSKGET   ; FETCH MAX TRACK
757 01C8 79       MOV    A,C
758 01C9 B7       ORA    A        ; CLEAR CARRY BIT
759 01CA 1F       RAR    ; DIVIDE BY 2
760 01CB 47       MOV    B,A
761 01CC C9       RET
762
763
764
765 PAGE

```

023 (OB IOS)

```

766
767 ;*****
768 ;*
769 ;* RECAL RECALIBRATE
770 ;*
771 ;*****
772
773 RECAL:
774 DHOME:
775 01CD C5 PUSH B ; SAVE (B) TO USE AS LOCAL REGISTER
776 RECALO:
777 01CE CDE601 CALL HOME ; JUST HOME
778 01D1 3A7C04 LDA ERRBUF
779 01D4 E6E8 ANI 0E8H
780 01D6 FE68 CPI 068H
781 01D8 CA0001 JZ RECAL2 ; JUMP IF DISK NOT READY
782 01DB C1 POP B
783 01DC C9 RET ; RETURN
784
785 RECAL2:
786 01DD 010101 LXI B,0101H
787 01E0 CD9003 CALL DSKERR ; DISPLAY ERROR ROUTINE
788 01E3 C3CE01 JMP RECALO ; RETRY
789
790 PAGE

```

024 (OB IOS)

```

791
792 ;*****
793 ;*
794 ;* HOME HOME DRIVE
795 ;*
796 ;*****
797
798 HOME:
799 01E6 3A6F04 LDA HSTDSK
800 01E9 210000 LXI H,NBRFLEX
801 01EC BE CMP M
802 01ED D20000 JNC WINDRE ; GOTO HARD DISK DRIVER
803 01F0 CD6403 HOME1: CALL MOTORCK ; CHECK IF MOTOR ON
804 01F3 3E07 MVI A,RESTORE ; RESTORE FUNCTION
805 01F5 D351 OUT DCOMD
806 01F7 CDF802 CALL FDCRD1
807 01FA 3A6F04 LDA HSTDSK ; OUT DISK (HEAD = 0)
808 01FD D351 OUT DCOMD
809 01FF CD0F03 CALL FDCIN1 ; WAIT ON INTERRUPT (FNC FINISHED)
810 0202 3A7C04 LDA ERRBUF
811 0205 E6E0 ANI 0E0H
812 0207 FE00 CPI 0C0H
813 0209 CAF001 JZ HOME1 ; RETRY - READY LINE CHANGED (MOTOR ON)
814 020C C9 RET ; RETURN
815
816
817
818 PAGE

```


025 (0BI05)

```
219
820 ;*****
821 ;*
822 ;* READID RETURNS ID BYTE IN [A]
823 ;*
824 ;*****
825
826 READID:
827 0200 3A0000 LDA RETRYC ; SET UP RETRY COUNTER
828 0210 47 MOV B,A ; IN (B)
829 RDID1:
830 0211 C5 PUSH B ; SAVE RETRY COUNTER
831 0212 CD2602 CALL READSEC1 ; READ SECTOR1
832 0215 C1 POP B ; RETRY COUNTER
833 0216 CA8A02 JZ CHECKID ; READ WAS OK
834 0219 GE01 MVI C,1 ; NO RECALIBRATE
835 021B CD9003 CALL DSKERR ; DETECT DISK ERRORS
836 021E 05 DCR B
837 021F C21102 JNZ RDID1 ; RETRY
838
839 RDIDXT2:
840 0222 3EFF MVI A,OFFH
841 0224 87 ORA A ; SET NON ZERO FOR ERROR
842 0225 C9 RET
843 PAGE
```

026 (0BIOS)

```

844
845 ;
846 ;*****
847 ;*
848 ;* READSEC1 READ SECTOR 1 (NCR STANDARD DISK) *
849 ;*
850 ;*****
851 ;
852 ;
853 ;
854 READSEC1:
855 0226 CDC01 CALL RECAL ; RESTORE DISK
856 0229 C05802 CALL ID ; READ CODED SECTORLENGTH FROM DISK RETURN IN A
857 022C C0 RNZ ; RET IF ERROR
858 022D 218304 LXI H,MOUNTPB ; PARAMETER BLOCK TO READ SECTOR 1
859 0230 117304 LXI D,COMBLO ; FDC CONTROL BLOCK
860 0233 010900 LXI B,9 ; LENGTH
861 0236 ED DB OEDH ; Z80 LDIR COMMAND (DE) (- (HL)
862 0237 B0 DB OBOH ; HL=HL+1; DE=DE+1; BC=BC-1 REPEAT UNTIL BC=0
863 0238 217804 LXI H,COMBLO+5
864 023B 77 MOV M,A ; STORE CODED SECTORLENGTH
865 023C 217404 LXI H,COMBLO+1
866 023F 3A6F04 LDA HSTDISK
867 0242 B6 ORA M
868 0243 77 MOV M,A ; COMBINE HEAD AND UNIT
869 0244 2100F6 LXI H,HSTBUF
870 0247 229104 SHLD PMAADR ; SET DMA ADDRESS
871 024A 0604 MVI B,DMAWRT
872 024C 21FF01 LXI H,HSTSIZD-1
873 024F C0AF02 CALL DMA1 ; INITIALIZE DMA
874 0252 0609 MVI B,9
875 0254 C0CA02 CALL FDCWAIT ; SEND BYTES TO FDC
876 0257 C9 RET
877 PAGE

```

027 (0B10S)

```

878
879
880 ;*****
881 ;*
882 ;*ID READS ID FIELD FROM DISK RETURNS CODED SECTORLENGTH IN A *
883 ;*
884 ;*****
885
886 0258 3E4A ID: MVI A, IDREAD OR DDEN ; READ ID COMMAND
887 025A D351 OUT DCOMD
888 025C CDF802 CALL FDCRD1
889 025F 3A6F04 LDA HSTDSK
890 0262 D351 OUT DCOMD ; SEND DISKNO. ( HEAD 0 )
891 0264 D813 ID01: IN SYSSTA ; SENSE IF INTERRUPT
892 0266 E608 ANI FDCIN
893 0268 CA6402 JZ ID01
894 026B CDF802 CALL FDCRD1
895 026E 0607 MVI B, 7 ; GET 7 BYTES
896 0270 C04E03 CALL GETBY1
897 0273 218204 LXI H, ERRBUF+6 ;
898 0276 7E MOV A, M ;LOAD CODED SECTORLENGTH
899 0277 C9 RET
900
901
902 PAGE

```

028 (0B10S)

```

903
904 ;*****
905 ;*
906 ;* CHECKOS CHECK IF AN O.S. DISK IS MOUNTED *
907 ;*
908 ;*****
909 CHECKOS:
910 0278 2103F6 LXI H, HSTBUF+3
911 027E 118302 LXI D, OSMSG
912 027E 0607 MVI B, 7
913 0280 C30000 JMP CMPRE ; COMPARE TWO STRINGS
914
915 0283 43504020320SMMSG: DB 'CPM 2.2'
916
917
918
919
920 PAGE

```

029 (0BIOS)

```

921
922 ;*****
923 ;*
924 ;* CHECKID CHECKS IF THE SELECTED DISK IS *
925 ;* AN NCR DISK *
926 ;*
927 ;*****
928 ;
929 ; ZERO SET IF VALID FORMAT
930 ; A = 0 -> TYP 0
931 ; A = 1 -> TYP 1
932 ; A = 2 -> TYP 2
933 ;
934 ;
935 ;
936 ;
937 CHECKID:
938 028A 210AF6 LXI H,HSTBUF+10
939 028D 11AA02 LXI D,CHECKMSG
940 0290 0605 MVI B,5
941 0292 C00000 CALL CMPRE
942 0295 C2A802 JNZ CKID3
943 0298 7E MOV A,M ; GET ID
944 0299 FE31 CPI '1'
945 029B CAA202 JZ CKID2 ; JUMP IS DDSS
946 029E 3D DCR A
947 029F FE32 CPI '2'
948 02A1 C0 RNZ ; ERROR
949 02A2 D630 CKID2: SUI '0' ; CONVERT IO BINARY
950 02A4 47 MOV B,A
951 02A5 AF XRA A
952 02A6 78 MOV A,B
953 02A7 C9 RET
954 02A8 AF CKID3: XRA A ; TYP 0
955 02A9 C9 RET
956 ;
957 ;
958 ;
959 ;
960 ;
961 02AA 4E43522046CHECKMSG: DB 'NCR F' ; FORMAT MESSAGE
962 PAGE

```

```

963
964 ;*****
965 ;#
966 ;# DMA INITIALIZE DMA
967 ;# HL - SECTOR SIZE
968 ;# B - DMA COMMAND
969 ;#
970 ;*****
971
972 DMA1:
973 02AF 3E43 MVI A,43H
974 02B1 80 ORA B
975 02B2 D32B OUT DMAM0 ; OUT MODE
976 02B4 E5 PUSH H
977 02B5 2A9104 LHLD PMAADR ; GET DMA ADDRESS
978 02B8 70 MOV A,L
979 02B9 D326 OUT COAD ; OUT ADDRESS LOW
980 02BB 7C MOV A,H
981 02BC D326 OUT COAD ; OUT ADDRESS HIGH
982 02BE E1 POP H
983 02BF 70 MOV A,L
984 02C0 D327 OUT COTC ; OUT DMA LENGTH LOW
985 02C2 7C MOV A,H
986 02C3 D327 OUT COTC ; OUT DMA LENGTH HIGH
987 02C5 3E03 MVI A,EMFD
988 02C7 D32A OUT DMAMB ; ENABLE FDC CHANNEL
989 02C9 C9 RET
990
991 PAGE

```

031 (08105)

```

992
993 ;*****
994 ;*
995 ;*   FDCWAIT   SEND COMMAND BYTES (LENGTH IN BC)
996 ;*           TO FDC
997 ;*
998 ;*
999 ;*****
1000
1001 FDCWAIT:
1002 02CA CD6403   CALL   MOTORCK   ; CHECK IF MOTOR ON
1003 02CD 217304   LXI    H,COMBLO
1004 02D0 05      DCR    B
1005 02D1 7E      FDCWAIT1:MOV  A,M
1006 02D2 D351   OUT    DCOMD   ; SEND BYTE TO FDC
1007 02D4 CDFF02   CALL   FDCRD1   ; WAIT UNTIL FDC IS READY TO RECEIVE NEXT
1008 02D7 23      INX    H
1009 02D8 05      DCR    B
1010 02D9 C2D102   JNZ    FDCWAIT1 ; NOT ALL SEND
1011 02DC 7E      MOV    A,M
1012 02DD D351   OUT    DCOMD   ; SEND LAST BYTE
1013 02DF CDFF02   CALL   DMA01
1014 02E2 D2F002   JMC    FDCMER   ; JUMP IF DISK NOT READY
1015 02E5 0607   MVI    B,07H
1016 02E7 CD4E03   CALL   GETBY1   ; GET STATUS BYTES
1017 02EA CAEE02   JZ     FDCWXT   ; GOOD STATUS
1018 02ED C9      RET           ; BAD STATUS
1019
1020 FDCWXT:
1021 02EE AF      XRA    A
1022 02EF C9      RET
1023 FDCMER:
1024 02F0 3E80   MVI    A,80H   ;SPEZIAL NOT READY
1025 FDCWE1:
1026 02F2 217C04   LXI    H,ERRBUF ;SAVE SPEZIAL ERROR CODE
1027 02F5 77      MOV    M,A     ;INTO ERROR BUFFER FOR
1028                                ;DISK NOT READY IN EXECUTION
1029 02F6 B7      ORA    A       ; PHASE
1030 02F7 C9      RET
1031 PAGE

```

032 (0BIOS)

```

1032
1033 ;*****
1034 ;*
1035 ;*   FDCRD1  WAIT UNTIL FDC IS READY
1036 ;*           ENTER WITH COMMAND IN [C]
1037 ;*
1038 ;*****
1039
1040 02F8 0850   FDCRD1: IN    DSTAT      ; GET MAIN STATUS
1041 02FA 17    RAL
1042 02FB 02F802 JNC    FDCRD1    ; JUMP IF FDC NOT READY
1043 02FE C9    RET
1044
1045
1046
1047
1048 ;*****
1049 ;*
1050 ;*   DMA01
1051 ;*
1052 ;*****
1053 DMA01:
1054 02FF 0813   IN    SYSSTA    ; GET STATUS
1055 0301 E604   ANI    04H      ; TEST READY LINE
1056 0303 C8    RZ
1057 0304 0850   IN    DSTAT
1058 0306 17    RAL
1059 0307 02FF02 JNC    DMA01    ; JUMP IF NO MASTER REQUEST
1060 030A 3E07   MVI    A,DISFD
1061 030C D32A   OUT    DMAMB
1062 030E C9    RET
1063 PAGE
1063 ; CARRY SET - DISK WAS READY

```


033 (0BIOS)

```

1064
1065 ;
1066 ;*****
1067 ;*
1068 ;* FDCINI1 WAIT UNTIL FDC HAS BEEN INTERRUPTED ;
1069 ;*
1070 ;*****
1071 ;
1072 ;
1073 FDCINI:
1074
1075 030F 0813 IN SYSSTA ;GET STATUS
1076 0311 E608 ANI FDCIN
1077 0313 CA0F03 JZ FDCIN1 ;JUMP IF NO INTERRUPT
1078 0316 CDF802 CALL FDCRD1
1079 ;
1080 ; RESET INTERRUPT
1081 ;
1082 0319 3E08 MVI A,FDCSIS ; SENSE INTERRUPTSTATUS
1083 031B 0351 OUT DCOMD
1084 031D CDF802 CALL FDCRD1 ; WAIT UNTIL FDC READY
1085 0320 0851 IN FDCRA
1086 0322 327C04 STA ERRBUF ; SAVE STO
1087 0325 CDF802 CALL FDCRD1 ; WAIT UNTIL FDC READY
1088 0328 0851 IN FDCRA
1089 032A CDF802 CALL FDCRD1
1090 032D C9 RET
1091 PAGE

```

034 (0BIOS)

```

1092
1093 ;
1094 ;
1095 ;*****
1096 ;*
1097 ;* SEEK1 SEEK AN SPECIFIED TRACK ;
1098 ;*
1099 ;*****
1100 ;
1101 ;
1102 032E CD6403 SEEK1: CALL MOTORCK ; CHECK IF MOTOR IS ON
1103 0331 3E0F MVI A,SEEKTRK
1104 0333 0351 OUT DCOMD ; SEND SEEK COMMAND
1105 0335 CDF802 CALL FDCRD1
1106 0338 3A6F04 LDA HSTDSK
1107 033B 6F MOV L,A ; SAVE DISK ;
1108 033C CD6801 CALL FORMSIDE ; GET HEAD BIT
1109 033F 85 ORA L ; COMBINE HEAD WITH DRIVE ;
1110 0340 0351 OUT DCOMD ; SEND UNIT&HEAD
1111 0342 CDF802 CALL FDCRD1
1112 0345 CD7601 CALL TRKTRN ; GET TRACK ;
1113 0348 0351 OUT DCOMD ; SEND TRACK ;
1114 034A CD0F03 CALL FDCINI
1115 034D C9 RET
1116 PAGE

```

035 (0BIOS)

```

1117
1118 ;*****
1119 ;*
1120 ;* GETBY1 GET SPECIFIED # OF STATUS BYTES FROM FDC *
1121 ;* B - # NUMBER OF STATUS BYTES *
1122 ;*
1123 ;*****
1124 ;
1125 ;
1126 034E 217C04 GETBY1: LXI H,ERRBUF
1127 0351 0B51 GETBY2: IN FDCRA
1128 0353 77 MOV M,A
1129 0354 CDF802 CALL FDCRD1 ;WAIT UNTIL FDC IS READY
1130 0357 23 INX H
1131 0358 05 DCR B
1132 0359 C25103 JNZ GETBY2 ; JUMP IF NOT ALL STATUS BYTES FETCHED
1133 035C 217C04 LXI H,ERRBUF
1134 035F 7E MOV A,M
1135 0360 E6C0 ANI OCOH
1136 0362 B7 ORA A
1137 0363 C9 RET ; RETURN WITH STATUS
1138 PAGE

```

036 (0BIOS)

```

1139
1140 ;*****
1141 ;*
1142 ;* MOTORCK CHECK IF MOTOR ON , OTHERWISE WAIT 1 SEC*
1143 ;*
1144 ;*****
1145
1146
1147 MOTORCK:
1148 0364 DB13 IN SYSSTA ; GET SYSTEM STATUS
1149 0366 E601 ANI MOTOROFF
1150 0368 0314 OUT MOTORON ; SWITCH MOTOR ON
1151 036A C8 RZ ; RETURN IF MOTOR WAS ON
1152 ;
1153 ; WAIT 1 SEC
1154 ;
1155 0368 05 PUSH D
1156 036C C5 PUSH B
1157 036D 160A MVI D,10
1158 036F 010035 TIM1: LXI B,03500H
1159 0372 08 TIM2: DCX B ; WAIT 100 MS
1160 0373 78 MOV A,B
1161 0374 B1 ORA C
1162 0375 C27203 JNZ TIM2
1163 0378 15 DCR D
1164 0379 C24503 JNZ TIM1
1165 037C C1 POP B
1166 037D D1 POP D
1167 037E C9 RET
1168 PAGE

```

037 (08105)

```
1169
1170 ;*****
1171 ;*
1172 ;*   RESET: INITIALIZE PARAMETERS
1173 ;*
1174 ;*****
1175
1176
1177 RESET:
1178 037F 3EFF      MVI   A,OFFH
1179 0381 0604      MVI   B,4
1180 0383 210000    LXI   H,DSKTYP
1181 RESET1:
1182 0386 77        MOV   M,A
1183 0387 23        INX   H
1184 0388 05        DCR   B
1185 0389 C28603    JNZ   RESET1
1186 038C C00000    CALL  CLOSE
1187 038F C9        RET
1188 PAGE
```

```

1189
1190 ;*****
1191 ;*
1192 ;*   DISK ERROR ROUTINE
1193 ;*   RETRY COUNTER IN [BC]
1194 ;*
1195 ;*   EXIT:
1196 ;*           B = 1 C = 1 --> OVERRIDE
1197 ;*           B > 1      --> RETRY
1198 ;*           NO RETURN  --> ABNORMAL TERMINATION
1199 ;*
1200 ;*****
1201 DSKERR:
1202 0390 E5      PUSH  H          ;SAVE REGISTERS
1203 0391 D5      PUSH  D
1204 0392 C5      PUSH  B
1205 0393 217C04 LXI   H,ERRBUF    ;GET STATUS WORD
1206 0396 7E      MOV   A,M          ;
1207 0397 E6C0    ANI   000H        ;TEST FOR SPEZIAL NOT READY
1208 0399 FE80    CPI   80H
1209 039B 010000 LXI   B,NOTRDYM    ; ERROR MESSAGE
1210 039E CAD103 JZ    DISPER        ;JUMP IF DISK NOT READY
1211 03A1 7E      MOV   A,M          ;           0
1212 03A2 E608    ANI   02H
1213 03A4 010000 LXI   B,WRDYRX     ; ERROR MESSAGE
1214 03A7 C2D103 JNZ   DISPER        ;NOT READY
1215 03AA 23      INX   H          ;   STATUS WORD
1216 03AB 7E      MOV   A,M          ;           1
1217 03AC E602    ANI   02H
1218 03AE 010000 LXI   B,WRPRCTM    ; ERROR MESSAGE
1219 03B1 C2D103 JNZ   DISPER        ; WRITE PROTECT
1220
1221 03B4 C1      POP   B          ;RESTORE RETRY COUNTER
1222 03B5 78      MOV   A,B          ;GET LOOP COUNTER
1223 03B6 FE01    CPI   1          ;
1224 03B8 C2C103 JNZ   ERLOOP        ;JUMP IF RETRY
1225 03B8 79      MOV   A,C          ;GET LOOP COUNTER 2
1226 03BC FE01    CPI   1          ;
1227 03BE CAC403 JZ    IOERR         ;I/O ERROR OR FATAL ERROR
1228
1229 03C1 D1      POP   D          ; RESSTORE REGISTERS
1230 03C2 E1      POP   H
1231 03C3 C9      RET
1232
1233 03C4 C5      PUSH  B          ;CORRECT STACK
1234 03C5 7E      MOV   A,M          ;   STATUS WORD
1235 03C6 E695    ANI   95H          ;           1
1236 03C8 010000 LXI   B,FATERRM    ; ERROR MESSAGE
1237 03CB C2D103 JNZ   DISPER        ;FATAL ERROR
1238 03CE 010000 LXI   B,IOERRM     ; ERROR MESSAGE
1239
1240 03D1 60      MOV   H,B
1241 03D2 69      MOV   L,C
1242
1243 03D3 E5      PUSH  H          ;SAVE POINTER
1244 03D4 3A6F04 LDA   HSTDISK      ;GET DRIVE #

```

```

039 (OBIO5)

1245 0307 3C      INR  A      ;CONVERT IT TO ASCII
1246 0308 F640   ORI  40H    ;
1247 030A 77     MOV  M,A    ;SAVE IT INTO ERROR MESSAGE
1248                DISPE2:
1249 0308 E5     PUSH H      ;SAVE CURRENT ERROR MESSAGE
1250 030C 2A0000  LHLD CURCOL ;GET CURSOR
1251 030F 220000  SHLD TMPCOL ;SAVE CURSOR TEMPORARELY
1252 03E2 210C18  LXI  H,01800H ;POSITION CURSOR
1253 03E5 220000  SHLD CURCOL
1254 03E8 E1     POP  H      ;RETRIEVE ERROR MESSAGE
1255 03E9 C00000  CALL DISPERR ;DISPLAY ERROR MESSEGE LINE 25
1256 03EC C00000  CALL CONIN  ;GET CONSOLE CHARACTER
1257 03EF E65F   ANI  05FH   ;CONVERT LOWER CASE TO UPPERCASE LETTER
1258 03F1 F5     PUSH PSW   ;SAVE INPUT CHARACTER
1259 03F2 2A0000  LHLD TMPCOL ;GET OLD CURSOR POSITION
1260 03F5 220000  SHLD CURCOL ;RESTORE IT
1261 03F8 217C04  LXI  H,ERRBUF ;GET STATUS 0 OF ERRBUF
1262 03FB 7E     MOV  A,M    ;TEST FOR SPEZIAL
1263 03FC E6C0   ANI  0C0H   ;NOT READY
1264 03FE FE80   CPI  080H   ;
1265 0400 CA2104  JZ   DISRE1 ;JUMP IF SPEZIAL NOT READY
1266 0403 7E     MOV  A,M    ;GET STATUS 0 OF ERRBUF
1267 0404 E608   ANI  08     ;TEST FOR NOT READY
1268 0406 C21D04  JNZ  DISMRY ;JUMP IF NOT READY
1269 0409 F1     POP  PSW   ;RESTORE INPUT CHARACTER
1270 040A FE4F   CPI  '0'    ;
1271 040C CA5404  JZ   DISOVE ;JUMP IF OVERRIDE
1272                DISRX:
1273 040F FE52   CPI  'R'    ;
1274 0411 CA4004  JZ   DISRET ;JUMP IF RETRY
1275 0414 FE58   CPI  'X'    ;
1276 0416 CA5804  JZ   DISBOT ;JUMP IF ABNORMAL TERMINATION
1277 0419 E1     POP  H      ;RESTORE OLD ERROR MESSAGE
1278 041A C3D303  JMP  DISPE1 ;DISPLAY MESSAGE ONCE MORE
1279                DISMRY:
1280 041D F1     POP  PSW   ;RESTORE INPUT CHARACTER
1281 041E C30F04  JMP  DISRX
1282                DISRE1:
1283 0421 F1     POP  PSW   ;GET INPUT CHARACTER
1284 0422 FE52   CPI  'R'    ;
1285 0424 E1     POP  H      ;RESTORE OLD ERROR MESSAGE
1286 0425 C2D303  JNZ  DISPE1 ;JUMP IF NOT 'R'
1287 0428 C06403  CALL MOTORCK ; CHECK IF MOTOR ON
1288 042B 0813   IN   SYSSTA ;GET STATUS
1289 042D E604   ANI  04H    ;
1290 042F CA0303  JZ   DISPE1 ;JUMP IF DISK NOT READY
1291 0432 3E07   MVI  A,DISFD
1292 0434 D32A   OUT  DMAMB  ; DISABLE FDC CHANNEL
1293                DISRE2:
1294 0436 0851   IN   FDCRA  ;GET BYTE FROM FDC
1295 0438 C0F802  CALL FDCRD1 ;WAIT TIL READY
1296 043B 17     RAL
1297 043C DA3604  JC   DISRE2 ;GET NEXT BYTE
1298 043F E5     PUSH H     ;CORRECT STACK
1299                DISRET:
1300 0440 E1     POP  H     ;

```

040 (08105)

```
1301 0441 3A0000      LDA   RSTC      ;GET OUTER LOOP COUNTER
1302 0444 4F         MOV   C,A
1303 0445 3A0000      LDA   RETRYC    ;GET INNER LOOP COUNTER
1304 0448 47         MOV   B,A
1305                DISEND:
1306 0449 E1         POP   H          ;CLEAR STACK
1307 044A C5         PUSH  B
1308 044B 3E18      MVI   A,24      ;CLEAR ERROR LINE
1309 044D C00000     CALL  CLRLIN
1310 0450 C1         POP   B
1311 0451 D1         POP   D          ;RESTORE REGISTERS
1312 0452 E1         POP   W
1313 0453 C9         RET
1314                DISOVE:      ;OVERRIDE
1315 0454 E1         POP   H          ;CLEAR STACK
1316 0455 010101     LXI   B,0101H   ;NO RETRY
1317 0458 C34904     JMP   DISEND
1318                DISBOT:
1319 045B 3E18      MVI   A,24      ;CLEAR ERROR LINE
1320 045D C00000     CALL  CLRLIN
1321 0460 AF         XRA   A
1322 0461 320400     STA   CURDISK   ;SELECT DRIVE "A"
1323 0464 C30000     JMP   WBOTE     ;ABNORMAL TERMINATION
1324
1325
1326                PAGE
```

041 (0810S)

```

1327
1328 ;*****
1329 ;*****
1330 ;*
1331 ;*          DISK WORKING STORAGE
1332 ;*
1333 ;*****
1334 ;*****
1335 ;*
1336 ;*          BLOCKING DATA AREA
1337 ;*
1338 ;*****
1339
1340
1341 0467      SEKDSK DS      1
1342 0468      SEKTRK DS      2
1343 046A      SEKSEC DS      1
1344
1345 0468      UMAOSK DS      1
1346 046C      UMATRK DS      2
1347 046E      UNASEC DS      1
1348
1349 046F      HSTOSK DS      1
1350 0470      HSTTRK DS      2
1351 0472      HSTSEC DS      1
1352
1353
1354
1355
1356

```

PAGE

042 (OBIO5)

```

1357
1358 ;*****
1359 ;*
1360 ;*          DISK COMMAND AND ERROR BUFFER
1361 ;*
1362 ;*****
1363
1364
1365 0473 COMBLO DS      9          ; FDC OR FIX COMMAND BLOCK
1366 047C ERRBUF DS      7          ; FDC ERROR BUFFER
1367
1368
1369
1370 ;*****
1371 ;*
1372 ;*          PARAMETER BLOCK FOR SECTOR 1 READ
1373 ;*
1374 ;*****
1375
1376 0483 46 MOUNTPB:DB      DDEN OR READDAT ; FUNCTION
1377 0484 00          DB      0          ; HEAD = 0
1378 0485 00          DB      0          ; CYLINDER = 0
1379 0486 00          DB      0          ; HEAD = 0
1380 0487 01          DB      1          ; SECTOR = 1
1381 0488 02          DB      DDIND      ; DENSITY INDICATOR
1382 0489 08          DB      FDCEOT     ; # OF SECTORS / TRACK
1383 048A 18          DB      GPL        ; GAP LENGTH
1384 048B FF          DB      OFFH       ; DTL
1385
1386
1387          PAGE

```

043 (OBIO5)

```

1388
1389
1390 ;*****
1391 ;*
1392 ;*          MISC DISK DATA
1393 ;*
1394 ;*****
1395
1396
1397 048C 00          BTRECS: DB      0          ; # OF O.S. SECTORS TO LOAD
1398 048D          DSKCMD DS      1
1399 048E          UNACNT DS      1
1400 048F          DMAADR DS      2
1401 0491          PMAADR DS      2
1402 0493          RSFLAG DS      1
1403 0494          READOP DS      1
1404 0495          WRTYPE DS      1
1405 0496          HSTACT DS      1
1406 0497          HSTWRT DS      1
1407 0498          SEKHST DS      1
1408 0499          ERFLAG DS      1          ;ERROR FLAG
1409 049A          SAVESEC DS      1          ;SAVE FOR OLD HSTSEC AFTER SECTORTRANSLATE
1410
1411 049B          END

```

044 (OBIOS)

ALV0	FA80	160	161																
ALV1	FAE0	162	163																
ALV2	FB71	164	166																
ALV3	FBC2	166	168																
ALV4	FC13	168	170																
ALV5	FC64	170	172																
ALV6	FC85	172	174																
ALV7	FD06	174	176																
ALV8	FD57	176	178																
ALV9	FDAB	178	180																
BLKSIZ	0800	237																	
BS	0008	204																	
BTRECS	048C	94	1397																
CHECKID	028A	833	937																
CHECKMSG	02AA	939	961																
CHECKOS	0278	99	909																
CKID2	02A2	945	949																
CKID3	02A8	942	954																
CLOSE	0C00	76	490	1186															
CLRLIN	0000	78	1309	1320															
CLRSCN	001A	208																	
CMPRE	0000	75	913	941															
COAD	0026	132	979	981															
COMBLO	0473	101	658	859	863	865	1003	1365											
COMIN	0000	78	1256																
COTC	0027	133	984	986															
CPMSPTD	0020	242																	
CR	0000	202																	
CRDTR1	0099	448	453																
CREAD	0028	100	360																
CSV0	FAA0	161	162																
CSV1	FB31	163	164																
CSV2	0000	165																	
CSV3	0000	167																	
CSV4	0000	169																	
CSV5	0000	171																	
CSV6	0000	173																	
CSV7	0000	175																	
CSV8	0000	177																	
CSV9	0000	179																	
CURCOL	0000	76	1250	1253	1260														
CURDISK	0004	201	1322																
CWRITE	0024	100	349																
DCOMD	0051	120	805	808	887	890	1006	1012	1083	1104	1110								
		1113																	
DDEN	0040	287	659	886	1376														
DDIND	0002	286	405	1381															
DDIO	0005	323	373	384	555														
DDIO1	00E4	560	562																
DDIOR	00E2	558	561																
DHOME	01CD	101	774																
DIO1	0004	319	332																
DIO2	0008	322	327																
DIRBUF	FA00	159	160																
DISBOT	0458	1276	1318																
DISEND	0449	1305	1317																

045 (0810S)

DISFD	0007	143	1060	1291							
DISKID	0000	101	316	352	362						
DISMRY	041D	1268	1279								
DISOVE	0454	1271	1314								
DISPE1	0303	1242	1278	1286	1290						
DISPE2	0308	1248									
DISPER	0301	1210	1214	1219	1237	1239					
DISPERR	0000	76	1255								
DISRE1	0421	1265	1282								
DISRE2	0436	1293	1297								
DISRET	0440	1274	1299								
DISRX	040F	1272	1281								
DMA01	02FF	1013	1053	1059							
DMA1	02AF	416	445	645	873	972					
DMAADR	048F	95	1400								
DMACOM	0028	134									
DMAMB	002A	137	988	1061	1292						
DMANO	002B	136	975								
DMAREAD	0008	140	414	631							
DMASET	00A6	100	472								
DMAST	0028	135									
DMAWRT	0004	139	443	633	871						
DPB0	FE99	181	182								
DPB1	FEA8	182	183								
DPB2	FEB7	183	184								
DPB3	FEC6	184	185								
DPB4	FED5	185	186								
DPB5	FEE4	186	187								
DPB6	FEF3	187	188								
DPB7	FF02	188	189								
DPB8	FF11	189	190								
DPB9	FF20	190									
DSKCMD	048D	98	351	361	372	383	556	1398			
DSKERR	0390	99	325	421	455	787	835	1201			
DSKGET	0000	78	608	617	716	756					
DSKMSC	0000	78	616								
DSKSET	00AB	99	488								
DSKSET1	00BE	495	498								
DSKSID	0000	78	715								
DSKSLC	0000	78	607								
DSKTRK	0000	78	755								
DSKTYP	0000	75	1180								
DSTAT	0050	121	1040	1057							
DSTK1	0181	738	741	746							
DSTK2	0184	734	744								
DSTRKTRN	019D	684	731								
DTRKSET	0000	94	541								
ENFD	0003	142	987								
ERFLAG	0499	95	337	1408							
ERLOOP	03C1	1224	1228								
ERRBUF	047C	94	101	778	810	897	1026	1086	1126	1133	1205
		1261	1366								
ESC	0018	207									
FATERRM	0000	77	1236								
FDCEOT	0008	284	407	1382							
FDCIN	0008	285	892	1076							

	046	(0B10S)										
FDCIN1	030F	809	1073	1077	1114							
FDCRA	0051	122	1085	1088	1127	1294						
FDCRD1	02F8	806	888	894	1007	1040	1042	1078	1084	1087	1089	
		1105	1111	1129	1295							
FDCSET	00F8	442	567	586								
FDCSET1	0143	632	634									
FDCSET2	0148	639	642									
FDCSET3	0150	638	643									
FDCSIS	0008	302	1082									
FDCWAIT	02CA	418	447	569	875	1001						
FDCWAIT1	02D1	1095	1010									
FDCWE1	02F2	1025										
FDCWER	02F0	1014	1023									
FDCWXT	02EE	1017	1020									
FF	000C	205										
FIXINIT	0000	75	499									
FLUSH	0000	75	489									
FMSID1	0174	683	685									
FORMAT	0042	100	397	423								
FORMSIDE	0168	594	663	679	1108							
FREAD	003A	102	382									
FWRITE	0033	102	371									
GETBY1	034E	896	1016	1126								
GETBY2	0351	1127	1132									
GETTRK2	018E	703	735	752								
GPL	0018	288	621	1383								
HOME	01E6	777	798									
HOME1	01F0	803	813									
HSTACT	0496	98	1405									
HSTBLKD	0004	241	242	243	244	245						
HSTBUF	F600	158	159	869	910	938						
HSTDSK	046F	98	399	563	603	665	713	753	799	807	866	
		889	1106	1244	1349							
HSTSEC	0472	98	440	531	599	744	1351					
HSTSIZE	0200	239	241	444	872							
HSTSPTD	0008	240	242	415	444							
HSTRK	0470	98	515	543	704	709	736	1350				
HSTWRT	0497	98	1406									
ID	0258	856	886									
ID01	0264	891	893									
IDREAD	000A	303	886									
IOBYTE	0003	200										
IOERR	03C4	1227	1232									
IOERRM	0000	77	1238									
LF	000A	203										
LINBUF	FDF9	180	181									
MOTORCK	0364	803	1002	1102	1147	1287						
MOTOROFF	0001	124	1149									
MOTORON	0014	131	1150									
MOUNTPB	0483	858	1376									
MOVE	0000	76										
NBRFLEX	0000	79	400	493	564	800						
NOTRODM	0000	77	1209									
NRDYRX	0000	77	1213									
OSMSG	0283	911	915									
PMAADR	0491	94	473	870	977	1401						

047 (0810S)

RANSEL	0010	127					
RDID1	0211	829	837				
RDIDXT2	0222	839					
RDTRAK	0078	100	438	457			
READDAT	0006	300	561	1376			
READID	0200	101	826				
READOP	0494	95	1403				
READSEC1	0226	99	831	854			
READTRK	0002	298					
RECAL	0100	331	773	855			
RECALD	010E	776	788				
RECAL2	0100	781	785				
RESET	037F	99	1177				
RESET1	0386	1181	1185				
RESTORE	0007	301	804				
RETRYC	0000	75	320	827	1303		
ROMSEL	0011	128					
RSFLAG	0493	98	1402				
RSTC	0000	75	317	1301			
RWERR	0010	330	335	424	458		
SAVESEC	049A	94	1409				
SECMASK	0003	243					
SECSET	0008	100	530				
SECSHFD	0002	270					
SEEK1	032E	413	624	1102			
SEEKTRK	000F	305	1103				
SEKDSK	0467	96	1341				
SEKHST	0498	96	1407				
SEKSEC	046A	96	1343				
SEKTRK	0468	96	1342				
SETTC	0012	129					
SETUPH	0155	404	588	658			
SIDE1	0004	281	741				
SYSSTA	0013	130	891	1054	1075	1148	1288
TIM1	036F	1158	1164				
TIM2	0372	1159	1162				
TMPCOL	0000	76	1251	1259			
TOF	000C	206					
TRKSET	0004	100	513				
TRKTE	0180	700	702	709			
TRKTRN	0176	590	697	1112			
TYPBYT	0191	681	698	713	732		
UNACHT	048E	97	1399				
UNADSK	0468	97	1345				
UNASEC	046E	97	1347				
UNATRK	046C	97	1346				
UNITHI	0002	283					
UNITLO	0000	282					
WBOTE	0000	78	1323				
WINDFM	0000	79	402				
WINDRE	0000	79	802				
WINDRW	0000	79	566				
WRITDAT	0005	299	559				
WRITFMT	0000	304	398				
WRPCTM	0000	77	1218				
WRTYPE	0495	95	1404				

1

003 (PBIOS)

```

69
70 ;*****
71 ;*****
72 ;*****
73 ;***** EXTERNAL DEFINITIONS *****
74 ;*****
75 ;*****
76 ;*****
77 ;
78 ;
79 ;
80 EXTRN MULT,COMPDEHL,FUNCCH,CRTTBL,SIOINIT,PIOINIT
81 EXTRN IOBSET,SRLINST,KEYST,SRLIN,KEYIN,SRLCOUT
82 EXTRN PRTOUT,SRLOUT,SRLSTAT,PRTSTAT,SUBHLDE
83 EXTRN KBDINIT,LANGUAGE,GETLPOS,SACTIVE,CLRLIN
84 EXTRN DESLCT,TRANS
85
86
87
88
89
90 PAGE
    
```

004 (PBIOS)

```

91
92 ;*****
93 ;*****
94 ;*****
95 ;***** ENTRY DEFINITIONS *****
96 ;*****
97 ;*****
98 ;*****
99 ;
100 ;
101 ;
102 PUBLIC DRQ,DRQADR,CURCOL,TRPCOL,TRNCHR,TRNSPC,POPREG,CLRLI1,ROUTE
103 PUBLIC CONST,COMIN,LIST,PUNCH,READER,LISTST,CIMIT,WINIT
104 PUBLIC COMOUT,MODCRT,WRCUPO,CLRLI1
105
106 PAGE
    
```

```

107
108 ;MACRO CALLS
109 KEYEQU
110 + ;*****
111 + ;*
112 + ;* KEYBOARD EQUATES 27/10/82 10:27 WF *
113 + ;*
114 + ;*****
115 0040+= KEYBASE EQU 00040H ; BASE ADDRESS OF KEYBOARD
116 +
117 0040+= WKEY EQU KEYBASE ; WRITE DATA TO KEYBOARD
118 0040+= RKEY EQU KEYBASE ; READ DATA FROM KEYBOARD
119 0041+= RSKY EQU KEYBASE+1 ; READ KEYBOARD STATUS
120 0041+= KBELL EQU KEYBASE+1 ; RING BELL
121 0041+= KCOUNT EQU KEYBASE+1 ; COUNTRY CODE PORT
122 +
123 0002+= IMPBFF EQU 02H ; INPUTBUFFER FULL
124 0001+= OUTPBFF EQU 01H ; OUTPUTBUFFER FULL
125 0080+= LGDAT EQU 80H ; LANGUAGE CODE READY FLAG
126 0001+= KBDAT EQU 01H ; DATA READY BIT
127 0001+= COUNTRY EQU 01H ; COMMAND TO GET COUNTRY CODE
128 0006+= MUSICC EQU 06H ; MUSIC COMMAND
129 + ;
130 + ENDM
131 VERSEQU
132 + ;*****
133 + ;*
134 + ;* SWITCHES 07.06.83 10:00 PH *
135 + ;*
136 + ;*****
137 +
138 0000+= CPMRAM EQU 0 ; CP/M RAM BASE
139 0002+= CPMVER EQU 2 ; CP/M VERSION
140 0000+= RELEASEH EQU 0 ; MCR RELEASE HIGH BYTE
141 0004+= RELEASE EQU 4 ; MCR RELEASE LOW BYTE
142 0000+= MCRVERH EQU 0 ; MCR VERSION HIGH BYTE
143 0004+= MCRVER EQU 4 ; MCR VERSION LOW BYTE
144 0001+= DMBRR EQU 1 ; D-NUMBER RELEASE
145 0000+= DMBRS EQU 0 ; D-NUMBER SUBISSUE
146 0FF7+= FWVER EQU 0FF7H ; LOCATION OF FWVERSION IN FIRMWARE
147 + PAGE

```

006 (PBIOS)

```

148 +
149 +          ENDM
150 CMTLEQU
151 + ;*****
152 + ;*
153 + ;*   CONTROL CHARACTER EQUATES 27/10/82 10:22 WF
154 + ;*
155 + ;*****
156 +
157 +
158 0003+=   IOBYTE EQU   3           ; I/O BYTE
159 0004+=   CURDISK EQU  4           ; CURRENT DISK NUMBER
160 000D+=   CR      EQU   0DH        ; CARRIGE RETURN
161 000A+=   LF      EQU   0AH        ; LINE FEED
162 0008+=   BS      EQU   08H        ; BACK SPACE
163 000C+=   FF      EQU   0CH        ; FORM FEED
164 000C+=   TOF     EQU   0CH        ; TOP OF FORM
165 001B+=   ESC     EQU   1BH        ; ESCAPE
166 001A+=   CLRSCN EQU   01AH       ; CLEAR SCREEN
167 +          PAGE
    
```


007 (PBIOS)

```

168 +
169 +          ENDM
170 DISKBFF
171 +
172 + ;*****
173 + ;#
174 + ;#          DISK BUFFER MACRO 11.03.83 17.15 PH
175 + ;#
176 + ;*****
177 +
178 +
179 F600+= HSTBUF EQU 0F600H          ;LENGTH 400H
180 FA00+= DIRBUF EQU HSTBUF+400H    ;LENGTH 128
181 FA80+= ALV0 EQU DIRBUF+128       ;LENGTH 32
182 FAA0+= CSV0 EQU ALV0+32          ;LENGTH 64
183 FAE0+= ALV1 EQU CSV0+64          ;LENGTH 81
184 FB31+= CSV1 EQU ALV1+81          ;LENGTH 64
185 FB71+= ALV2 EQU CSV1+64          ;LENGTH 81
186 0000+= CSV2 EQU 0                ;NOT USED FOR HARD DISK
187 F8C2+= ALV3 EQU ALV2+81          ;LENGTH 81
188 0000+= CSV3 EQU 0                ;NOT USED FOR HARD DISK
189 FC13+= ALV4 EQU ALV3+81          ;LENGTH 81
190 0000+= CSV4 EQU 0                ;NOT USED FOR HARD DISK
191 FC64+= ALV5 EQU ALV4+81          ;LENGTH 81
192 0000+= CSV5 EQU 0                ;NOT USED FOR HARD DISK
193 FC85+= ALV6 EQU ALV5+81          ;LENGTH 81
194 0000+= CSV6 EQU 0                ;NOT USED FOR HARD DISK
195 FD06+= ALV7 EQU ALV6+81          ;LENGTH 81
196 0000+= CSV7 EQU 0                ;NOT USED FOR HARD DISK
197 FD57+= ALV8 EQU ALV7+81          ;LENGTH 81
198 0000+= CSV8 EQU 0                ;NOT USED FOR HARD DISK
199 FDA8+= ALV9 EQU ALV8+81          ;LENGTH 81
200 0000+= CSV9 EQU 0                ;NOT USED FOR HARD DISK
201 FDF9+= LIMBUF EQU ALV9+81        ;LENGTH 160
202 FE99+= DPB0 EQU LIMBUF+160       ;LENGTH 15
203 FEA8+= DPB1 EQU DPB0+15          ;LENGTH 15
204 FEB7+= DPB2 EQU DPB1+15          ;LENGTH 15
205 FEC6+= DPB3 EQU DPB2+15          ;LENGTH 15
206 FED5+= DPB4 EQU DPB3+15          ;LENGTH 15
207 FEE4+= DPB5 EQU DPB4+15          ;LENGTH 15
208 FEF3+= DPB6 EQU DPB5+15          ;LENGTH 15
209 FF02+= DPB7 EQU DPB6+15          ;LENGTH 15
210 FF11+= DPB8 EQU DPB7+15          ;LENGTH 15
211 FF20+= DPB9 EQU DPB8+15          ;LENGTH 15
212 +          ENDM
213 DISKCEQU
214 + ;*****
215 + ;#
216 + ;#          DISK CONTROLLER EQUATES 30.11.82 11:56 PH
217 + ;#
218 + ;*****
219 +
220 +
221 0051+= DC0MD EQU 00051H          ; DISK COMMAND PORT
222 0050+= DSTAT EQU 00050H          ; DISK STATUS PORT
223 0051+= FDCRA EQU 00051H          ; READ DATA FROM FDC

```

008 (PB10S)

```
224      +      ;
225 0001+=    MOTOROFF EQU 01H ; MOTOR OFF INDICATOR (SYSSTA)
226      +      ;
227      +      ;
228 0010+=    RAMSEL EQU 00010H ; RAM SELECT
229 0011+=    ROMSEL EQU 00011H ; ROM SELECT
230 0012+=    SETTC EQU 00012H ; SET TC
231 0013+=    SYSSTA EQU 00013H ; SYSTEM STATUS PORT
232 0014+=    MOTORON EQU 00014H ; MOTOR ON
233 0026+=    COAD EQU 00026H ; FDC DMA CHANNEL
234 0027+=    COTC EQU 00027H ; FDC DMA CHANNEL
235 0028+=    DMACOM EQU 00028H ; DMA COMMAND
236 0028+=    DMAST EQU 00028H ; READ DMA STATUS
237 0028+=    DMAMO EQU 00028H
238 002A+=    DMAMB EQU 0002AH
239      +
240 0004+=    DMAWRT EQU 004H ; DMA WRITE COMMAND
241 0008+=    DMAREAD EQU 008H ; DMA READ COMMAND
242      +
243 0003+=    ENFD EQU 003H ; ENABLE FDC CHANNEL (CH3)
244 0007+=    DISFD EQU 007H ; DISABLE FDC CHANNEL (CH3)
245      +
246      +
247      +      PAGE
```

```

248      +
249      +          ENDM
250      ;*****
251      ;*****
252      ;*****
253      ;*****      BIOS DEVICE RQTINES      *****
254      ;*****
255      ;*****
256      ;*****
257      ;*****
258      ;*
259      ;#      CINIT SET I/O BYTE
260      ;#          GET COUNTRY CODE FROM KDB
261      ;#          INIT PRINTER
262      ;*
263      ;*****
264
265
266
267      CINIT:
268      0000 3A0000      LDA      IOBSET
269      0003 320300      STA      IOBYTE      ; SET INTEL I/O BYTE (ADDR 3)
270      0006 C00000      CALL     KBDINIT      ; GET COUNTRY CODE OF KBD
271      0009 C00000      CALL     DESLCT      ; DESELECT DRIVE 1
272      ;
273      ;          GET MONO / COLOR INDICATOR FROM FIRMWARE
274      ;
275      EB39 → 000C D311      OUT      ROMSEL      ; ENABLE FIRMWARE
276      000E 3AF90F      LDA      FWVER+2      ; GET COLOR INDICATOR
277      0011 325407      STA      COLORI      ; SAVE IT
278      0014 D310      OUT      RAMSEL      ; ENABLE RAM
279      WINIT:
280      0016 210000      LXI      H,0
281      0019 220000      SHLD    SACTIVE      ; RESET PRINTER ACTIVE FLAG
282      001C 3A0300      LDA      IOBYTE      ; INIT PRINTER
283      001F 07          RLC
284      0020 07          RLC
285      0021 C09800      CALL     ROUTE
286      0024 0000      DW      SIOINIT
287      0026 0000      DW      SIOINIT
288      0028 0000      DW      PIOINIT
289      002A 0000      DW      SIOINIT
290
291
292
293
294      PAGE

```

010 (PBIOS)

```

295
296
297 ;*****
298 ;*
299 ;* CONST GET CONSOLE STATUS
300 ;*
301 ;*****
302
303
304 CONST:
305 002C 3A0300 LDA IOBYTE
306 002F C09800 CALL ROUTE
307 0032 0000 DW SRLINST ; TTY:
308 0034 0000 DW KEYST ; CRT:
309 0036 0000 DW KEYST ; CRT:
310 0038 0000 DW SRLINST ; TTY:
311
312
313 ;*****
314 ;*
315 ;* COMIN GET CONSOLE CHARACTER
316 ;*
317 ;*****
318
319
320
321 COMIN:
322 003A 3A0300 LDA IOBYTE
323 003D C09800 CALL ROUTE
324 0040 0000 DW SRLIN ; TTY:
325 0042 0000 DW KEYIN ; CRT:
326 0044 0000 DW KEYIN ; CRT:
327 0046 0000 DW SRLIN ; TTY:
328
329
330 ;*****
331 ;*
332 ;* CONOUT OUTPUT CHARACTER TO CONSOLE
333 ;*
334 ;*****
335
336
337
338 CONOUT:
339 0048 3A0300 LDA IOBYTE
340 004B C09800 CALL ROUTE
341 004E 0000 DW SRLCOUT ; TTY:
342 0050 A700 DW CRTOUT ; CRT:
343 0052 A700 DW CRTOUT ; CRT:
344 0054 0000 DW SRLCOUT ; TTY:
345
346 PAGE
    
```

347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390

```
*****  
;*  
;* PUNCH OUTPUT CHARACTER TO THE PUNCH DEVICE *  
;*  
*****
```

```
PUNCH: LDA I0BYTE  
RRC  
RRC  
RRC  
RRC  
CALL ROUTE  
DW SRLCOUT ; TTY:  
DW CRTOUT ; CRT:  
DW PRTOUT ; LPT:  
DW CRTOUT ; CRT:
```

```
*****  
;*  
;* READER GET CHARACTER FROM READER DEVICE *  
;*  
*****
```

```
READER:  
LDA I0BYTE  
RRC  
RRC  
CALL ROUTE  
DW SRLIN ; TTY:  
DW KEYIN ; CRT:  
DW KEYIN ; CRT:  
DW SRLIN ; TTY:
```

PAGE

012 (PBIOS)

```

391
392
393
394
395 ;*****
396 ;#
397 ;# LIST OUTPUT CHARACTER TO LIST DEVICE
398 ;#
399 ;*****
400
401
402
403 LIST:
404
405 0078 3A0300 LDA IOBYTE
406 007B 07 RLC
407 007C 07 RLC
408 007D CD9800 CALL ROUTE
409 0080 0000 DW SRLOUT ; TTY:
410 0082 A700 DW CRTOUT ; CRT:
411 0084 0000 DW PRTOUT ; LPT:
412 0086 0000 DW SRLOUT ; TTY:
413
414
415
416 ;*****
417 ;#
418 ;# LISTST GET STATUS FROM LIST DEVICE
419 ;#
420 ;*****
421
422
423
424 LISTST:
425 0088 3A0300 LDA IOBYTE
426 008B 07 RLC
427 008C 07 RLC
428 008D CD9800 CALL ROUTE
429 0090 0000 DW SRLSTAT ; TTY:
430 0092 0000 DW KEYST ; CRT:
431 0094 0000 DW PRTSTAT ; LPT:
432 0096 0000 DW SRLSTAT ; TTY:
433
434 PAGE

```

```

435
436
437 ;*****
438 ;*                                     *
439 ;*   ROUTE   TRANSFER TO SELECTED PHYSICAL DEVICE   *
440 ;*                                     *
441 ;*                                     *
442 ;*****
443
444
445 ROUTE:
446 0098 E603      ANI    3           ; MASK BIT 0 AND 1
447 009A 87       ADD    A           ; 2 BYTE OFFSET
448 0098 E1       POP    H           ; HL - START ADDRESS OF TABLE
449 009C 85       ADD    L
450 009D 6F       MOV    L,A
451 009E D2A200   JNC    ROUT1
452 00A1 24       INR    H
453 ROUT1:
454 00A2 7E       MOV    A,M         ;HL - ENTRY ADDRESS IN TABLE
455 00A3 23       INX    H
456 00A4 66       MOV    H,M
457 00A5 6F       MOV    L,A
458 00A6 E9       PCHL              ; TRANSFER TO SELECTED PHYSICAL DRIVER
459
460 PAGE

```

014 (PBIOS)

```

461
462
463
464 ;*****
465 ;*****
466 ;*****
467 ;***** CRT DRIVER *****
468 ;*****
469 ;*****
470 ;*****
471 ;*****
472 ;*
473 ;* CRT EQUATES *
474 ;*
475 ;*****
476
477
478 GDC EQU
479 + ; 'GDC-EQUATES' 27/10/82 10:00 WF
480 + ;
481 + ;
482 + ; READ
483 00A0+= GDCSTA EQU 0A0H ;STATUS PORT
484 00A1+= FIFO EQU 0A1H ;GDC FIFO PORT ADDR
485 + ;
486 + ;
487 + ; WRITE
488 00A0+= GDCPAR EQU 0A0H ;PARAMETER INTO FIFO
489 00A1+= GDCCOM EQU 0A1H ;COMMAND INTO FIFO
490 + ;
491 + ;
492 + ; ORGNMISATION OF GRAPHIC RAM
493 + ;
494 + ; 576 X 400 PIXELS
495 + ;
496 1FFF+= GRAEND EQU 1FFFH ;END ADDRESS OF GRAPHIC RAM
497 0048+= MRWAPL EQU 72 ;NUMBER OF WORD ADDR PER LINE
498 0024+= WPL EQU MRWAPL/2 ;WORDS / LINE
499 000A+= LPC EQU 10 ;LINES / CHARACTER
500 + ;
501 + ; MEANING OF GDC STATUS BITS
502 + ;
503 0001+= DATRDY EQU 01H ;A BYTE IS AVAILABLE TO READ
504 0002+= FIFULL EQU 02H ;FIFO IS FULL
505 0004+= FIFEMP EQU 04H ;FIFO IS EMPTY
506 0008+= DRWINP EQU 08H ;DRAWING IN PROCESS
507 0010+= DMAEXC EQU 10H ;DMA DATA TRANSFER IN PROCESS
508 0020+= VERETR EQU 20H ;VERTICAL RETRACE IN PROCESS
509 0040+= HORETR EQU 40H ;HORIZONTAL RETRACE IN PROCESS
510 0080+= LIPDET EQU 80H ;LIGHT PEN DETECT (ADDRESS VALID)
511 + ;
512 + ;
513 + ; COMMANDS
514 + ;
515 0000+= GDCRES EQU 0 ;RESET - BLANK DISPLAY, IDLE MODE, INITIALIZE
516 006E+= VSYNCS EQU 06EH ;SLAVE MODE

```


015 (PBIOS)

```

517 006F+= VSYNC EQU 06FH ;MASTER MODE
518 004B+= CCHAR EQU 04BH ;CURSOR & CHARACTER CHARACTERISTICS
519 0068+= START EQU 068H ;START DISPLAY & END IDLE MODE
520 0046+= ZOOM EQU 046H ;SPECIFY ZOOM FACTOR
521 0049+= CURS EQU 049H ;SPECIFY CURSOR POSITION
522 0047+= PITCH EQU 047H ;PITCH SPECIFICATION
523 004A+= MASK EQU 04AH ;LOAD MASK REGISTER
524 004C+= FIGS EQU 04CH ;SPECIFY FIGURE DRAWING PARAMETER
525 006C+= FIGD EQU 06CH ;START FIGURE DRAW
526 0068+= GCHRD EQU 068H ;START GRAPHICS CHARACTER DRAW
527 00E0+= CURD EQU 0E0H ;READ CURSOR ADDRESS
528 00C0+= LPRD EQU 0C0H ;READ LIGHT PEN ADDRESS
529 + ;
530 0070+= PRAM EQU 070H ;LOAD PARAMETER RAM
531 0000+= PRAMSA EQU 0 ;LOWER 4 BITS ARE STARTING ADDRESS IN RAM
532 + ;( COMMAND + SA )
533 + ;
534 0020+= WDAT EQU 020H ;WRITE DATA INTO DISPLAY MEMORY
535 + ;( COMMAND + TYPE + MODE )
536 + ;DATA TRANSFER TYPES
537 0000+= TYWORD EQU 0 ;WORD, LOW THEN HIGH BYTE
538 0010+= TYLOBY EQU 010H ;LOW BYTE OF THE WORD
539 0018+= TYHIBY EQU 018H ;HIGH BYTE OF THE WORD
540 + ;MODE OF RMW MEMORY CYCLE
541 0000+= MOREPL EQU 0 ;REPLACE WITH PATTERN
542 0001+= MOCMP EQU 01H ;COMPLEMENT
543 0002+= MORES EQU 02H ;RESET TO 0
544 0003+= MOSET EQU 03H ;SET TO 1
545 + ;
546 00A0+= RDAT EQU 0A0H ;READ DATA FROM DISPLAY MEMORY
547 + ;( COMMAND + TYPE )
548 + ;TYPES AS AT WDAT
549 + ;
550 00A4+= DMAR EQU 0A4H ;DMA READ REQUEST
551 + ;( COMMAND + TYPE )
552 + ;TYPES AS AT WDAT
553 + ;
554 0024+= DMAW EQU 024H ;DMA WRITE REQUEST
555 + ;( COMMAND + TYPE + MODE )
556 + ;TYPES AND MODES AS AT WDAT
557 + ;
558 + ; PARAMETERS
559 + ;
560 + ; RESET
561 + ;
562 0000+= RESMOP EQU 0 ;MODE OF OPERATION SELECT BITS
563 + ;( RESMOP + DISPLAY + FRAME + DYNRAM + WINDOW )
564 + ;DISPLAY MODE
565 0000+= MIXGAC EQU 0H ;MIXED GRAPHICS & CHARACTER
566 0002+= GRAMOD EQU 02H ;GRAPHICS MODE
567 0020+= CHAMOD EQU 020H ;CHARACTER MODE
568 + ;VIDEO FRAMING
569 0000+= NOINTL EQU 0 ;NON-INTERLACED
570 0008+= INLRPF EQU 08H ;INTERLACED REPEAT FIELD FOR CHARACTER DISPLAYS
571 0009+= INTLAC EQU 09H ;INTERLACED
572 + ;DYNAMIC RAM REFRESH CYCLES ENABLE

```

016 (PBIOS)

573	0000+ =	SATRM	EQU	0	;NO REFRESH - STATIC RAM
574	0004+ =	DYNRAM	EQU	04H	;REFRESH - DYNAMIC RAM
575	+				;DRAWING TIME WINDOW
576	0000+ =	DRWALL	EQU	0	;DRAWING DURING ACTIVE DISPLAY TIME AND RETRACE
577	0010+ =	DRWRET	EQU	010H	;DRAWING ONLY DURING RETRACE BLANKING
578	+	;			
579	0050+ =	LWID	EQU	80	;CHARACTERS / LINE
580	0018+ =	LINES	EQU	24	;LINES / SCREEN
581	+		PAGE		

```

582      *
583      *          EMOH
584
585      ;*****
586      ;*
587      ;*      CRTOUT      CRT DRIVER
588      ;*
589      ;*****
590
591
592      00A7 C5      CRTOUT: PUSH      B          ;SAVE REGISTER
593      00A8 D5          PUSH      D
594      00A9 E5          PUSH      H
595      00AA 3A4D07      LDA      ESCFLG      ;
596      00AD B7          ORA      A          ; DO ESCAPE SEQUENCE
597      00AE C2D302      JNZ      ESCAPE      ;
598      00B1 3A4C07      LDA      DRQ      ;
599      00B4 B7          ORA      A          ;
600      00B5 C8C00      JZ      VID1      ; JUMP IF NO DATA REQUEST
601      00B8 2A4E07      LHLD     DRQADR      ;
602      00BB E9          PCHL      ; GO TO DATA REQUESTING ROUTINE
603
604
605      00BC 79          VID1:  MOV      A,C          ; CHARACTER --) A
606      00BD E67F          ANI      07FH      ; NO DISPLAY OF CHARACTERS ) 7FH
607      00BF FE20          CPI      ' '      ;
608      00C1 DA8E02      JC      FINDCH      ; JUMP IF CONTROL CHR
609
610      00C4 CDAB02      PUTCHR: CALL     TRNCHR      ; CHARACTER TRANSLATION
611      00C7 79          MOV      A,C          ; GET CHARACTER
612      00C8 CD0200      CALL     MODCRT      ; DISPLAY CHARACTER
613      00CB CD6502      CALL     NXTCUR      ; NEXT CURSOR POSITION
614
615
616      POPREG:
617      ;
618      00CE E1          POP      H          ; RESTORE REGISTER
619      00CF D1          POP      D          ;
620      00D0 C1          POP      B          ;
621      00D1 C9          RET          ; RETURN
622      PAGE

```

012 (PBIUS)

```

622
623
624 ;*****
625 ;*
626 ;* MODCRT PLACE CHARACTER INTO GDC-MEMORY *
627 ;* A - CHARACTER TO TRANSFER *
628 ;*
629 ;*****
630 ;
631 ;
632 MODCRT:
633 0002 F5 PUSH PSW ; SAVE CHARACTER
634 0003 3A4607 LDA CURCOL
635 0006 FE50 CPI LWID
636 ; SCROLL WHOLE SCREEN
637 0008 CC5106 CZ SCLUP4 ; SCROLL IF COLUMN = LWID
638 000B F1 POP PSW ; GET CHARACTER
639 000C 5F MOV E,A
640 000D 3A5007 LDA INVFLG ;GET ATTRIBUTE
641 00E0 57 MOV D,A
642 00E1 CDE500 CALL WRGCHR ; WRITE CHARACTER
643 00E4 C9 RET
644 PAGE
    
```

EC 635 =>

```

645
646          GDCMIX
647      +      ;      GDC DRIVER FOR MIXED MODE 20/01/83 12:30 WF
648      +      ;
649      +      ;      WRITE ONE CHARACTER IN MIXED MODE
650      +      ;      ENTRY REGISTER:
651      +      ;      D = ATTRIBUTE
652      +      ;      E = CHARACTER
653      +      ;
654      +      ;      WRGCHR:
EC 54 - 655 00E5+3E20 1      MVI      A,WDAT OR TYWORD OR MOREPL
656 00E7+C0C07 1      CALL     OUTCMD
657 00EA+78      1      MOV      A,E          ;OUTPUT CHARACTER
658 00EB+CD1207 1      CALL     OUTPAR
659 00EE+7A      1      MOV      A,D          ;OUTPUT ATTRIBUTE
660 00EF+CD1207 1      CALL     OUTPAR
661 00F2+C9      1      RET
662      +      ;
663      +      ;      READ ONE CHARACTER FROM GDC IN MIXED MODE
664      +      ;
665      +      ;      EXIT REGISTERS:
666      +      ;      D = ATTRIBUTE
667      +      ;      E = CHARACTER
EC 65 - 668      +      ;      RDGCHR:
669 00F3+3E4C 1      MVI      A,FIGS          ;FIGS COMMAND
670 00F5+C0C07 1      CALL     OUTCMD
671 00F8+3E02 1      MVI      A,2          ;DIRECTION = 2
672 00FA+CD1207 1      CALL     OUTPAR
673 00FD+3E01 1      MVI      A,1          ;DC = 1
674 00FF+CD1207 1      CALL     OUTPAR
675 0102+AF      1      XRA      A          ;
676 0103+CD1207 1      CALL     OUTPAR
677 0106+3EAD 1      MVI      A,RDAT OR TYWORD
678 0108+C0C07 1      CALL     OUTCMD
679 010B+CD2C07 1      CALL     INPAR          ;GET ASCII CHARACTER
680 010E+5F      1      MOV      E,A
681 010F+CD2C07 1      CALL     INPAR          ;GET ATTRIBUTE
682 0112+57      1      MOV      D,A
683 0113+C9      1      RET
684      +      ;
685      +      ;
686      +      ;      SPCLEAR1:
687 0114+05      1      PUSH     D
688 0115+09      1      DAD      B          ;TEST IF LENGTH OUT OF RANGE
689 0116+110007 1      LXI      D,0700H
690 0119+CD0000 1      CALL     COMPDEHL
691 011C+01      1      POP      D
692 011D+DA3601 1      JC      SPCLEAR2          ;IN RANGE
693 0120+CA3601 1      JZ      SPCLEAR2
694 0123+05      1      PUSH     D
695 0124+110007 1      LXI      D,700H
696 0127+CD0000 1      CALL     SUBHLDE
697 012A+01      1      POP      D
698 012B+CD3601 1      CALL     SPCLEAR2
699 012E+44      1      MOV      B,H
700 012F+4D      1      MOV      C,L

```

```
                                020  (P810S)
701  0130+210000  1    LXI    H,0
702  0133+CD7001  1    CALL   SETCUR1      ;CLEAR REST
703      +      SPCLEAR2:
704  0136+08     1    DCX    B
705  0137+CD8701  1    CALL   SETMSK
706  013A+3E4C   1    MVI    A,FIGS
707  013C+CD0C07  1    CALL   OUTCMD
708  013F+3E02   1    MVI    A,2
709  0141+CD1207  1    CALL   OUTPAR
710  0144+79     1    MOV    A,C
711  0145+CD1207  1    CALL   OUTPAR
712  0148+78     1    MOV    A,B
713  0149+CD1207  1    CALL   OUTPAR
714  014C+3E20   1    MVI    A,WDAT OR TYWORD OR MOREPL
715  014E+CD0C07  1    CALL   OUTCMD
716  0151+78     1    MOV    A,E
717  0152+CD1207  1    CALL   OUTPAR
718  0155+3A5007  1    LDA    INVFLG      ;GET INVERSE FLAG
719  0158+CD1207  1    CALL   OUTPAR
720  015B+C9     1    RET
721      +      PAGE
```

021 (PBIOS)

```

722      +
723      +      ;
724      +      ;   SET CURSOR
725      +      ;   ENTRY REGISTER:
726      +      ;   HL = GDC CURSOR POSITION
727      +      ;
728      +      SETCUR:
729      015C+D5      1      PUSH      D
730      015D+E5      1      PUSH      H
731      015E+2A5D02  1      LHLD      SP1
732      0161+D1      1      POP       D      ;CORRECR CURSOR POSITION
733      0162+19      1      DAD       D
734      0163+110007  1      LXI      D,0700H
735      0166+CD0000  1      CALL     COMPDEHL ;TEST IF OVERFLOW
736      0169+DA6F01  1      JC       SETCUR2  ;NOT OVERFLOW
737      016C+CD0000  1      CALL     SUBHLDE
738      +      SETCUR2:
739      016F+D1      1      POP       D
740      +      SETCUR1:
741      0170+D5      1      PUSH      D
742      0171+3E49      1      MVI      A,CURS      ;CURSOR COMMAND
743      0173+CD0C07  1      CALL     OUTCMD
744      0176+7D      1      MOV      A,L      ;CURSOR POSITION LOW
745      0177+CD1207  1      CALL     OUTPAR
746      017A+7C      1      MOV      A,H      ;CURSOR POSITION HIGH
747      017B+CD1207  1      CALL     OUTPAR
748      017E+AF      1      XRA      A
749      017F+CD1207  1      CALL     OUTPAR
750      0182+CD8701  1      CALL     SETMSK      ;SET MASK
751      0185+D1      1      POP       D
752      0186+C9      1      RET
753      +      ;
754      +      ;
755      +      ;   SET MASK
756      +      ;
757      +      SETMSK:
758      0187+3E4A      1      MVI      A,MASK
759      0189+CD0C07  1      CALL     OUTCMD
760      018C+3EFF      1      MVI      A,-1
761      018E+CD1207  1      CALL     OUTPAR
762      0191+CD1207  1      CALL     OUTPAR
763      0194+C9      1      RET
764      +      ;
765      +      ;
766      +      RDLIN:
767      0195+E5      1      PUSH      H
768      0196+C5      1      PUSH      B
769      0197+3E4C      1      MVI      A,FIGS      ;FIGS COMMAND
770      0199+CD0C07  1      CALL     OUTCMD
771      019C+3E02      1      MVI      A,2      ;DIRECTION = 2
772      019E+CD1207  1      CALL     OUTPAR
773      01A1+3E50      1      MVI      A,80      ;LENGTH = 80 WORDS
774      01A3+CD1207  1      CALL     OUTPAR
775      01A6+AF      1      XRA      A
776      01A7+CD1207  1      CALL     OUTPAR
777      01AA+3EA0      1      MVI      A,RDAT      ;READ INTO BUFFER

```

```

                                022  (PBIOS)

778  01AC+C00C07  1  CALL  OUTCMD
779  01AF+21F9FD  1  LXI   H,LINBUF      ;BUFFER ADDRESS
780  01B2+06A0   1  MVI   B,160        ;NUMBER OF BYTES
781  +          RDLIN1:
782  01B4+CD2C07  1  CALL  INPAR        ;GET CHARACTER
783  01B7+77     1  MOV   M,A
784  01B8+23     1  INX   H
785  01B9+05     1  DCR   B
786  01BA+C28401  1  JNZ   RDLIN1
787  01BD+C1     1  POP   B
788  01BE+E1     1  POP   H
789  01BF+C9     1  RET
790  +          ;
791  +          ;
792  +          ;
793  +          WRLIN:
794  01C0+E5     1  PUSH  H
795  01C1+C5     1  PUSH  B
796  01C2+3E4C   1  MVI   A,FIGS      ;FIGS COMMAND
797  01C4+C00C07  1  CALL  OUTCMD
798  01C7+3E02   1  MVI   A,2        ;DIRECTION = 2
799  01C9+CD1207  1  CALL  OUTPAR
800  01CC+AF     1  XRA   A        ;DC = 0
801  01CD+CD1207  1  CALL  OUTPAR
802  01D0+CD1207  1  CALL  OUTPAR
803  01D3+3E20   1  MVI   A,WDAT OR TYWORD OR MOREPL
804  01D5+C00C07  1  CALL  OUTCMD
805  01D8+21F9FD  1  LXI   H,LINBUF      ;START ADDRESS OF BUFFER
806  01DB+06A0   1  MVI   B,160        ;LENGTH
807  +          WRLIN1:
808  01D0+7E     1  MOV   A,M        ;GET CHARACTER FROM BUFFER
809  01DE+CD1207  1  CALL  OUTPAR
810  01E1+23     1  INX   H
811  01E2+05     1  DCR   B
812  01E3+C2D001  1  JNZ   WRLIN1
813  01E6+C1     1  POP   B
814  01E7+E1     1  POP   H
815  01E8+C9     1  RET
816  +          ;
817  +          INIT10:
818  01E9+F5     1  PUSH  PSW
819  01EA+210000  1  LXI   H,0
820  01ED+225002  1  SHLD  SP1      ;START OF PAGE 1 = 0
821  01F0+226102  1  SHLD  SP2      ;START OF PAGE 2 = 0
822  01F3+AF     1  XRA   A
823  +          ;LENGTH OF PAGE 2 = 0
824  01F4+326402  1  STA   LP22
825  01F7+3E19   1  MVI   A,25      ;LENGTH OF PAGE 1 = 25
826  01F9+326002  1  STA   LP12
827  01FC+3E4C   1  MVI   A,FIGS
828  01FE+C00C07  1  CALL  OUTCMD
829  0201+3E02   1  MVI   A,2
830  0203+CD1207  1  CALL  OUTPAR
831  0206+F1     1  POP   PSW
832  0207+C9     1  RET
833  +          ;

```


023 (PB10S)

```

834      +      ;
835      +      ;
836      +      ;
837      +      CUROFF:
838 0208+3E4B 1     MVI   A,CCHAR
839 020A+CD0C07 1     CALL  OUTCMD
840 0200+3E0F 1     MVI   A,OFH
841 020F+CD1207 1     CALL  OUTPAR      ;CURSOR OFF
842 0212+C9 1     RET
843      +      ;
844      +      ;
845      +      CURON:
846 0213+3E4B 1     MVI   A,CCHAR      ;CURSOR ON
847 0215+CD0C07 1     CALL  OUTCMD
848 0218+3E8F 1     MVI   A,8FH
849 021A+CD1207 1     CALL  OUTPAR
850 021D+3ECE 1     MVI   A,OCEH
851 021F+CD1207 1     CALL  OUTPAR
852 0222+3E72 1     MVI   A,72H
853 0224+CD1207 1     CALL  OUTPAR
854 0227+C9 1     RET
855      +      ;
856      +      ;
857      +      SCROLL:
858 0228+210000 1     LXI   H,0      ;GET START OF PAGE 1
859 022B+015000 1     LXI   B,80
860 022E+CD0E06 1     CALL  CLEARSP      ;CLEAR TOP LINE
861 0231+2A5002 1     LHL   SP1
862 0234+115000 1     LXI   D,80      ;START OF PAGE 1 + ONE LINE
863 0237+19 1     DAD   D
864 0238+225002 1     SHLD  SP1
865 023B+3A6002 1     LDA   LP12      ;LENGTH OF PAGE 1 - 1
866 023E+30 1     DCR   A
867 023F+CCE901 1     CZ    INIT10      ;IF LENGTH = 0 INIT ONCE MORE
868 0242+CA4F02 1     JZ    SCROL1
869 0245+326002 1     STA   LP12
870 0248+3A6402 1     LDA   LP22      ;LENGTH OF PAGE 2 + 1
871 024B+3C 1     INR   A
872 024C+326402 1     STA   LP22
873      +      SCROL1:
874 024F+3E70 1     MVI   A,PRAM+0      ;INITIALIZE PAGES
875 0251+CD0C07 1     CALL  OUTCMD
876 0254+0608 1     MVI   B,8
877 0256+215002 1     LXI   H,SP1
878 0259+CD3207 1     CALL  SENPAR
879 025C+C9 1     RET
880      +      ;
881      +      ;
882 025D+0000  SP1:  DW    0      ;START OF PAGE 1
883 025F+00  LP11:  DB    0      ;LENGTH OF PAGE 1 LOW
884 0260+00  LP12:  DB    0      ;LENGTH OF PAGE 1 HIGH
885 0261+0000  SP2:  DW    0      ;START OF PAGE 2
886 0263+00  LP21:  DB    0      ;LENGTH OF PAGE 2 LOW
887 0264+00  LP22:  DB    0      ;LENGTH OF PAGE 2 HIGH
888      +      ;
889      +      ;

```

024 (PB10S)

```

890      +      PAGE

```

025 (FBIOS)

```

891      +
892      +          ENDM
893
894      ;*****
895      ;*
896      ;*      NXTCUR  NEXT CURSOR POSITION
897      ;*
898      ;*****
899
900
901      NXTCUR:
902      0265 3A4607      LDA      CURCOL      ; GET CURRENT COLUMN NUMBER
903      0268 3C          INR      A          ; COLUMN +1
904      0269 FE50      CPI      LWID      ; IF CURRENT COLUMN + 1 = WIDTH
905      0268 D27202      JNC      NXTCR1     ; THEN JUMP
906      026E 324607      STA      CURCOL      ; SAVE NEW COLUMN
907      0271 C9          RET
908
909
910      NXTCR1:
911      0272 3A4707      LDA      CURROW      ; GET CURRENT ROW
912      0275 FE17      CPI      LINES-1     ; IF ON LAST ROW
913      0277 C27F02      JNZ      NXTCR2     ; "JUMP IF NOT ON LAST ROW"
914      027A 214607      LXI      H,CURCOL
915      027D 34          INR      M          ; INDICATE COLUMN FOR SCROLLING
916      027E C9          RET
917
918
919      027F AF          NXTCR2: XRA      A          ; SET
920      0280 324607      STA      CURCOL      ; COLUMN TO ZERO
921      0283 3A4707      LDA      CURROW
922      0286 3C          INR      A          ; ROW = ROW +1
923      0287 324707      STA      CURROW
924      028A CDED06      CALL     WRCUPO     ;SET CURSOR
925      028D C9          RET
926
927
928      PAGE

```

026 (PRIOS)

```

929
930 ;*****
931 ;*
932 ;* FINDCH SEARCH CHARACTER IN CONTROL CHARACTER TABLE *
933 ;*
934 ;*****
935
936
937 FINDCH:
938 028E 2A4A07 LHL CURPOS ; CURSOR POSITION TO TOP OF STACK
939 0291 E5 PUSH H ;
940 0292 21EA02 LXI H,CONTBL ; POINTER TO COMMAND TABLE
941 ;
942 ;
943 ; GO TO FIND ROUTINE
944 ;
945 ;
946
947 PAGE

```

027 (PRIOS)

```

948
949
950 ;*****
951 ;*
952 ;* FINDAD TRANSFER CONTROL TO A SELECTED *
953 ;* ROUTINE *
954 ;*
955 ;*
956 ;*****
957
958
959 FINDAD:
960 0295 BE CMP M ; CHARACTER FOUND
961 0296 CAA402 JZ LOOKE ; JUMP IF CHR FOUND
962 0299 46 MOV B,M ; MOVE CHARACTER FROM TABLE --> B
963 029A 04 INR B ; TEST FOR LAST CHARACTER
964 029B CAA402 JZ LOOKE ; JUMP AT END
965 029E 23 INX H
966 029F 23 INX H
967 02A0 23 INX H ; NEXT ENTRY
968 02A1 C39502 JMP FINDAD ;
969
970 02A4 23 INX H ; GET ADDRESS --> HL
971 02A5 5E MOV E,M
972 02A6 23 INX H
973 02A7 56 MOV D,M
974 02A8 EB XCHG ;
975 02A9 E3 XTHL ; PUT ADDRESS TO TOP OF STACK
976 02AA C9 RET ; GO TO IT
977
978
979
980
981 PAGE

```

028 (PBIOS)

```

982
983 ;*****
984 ;*
985 ;* TRNCHR TRANSLATE CHARACTER ACCORDING TO THE LANGUAGE *
986 ;* CODE OF THE KBD (IF REQUIRED) *
987 ;* ENTER WITH CHAR IN [C] *
988 ;* EXIT WITH TRANSLATED CHARACTER IN [C] *
989 ;*
990 ;*****
991
992
993 TRNCHR:
994 02AB 3A0000 LDA TRANS
995 02AE B7 ORA A
996 02AF C2C802 JNZ TRNCHR2 ; JUMP IF HEBREW TRANSLATION REQUIRED
997 TRNSPC:
998 02B2 3A0000 LDA LANGUAGE ; GET LANGUAGE CODE
999 ;
1000 ; FIND TRANSLATION TABLE ADDRESS
1001 ;
1002 02B5 210000 LXI H,CRTTBL
1003 02B8 C00000 CALL GETLPOS ; GET ADDRESS OF TRANSLATION TABLE
1004 02BB 46 MOV B,M ; B - LENGTH OF SPEC. TRANSLATION ENTRY
1005 ;
1006 ;
1007 ; TRANSLATE IF REQUIRED
1008 ;
1009 TRNCHR1:
1010 02BC 05 DCR B ; DECREMENT LENGTH
1011 02BD 23 INX H
1012 02BE 7E MOV A,M ; GET CHARACTER
1013 02BF C8 RZ ; RETURN IF NO TRANSLATION REQ.
1014 02C0 05 DCR B ; DECREMENT LENGTH
1015 02C1 B9 CMP C
1016 02C2 23 INX H
1017 02C3 C2B802 JNZ TRNCHR1 ; JUMP IF NOT =
1018 02C6 4E MOV C,M ; TRANSLATED CHAR. -> C
1019 02C7 C9 RET
1020
1021 TRNCHR2:
1022 02C8 79 MOV A,C
1023 02C9 FE60 CPI 60H
1024 02CB D8 RC ; RETURN IF NO TRANSLATION REQUIRED
1025 02CC FE78 CPI 78H
1026 02CE D0 RNC ; RETURN IF NO TRANSLATION REQUIRED
1027 02CF E61F ANI 01FH ; TRANSLATE 60H-7AH -> 00H-1AH
1028 02D1 4F MOV C,A ; CHARACTER -> C
1029 02D2 C9 RET
1030
1031
1032
1033 PAGE

```

```

1034
1035 ;*****
1036 ;*
1037 ;* ESCAPE ESCAPE HANDLING *
1038 ;* A - CONTROL CHARACTER *
1039 ;*
1040 ;*****
1041
1042
1043
1044
1045 02D3 AF ESCAPE: XRA A ;
1046 02D4 324D07 STA ESCFLG ; RESET ESCAPE FLAG
1047 02D7 79 MOV A,C ;
1048 02D8 2A4A07 LHL CURPOS ;
1049 02D8 E5 PUSH H ; DUMMY PUSH
1050 02DC 210803 LXI H,ESCTB ; ESCAPE TABLE
1051 02DF C39502 JMP FINDAO ; SEARCH THE CORRECT ESCAPE SEQUENCE
1052
1053
1054
1055
1056
1057 02E2 3EFF SETESC: MVI A,OFFH ; SET ESCAPE FLAG
1058 02E4 324D07 STA ESCFLG
1059 02E7 C3CE00 JMP POPREG
1060
1061
1062
1063 PAGE

```

030 (PBIOS)

```

1064
1065 ;*****
1066 ;*                                     *
1067 ;*   CONTBL  SINGLE BYTE CRT CONTROL FUNCTIONS   *
1068 ;*                                     *
1069 ;*****
1070
1071
1072 CONTBL:
1073 02EA 07          DB      07H
1074 02EB 3304      DW      BELL    ; RING THE BELL
1075 02ED 08          DB      08H
1076 02EE 5F05      DW      BACKSP  ; NON DESTRUCTIVE BACKWARD SPACE
1077 02F0 0A          DB      0AH
1078 02F1 5305      DW      LINEFD  ; LINE FEED
1079 02F3 0B          DB      0BH
1080 02F4 8305      DW      RLFEED  ; REVERSE LINE FEED
1081 02F6 0C          DB      0CH
1082 02F7 2805      DW      FSPACE  ; NON DESTRUCTIVE FORWARD SPACE
1083 02F9 0D          DB      0DH
1084 02FA 5905      DW      CARRET  ; CARRIAGE RETURN
1085 02FC 17          DB      17H
1086 02FD 2205      DW      EREOL   ; ERASE TO END OF LINE
1087 02FF 1A          DB      1AH
1088 0300 A305      DW      CLEAR   ; CLEAR SCREEN
1089 0302 1B          DB      1BH
1090 0303 E202      DW      SETESC  ; ESCAPE SEQUENCES
1091 0305 1E          DB      1EH
1092 0306 9305      DW      CURHOM  ; HOME CURSOR
1093 0308 FF          DB      OFFH   ; END OF TABLE
1094 0309 CE00      DW      POPREG  ; DO NOTHING ROUTINE
1095
1096
1097 PAGE

```

```

1098
1099
1100 ;*****
1101 ;*
1102 ;*   ESCTBL  TABLE OF IMPLEMENTED ESCAPE FUNCTIONS   *
1103 ;*
1104 ;*****
1105
1106
1107
1108 0308 2A      ESCTBL: DB    2AH
1109 030C A305   DW    CLEAR      ; CLEAR SCREEN
1110 030E 3A     DB    3AH
1111 030F A305   DW    CLEAR      ; CLEAR SCREEN
1112 0311 29     DB    29H
1113 0312 7B03  DW    HALFSET    ; SET HALF INTENSITY
1114 0314 28     DB    28H
1115 0315 9103  DW    HALFRES    ; RESET HALF INTENSITY
1116 0317 59     DB    'Y'
1117 0318 BE05  DW    CLREOS     ; ERASE TO END OF SCREEN
1118 031A 79     DB    'y'
1119 031B BE05  DW    CLREOS     ; ERASE TO END OF SCREEN
1120 031D 54     DB    'T'
1121 031E 2205  DW    EREOL      ; ERASE TO END OF LINE
1122 0320 74     DB    't'
1123 0321 2205  DW    EREOL      ; ERASE TO END OF LINE
1124 0323 45     DB    'E'
1125 0324 7904  DW    INLINE     ; INSERT LINE
1126 0326 52     DB    'R'
1127 0327 8204  DW    DELINE     ; DELETE LINE
1128 0329 51     DB    'Q'
1129 032A 8804  DW    INCHAR     ; INSERT CHARACTER
1130 032C 57     DB    'W'
1131 032D DA04  DW    DECHAR     ; DELETE CHARACTER
1132 032F 30     DB    '='
1133 0330 3E03  DW    POSCUR     ; POSITION CURSOR
1134 0332 4D     DB    'M'
1135 0333 4E04  DW    MUSIC      ; MUSIC
1136 0335 46     DB    'F'
1137 0336 0000  DW    FUNCCH     ; CHANGE FUNCTION
1138 0338 47     DB    'G'
1139 0339 A703  DW    REVERSE    ; INVERSE VIDEO
1140 033B FF     DB    OFFH
1141 033C CE00  DW    POPREG
1142
1143
1144
1145 PAGE

```

032 (PBIOS)

```

1146
1147 ;*****
1148 ;#
1149 ;#   POSCUR   POSITION CURSOR
1150 ;#
1151 ;*****
1152
1153
1154 POSCUR:
1155 033E 3EFF      MVI   A,OFFH      ; SET
1156 0340 324C07   STA   DRQ          ; DATA REQUEST FLAG
1157 0343 214C03   LXI   H,GETY       ;
1158 0346 224E07   SHLD  DRQADR       ; SET DRQ ADDRESS
1159 0349 C3CE00   JMP   POPREG       ; RETURN
1160
1161
1162 GETY:
1163 034C 79        MOV   A,C           ; GET Y-POSITION
1164 034D D620      SUI   ' '          ; SUBTRACT 20H
1165 034F FE19      CPI   25
1166 0351 D25703   JNC   GETY1        ; JUMP IF OUT OF RANGE
1167 0354 325307   STA   YPOS         ; SAVE Y-POSITION
1168
1169 GETY1:
1170 0357 216003   LXI   H,GETX       ;
1171 035A 224E07   SHLD  DRQADR       ; NEXT DRQ ADDRESS
1172
1173
1174 GETX:  MOV   A,C           ; GET X-POSITION
1175 0361 D620      SUI   ' '          ; SUBTRACT 20H
1176 0363 FE50      CPI   80
1177 0365 D26803   JNC   GETX1        ; JUMP IF OUT OF RANGE
1178 0368 324607   STA   CURCOL       ; SAVE X POSITION
1179
1180 GETX1:
1181 0368 AF        XRA   A            ; CLEAR
1182 036C 324C07   STA   DRQ          ; DATA REQUEST FLAG
1183 036F 3A5307   LDA   YPOS         ; GET Y-POSITION
1184 0372 324707   STA   CURROW       ; SET Y-POS
1185 0375 CD0D06   CALL  WRCUPO
1186 0378 C3CE00   JMP   POPREG       ;
1187
1188
1189 PAGE

```



```

1190
1191 ;*****
1192 ;#
1193 ;* HALFSET SET FLAG FOR HALF INTENSITY
1194 ;#
1195 ;*****
1196
1197
1198 HALFSET:
1199 0378 0604 MVI B,4
1200 037D 3A5407 LDA COLORI ; GET COLOR INDEX
1201 0380 FE43 CPI 'C'
1202 0382 C28703 JNZ INVS1 ; JUMP IF MONOCHROM
1203 0385 0605 MVI B,5
1204
1205 INVS1:
1206 0387 3A5007 LDA INVFLG ; GET ATTRIBUTE BYTE
1207 038A B0 ORA B ; SET HALF INTENSITY
1208 0388 325007 STA INVFLG ; SET HALF INTENSITY FLAG
1209 038E C3CE00 JMP POPREG
1210
1211
1212 ;*****
1213 ;#
1214 ;* HALFRES RESET HALF INTENSITY
1215 ;#
1216 ;*****
1217
1218 HALFRES:
1219 0391 06FB MVI B,0FBH
1220 0393 3A5407 LDA COLORI ; GET COLOR INDEX
1221 0396 FE43 CPI 'C'
1222 0398 C29D03 JNZ INVR1 ; JUMP IF MONOCHROM
1223 0398 06FA MVI B,0FAH
1224
1225 INVR1:
1226 039D 3A5007 LDA INVFLG ; GET ATTRIBUTE BYTE
1227 03A0 A0 ANA B ; RESET HALF INTENSITY FLAG
1228 03A1 325007 STA INVFLG
1229 03A4 C3CE00 JMP POPREG
1230
1231
1232 PAGE

```

034 (PBIOS)

```

1233
1234
1235 ;*****
1236 ;*
1237 ;* REVERSE SET / RESET OF INVERSE VIDEO
1238 ;*
1239 ;*****
1240 ;
1241 ;
1242 ;
1243 ;
1244 REVERSE:
1245 03A7 3EFF MVI A,OFFH ; SET DATA REQUEST FLAG
1246 03A9 324C07 STA DRQ
1247 03AC 21B503 LXI H,RV0
1248 03AF 224E07 SHLD DRQADR ; SAVE DRQ ADDRESS
1249 03B2 C3CE00 JMP POPREG
1250 ;
1251 ; SET/RESET INVERS AND/OR BLINKING
1252 ;
1253 RV0:
1254 03B5 79 MOV A,C ; TEST FOR SET / RESET INVERSE VIDEO
1255 03B6 FE30 CPI '0'
1256 03B8 C2F903 JNZ RV1 ; IF RESET INVERSE AND BLINKING
1257 03BB 3A5007 LDA INVFLG
1258 03BE E6FD ANI OFDH ; RESET BLINKING
1259 03C0 325007 STA INVFLG
1260 03C3 3A5407 LDA COLORI ; GET COLOR INDEX
1261 03C6 FE43 CPI 'C'
1262 03C8 C2F103 JNZ RV2 ; JUMP IF MONOCHROM
1263 03CB 3A5107 LDA REVVID ; GET REVERSE VIDEO ON OFF FLAG
1264 03CE B7 ORA A
1265 03CF CA2C04 JZ RVE1 ; RETURN IF REVERSE VIDEO STILL RESET
1266 03D2 AF XRA A
1267 03D3 325107 STA REVVID ; SET REVERSE VIDEO OFF
1268
1269 03D6 3A5007 RV3: LDA INVFLG
1270 03D9 47 MOV B,A ; SAVE FOR LATER
1271 03DA 07 RLC
1272 03DB 07 RLC
1273 03DC 07 RLC
1274 03DD 2F CMA ; COMPLEMENT FOREGROUND COLOR
1275 03DE E6E0 ANI DEOH ; MASK NEW BACKGROUND COLOR
1276 03E0 4F MOV C,A ; SAVE FOR LATER
1277 03E1 78 MOV A,B ; GET ORIGINAL
1278 03E2 0F RRC
1279 03E3 0F RRC
1280 03E4 0F RRC
1281 03E5 2F CMA ; COMPLEMENT BACKGROUND COLOR
1282 03E6 E61C ANI D1CH ; MASK NEW FOREGROUND COLOR
1283 03E8 B1 ORA C ; COMBINE WITH BACKGROUND COLOR
1284 03E9 4F MOV C,A ; SAVE AGAIN
1285 03EA 78 MOV A,B
1286 03EB E603 ANI 03H ; MASK BLINK AND HALF DENSITY
1287 03ED B1 ORA C
1288 03EE C32904 JMP RVE

```

```

035 (PBIOS)

1289
1290 03F1 3A5007
1291 03F4 E6FE
1292 03F6 C32904
1293
1294 03F9 FE32
1295 03FB C20604
1296 03FE 3A5007
1297 0401 F602
1298 0403 C32904
1299
1300 0406 FE34
1301 0408 C22C04
1302 040B 3A5407
1303 040E FE43
1304 0410 CA1804
1305 0413 3A5007
1306 0416 F601
1307 0418 C32904
1308
1309 041B 3A5107
1310 041E B7
1311 041F C22C04
1312 0422 3C
1313 0423 325107
1314 0426 C3D603
1315
1316 0429 325007
1317
1318 042C AF
1319 042D 324C07
1320 0430 C3CE00
1321
1322
1323

RV2:
LDA INVFLG ; THEN
ANI OFEH ; RESET INVERSE VIDEO FLAG
JMP RVE

RV1:
; ELSE
CFI '2'
JNZ RV5 ; IF SET BLINKING
LDA INVFLG
ORI 02H ; SET BLINKING
JMP RVE ; RETURN

RV5:
CPI '4' ; IF SET INVERSE
JNZ RV1 ;
LDA COLORI ; GET COLOR INDEX
CPI 'C'
JZ RV4 ; JUMP IF COLOR
LDA INVFLG ; THEN
ORI 1 ; SET INVERSE VIDEO FLAG
JMP RVE ; ENDDIF

RV4:
LDA REVVID ; GET REVERSE VIDEO ON OFF FLAG
ORA A
JNZ RVE1 ; RETURN IF REVERSE VIDEO ALWAYS SET
INR A
STA REVVID ; SET REVERSE VIDEO ON
JMP RV3

RVE:
;
STA INVFLG ; SET/RESET ATTRIBUTE'S
; ENDDIF
RVE1:
XRA A ; RESET DRQ FLAG
STA DRQ
JMP POPREG ; RETURN

PAGE

```

036 (PBIOS)

```

1324
1325
1326 ;*****
1327 ;#
1328 ;# BELL RING THE BELL
1329 ;#
1330 ;*****
1331
1332 BELL:
1333 0433 C03904 CALL BELLOUT ; BELL
1334 0436 C3CED0 JMP POPREG
1335 ;
1336 ; A - COMMAND/DATA BYTE
1337 ;
1338 BELLOUT:
1339 0439 4F MOV C,A ; SAVE COMMAND/DATA BYTE
1340 043A D841 BELL2: IN RSKEY ; GET STATUS FROM KBD
1341 043C E601 ANI OUTPBF
1342 043E CA4304 JZ BELL1 ; JUMP IF OUTPUTBUFFER EMPTY
1343 0441 D840 IN RKEY ; DUMMY READ NEEDED BY THE 8041
1344 0443 D841 BELL1: IN RSKEY
1345 0445 E602 ANI IMPBFF
1346 0447 C23A04 JNZ BELL2 ; JUMP IF INPUT BUFFER FULL
1347 044A 79 MOV A,C ; RESTORE COMMAND/DATA BYTE
1348 044B D341 F02B OUT KBELL
1349 044D C9 RET
1350
1351 PAGE

```

```

1352
1353 ;*****
1354 ;*
1355 ;* MUSIC   PLAY MUSIC INTERFACE
1356 ;*
1357 ;*****
1358 ;
1359 ;
1360 ;
1361 ;
1362 044E 3EFF MUSIC: MVI   A,OFFH
1363 0450 324C07      STA   DRQ       ; SET DATA REQ. FLAG
1364 0453 216104     LXI   H,GETFRQ
1365 0456 224E07     SHLD  DRQADR     ; SET DATA REQUEST ADDRESS
1366 0459 3E06      MVI   A,MUSICC
1367 0458 CD3904     CALL  BELLOUT    ; SEND COMMAND
1368 045E C3CE00     JMP   POPREG
1369 ;
1370 GETFRQ:
1371 0461 79         MOV   A,C
1372 0462 CD3904     CALL  BELLOUT    ; SEND FREQUENCY
1373 0465 216E04     LXI   H,GETLEN
1374 0468 224E07     SHLD  DRQADR     ; SET DATA REQUEST ADDRESS
1375 046B C3CE00     JMP   POPREG
1376 ;
1377 ;
1378 ;
1379 GETLEN:
1380 046E 79         MOV   A,C
1381 046F CD3904     CALL  BELLOUT    ; SEND LENGTH OF FREQ.
1382 0472 AF         XRA   A
1383 0473 324C07     STA   DRQ       ; RESET DATA REQUEST FLAG
1384 0476 C3CE00     JMP   POPREG
1385
1386
1387
1388 PAGE

```

038 (PBIO5)

```

1389
1390
1391 ;*****
1392 ;*
1393 ;*   INLINE   INSERT LINE AT CURRENT CURSOR POSITION   *
1394 ;*
1395 ;*****
1396
1397
1398 0479 3A4707  INLINE: LDA   CURROW       ; GET Y-POSITION
1399 047C C08106        CALL   SCROLLDN    ; MOVE REST OF SCREEN DOWN
1400 047F C3CE00        JMP    POPREG
1401
1402
1403
1404 ;*****
1405 ;*
1406 ;*   DELINE  DELETE LINE AT CURRENT CURSOR POSITION   *
1407 ;*
1408 ;*****
1409
1410
1411 DELINE:
1412 0482 3A4707  LDA   CURROW       ; GET Y-POSITION
1413 0485 C02A06  CALL   SCROLLUP    ; MOVE REST OF SCREEN UP
1414 0488 C3CE00  JMP    POPREG       ;
1415
1416
1417 PAGE

```

```

1418
1419 ;*****
1420 ;*
1421 ;* INCHAR INSERT CHARACTER AT CURRENTCURSOR POSITION *
1422 ;*
1423 ;*****
1424
1425
1426 INCHAR:
1427 048B 2A4607 LHL D CURCOL
1428 048E CDF806 CALL WRHLPOS
1429 0491 3A4607 LDA CURCOL
1430 0494 4F MOV C,A
1431 0495 3E4F MVI A,LWID-1 ; A - NUMBER OF CHAR TO MOVE
1432 0497 91 SUB C
1433 0498 CAC204 JZ INCHA2
1434
1435 049B 4F MOV C,A
1436
1437 049C 2A4607 LHL D CURCOL ; HL - CURRENT POSITION
1438 049F 2E4E MVI L,LWID-2 ; START ADDRESS FOR MOVE
1439
1440 04A1 CDF806 CALL WRHLPOS
1441 04A4 CD0802 CALL CUROFF ; CURSOR OFF
1442
1443 INCHA1:
1443 04A7 E5 PUSH H
1444 04A8 CD5C01 CALL SETCUR ; SET CURSOR
1445 04AB CDF300 CALL RDGCHR ; GET CHARACTER
1446 04AE E1 POP H
1447 04AF 23 INX H
1448 04B0 E5 PUSH H
1449 04B1 CD5C01 CALL SETCUR ; SET CURSOR
1450 04B4 CDE500 CALL WRGCHR ; SET CHARACTER
1451 04B7 E1 POP H
1452 04B8 2B DCX H
1453 04B9 2B DCX H
1454 04BA 0D DCR C
1455 04BB C2A704 JNZ INCHA1 ; JUMP IF NOT ALL CHAR MOVED
1456 04BE CD1302 CALL CURON ; CURSOR ON
1457 04C1 23 INX H
1458
1459 INCHA2:
1459 04C2 E5 PUSH H
1460 04C3 0E20 MVI C,' ' ; INSERT A SPACE
1461 04C5 CDAB02 CALL TRNCHR ; TRANSLATE THE SPACE
1462 04C8 E1 POP H
1463 04C9 59 MOV E,C ; CHARACTER --> E
1464 04CA CD5C01 CALL SETCUR ; SET CURSOR
1465 04CD 3A5007 LDA INVFLG ; GET ATTRIBUTE
1466 04D0 57 MOV D,A
1467 04D1 CDE500 CALL WRGCHR ; CLEAR CHARACTER
1468 04D4 CDED06 CALL WRCUPO ; SET CURSOR
1469 04D7 C3CE00 JMP POPREG
1470
1471
1472
1473 PAGE

```

040 (PBIOS)

```

1474
1475 ;*****
1476 ;*
1477 ;*   DECHAR: DELETE CHARACTER AT CURRENT CURSOR POSITION
1478 ;*
1479 ;*****
1480
1481
1482 DECHAR:
1483 04DA 2A4607      LHL  CURCOL
1484 04DD CDFB06      CALL WRHLPOS
1485 04E0 3A4607      LDA  CURCOL
1486 04E3 4F          MOV  C,A
1487 04E4 3E4F        MVI  A,LWID-1
1488 04E6 91          SUB  C          ; A - NUMBER OF CHAR TO MOVE
1489 04E7 CADA05      JZ   DECHA2    ; JUMP IF NO CHAR TO MOVE
1490
1491 04EA 4F          MOV  C,A
1492
1493 04EB 23          INX  H          ; HL - START ADDRESS TO MOVE
1494 04EC C00802      CALL CUROFF    ; CURSOR OFF
1495
1496 DECHA1:
1497 04EF E5          PUSH H
1498 04F0 C05C01      CALL SETCUR    ; SET CURSOR
1499 04F3 CDF300      CALL RDGCHR    ; GET CHARACTER
1500 04F6 E1          POP  H
1501 04F7 2B          DCX  H
1502 04F8 E5          PUSH H
1503 04F9 C05C01      CALL SETCUR    ; SET CURSOR
1504 04FC C0E500      CALL WRGCHR    ; SET CHARACTER
1505 04FF E1          POP  H
1506 0500 23          INX  H
1507 0501 23          INX  H
1508 0502 00          DCR  C
1509 0503 C2EF04      JMZ  DECHA1    ; JUMP IF NOT ALL CHAR MOVED
1510 0506 C01302      CALL CURON     ; CURSOR ON
1511 0509 2B          DCX  H
1512 DECHA2:
1513 050A E5          PUSH H
1514 050B 0E20        MVI  C,' '     ; INSERT A SPACE
1515 050D C0AB02      CALL TRNCHR    ; TRANSLATE THE SPACE
1516 0510 E1          POP  H
1517 0511 59          MOV  E,C       ; CHARACTER --> E
1518 0512 C05C01      CALL SETCUR    ; SET CURSOR
1519 0515 3A5007      LDA  INVFLG    ; GET ATTRIBUTE
1520 0518 57          MOV  D,A
1521 0519 C0E500      CALL WRGCHR    ; CLEAR LAST CHARACTER
1522 051C C0E006      CALL WRCUPO    ; SET CURSOR
1523 051F C3CE00      JMP  POPREG
1524
1525
1526
1527 PAGE

```


041 (PBIOS)

```
1528
1529 ;*****
1530 ;*
1531 ;* EREOL ERASE FROM CURRENT POSITION TO END OF LINE *
1532 ;*
1533 ;*****
1534
1535
1536 0522 00E805 EREOL: CALL CLREOL ; CLEAR TO END OF LINE
1537 0525 03CE00 JMP POPREG ;
1538
1539
1540 PAGE
```

042 (PBIOS)

```

1541
1542 ;*****
1543 ;*
1544 ;* FSPACE FORWARD SPACE
1545 ;*
1546 ;*****
1547
1548 FSPACE:
1549 0528 3A4607 LDA CURCOL ; GET X-POSITION + 1
1550 0528 3C INR A ;
1551 052C 324607 STA CURCOL
1552 052F FE50 CPI LVID ;
1553 0531 DA3A05 JC FSPACE1 ; FALL THRU IF LINEFEED REQUIRED
1554 0534 CD2206 CALL COLUMO ; SET X-POSITION TO 0
1555 0537 CD4005 CALL LFEED ; LINEFEED
1556
1557 053A CD0D06 FSPACE1: CALL WRCUPO ; SET CURSOR
1558 053D C3CE00 JMP POPREG ;
1559
1560
1561
1562 ;*****
1563 ;*
1564 ;* LFEED LINE FEED
1565 ;*
1566 ;*****
1567
1568 LFEED:
1569 0540 3A4707 LDA CURROW ; Y-POSITION + 1
1570 0543 3C INR A ;
1571 0544 FE18 CPI LINES
1572 0546 D25005 JNC LFEED1 ; JUMP IF SCROLL REQUIRED
1573 0549 324707 STA CURROW
1574 054C CD0D06 CALL WRCUPO ; UPDATE CURRENT CURSOR POSITION
1575 054F C9 RET
1576
1577 0550 C35A06 LFEED1: JMP SCLUP3
1578
1579
1580
1581 ;*****
1582 ;*
1583 ;* LINEFO LINEFEED
1584 ;*
1585 ;*****
1586
1587
1588 LINEFO:
1589 0553 CD4005 CALL LFEED ; LINEFEED
1590 0556 C3CE00 JMP POPREG
1591
1592
1593 PAGE

```

043 (PBIOS)

```

1594
1595 ;*****
1596 ;*
1597 ;*   CARRET   CARRIAGE RETURN
1598 ;*
1599 ;*****
1600
1601 CARRET:
1602 0559 C02206   CALL   COLUMO       ; SET X-POSITION = 0
1603 055C C3CE00   JMP    POPREG           ;
1604
1605
1606
1607 ;*****
1608 ;*
1609 ;*   BACKSP  BACKSPACE
1610 ;*
1611 ;*****
1612
1613 BACKSP:
1614 055F 3A4607   LDA    CURCOL
1615 0562 30       DCR    A                ; X-POSITION - 1
1616 0563 F27A05   JP     BKSP1           ; JUMP IF X-POSITION >= 0
1617 0566 3A4707   LDA    CURROW
1618 0569 30       DCR    A                ; Y-POSITION - 1
1619 056A F27505   JP     BKSP2           ; JUMP IF Y-POSITION >= 0
1620 056D 3E00     MVI    A,0             ;
1621 056F 324707   STA    CURROW          ; SET Y-POSITION = 0
1622 0572 C37A05   JMP    BKSP1
1623
1624 0575 324707   STA    CURROW          ; SET Y-POSITION
1625 0578 3E4F     MVI    A,LWID-1        ; X-POSITION = SCREEN WIDTH
1626
1627 057A 324607   STA    CURCOL          ; SET NEW X-POSITION
1628 057D C0ED06   CALL   WRCUPO          ; SET CURSOR
1629 0580 C3CE00   JMP    POPREG           ;
1630
1631 PAGE

```

044 (PBIOS)

```
1632
1633 ;*****
1634 ;*                                     *
1635 ;*      RLFEED REVERSE LINE FEED      *
1636 ;*                                     *
1637 ;*****
1638
1639 RLFEED:
1640 0583 3A4707      LDA      CURROW      ; Y-POSITION - 1
1641 0586 30          DCR      A          ;
1642 0587 FA8005      JM       RLFEED1
1643 058A 324707      STA      CURROW      ;
1644
1645 058D C0E006      CALL     WRCUPO      ; SET CURSOR
1646 0590 C3CE0C      JMP      FOPREG
1647
1648
1649 PAGE
```

```

1650
1651 ;*****
1652 ;*
1653 ;* CURHOM CURSOR HOME
1654 ;*
1655 ;*****
1656
1657 CURHOM:
1658 0593 CD9905 CALL HOME ; CURSOR HOME
1659 0596 C3CE00 JMP POPREG ;
1660
1661
1662 ;*****
1663 ;*
1664 ;* HOME HOME CURSOR SUBROUTINE
1665 ;*
1666 ;*****
1667
1668 HOME:
1669 0599 210000 LXI H,0 ; X-POSITION = 0
1670 059C 224607 SHLD CURCOL ; Y-POSITION = 0
1671 059F CDE006 CALL WRCUPO ; SET CURSOR
1672 05A2 C9 RET
1673
1674
1675
1676 ;*****
1677 ;*
1678 ;* CLEAR CLEAR SCREEN, CURSOR HOME
1679 ;*
1680 ;*****
1681
1682
1683 CLEAR:
1684 05A3 CD0802 CALL CUROFF
1685 05A6 CDE901 CALL INIT10 ; INITIALIZE PAGES
1686 05A9 CD4F02 CALL SCROL1
1687 05AC 210000 LXI H,0 ; START OF CRT RAM
1688 05AF 010007 LXI B,LINES*LWID+LWID ; LENGTH TO CLEAR
1689 05B2 CD0E06 CALL CLEARSP
1690 05B5 CD9905 CALL HOME ; CURSOR HOME
1691 05B8 CD1302 CALL CUROW
1692 05BB C3CE00 JMP POPREG ;
1693
1694
1695
1696
1697
1698 PAGE

```

046 (PRIOS)

```

1699
1700 ;*****
1701 ;*
1702 ;* CLREOS CLEAR FROM CURRENT POSITION TO END OF SCREEN *
1703 ;*
1704 ;*****
1705
1706
1707 CLREOS:
1708 058E 3A4707 LDA CURROW
1709 05C1 4F MOV C,A
1710 05C2 3E17 MVI A,LINES-1 ; A = NUMBER OF LINES TO CLEAR
1711 05C4 91 SUB C
1712 05C5 CAE205 JZ CLREOS1 ; JUMP IF ONLY ONE LINE TO CLEAR
1713
1714 05C8 2E00 MVI L,0
1715 05CA 61 MOV H,C
1716 05CB 24 INR H
1717 05CC CDFB06 CALL WRHLPOS ; HL - START OF CLEAR AREA
1718 05CF E5 PUSH H
1719 05D0 215000 LXI H,LWID
1720 05D3 CD0000 CALL MULT ;
1721 05D6 4D MOV C,L
1722 05D7 44 MOV B,H ; BC - NUMBER OF BYTES TO CLEAR
1723 05D8 E1 POP H
1724 05D9 CD0802 CALL CUROFF
1725 05DC CD0E06 CALL CLEARSP ; CLEAR BC BYTES STARTING WITH HL
1726 05DF CD1302 CALL CURON
1727 CLREOS1:
1728 05E2 CDE805 CALL CLREOL ; CLEAR TO END OF LINE
1729 05E5 C3CE00 JMP POPREG
1730
1731 PAGE

```

047 (PBIOS)

```

1732
1733 ;*****
1734 ;*                                     *
1735 ;*   CLREOL  CLEAR FROM CURRENT POSITION TO END OF LINE   *
1736 ;*                                     *
1737 ;*****
1738
1739 CLREOL:
1740 05E8 3A4607      LDA   CURCOL
1741 05EB 47          MOV   B,A
1742 05EC 6F          MOV   L,A
1743 05ED 3F50        MVI   A,LWTO
1744 05EF 90          SUB   B           ; A - NUMBER OF BYTES TO CLEAR
1745 05F0 C8          RZ           ; NOTHING TO CLEAR
1746 05F1 0600        MVI   B,0           ;
1747 05F3 4F          MOV   C,A           ;
1748 05F4 C5          PUSH  B
1749 05F5 3A4707      LDA   CURROW
1750 05F8 67          MOV   H,A
1751 05F9 CDFB06      CALL  WRHLPOS      ; HL - A(CHARACTER IN CRT-RAM)
1752 05FC C1          POP   B           ; BC - # OF BYTES TO CLEAR
1753 05FD C00E06      CALL  CLEARSP     ; CLEAR REQUESTED NUMBER OF BYTES
1754 0600 C9          RET
1755
1756
1757
1758 ;*****
1759 ;*                                     *
1760 ;*   CLRLIN  CLEAR LINE                                     *
1761 ;*           A - LINE NUMBER                               *
1762 ;*                                     *
1763 ;*****
1764
1765 CLRLI1:
1766 0601 67          MOV   H,A           ;
1767 0602 2E00        MVI   L,0           ; X-POSITION = 0
1768 0604 CDFB06      CALL  WRHLPOS     ; HL - A (LINE IN CRT RAM)
1769 0607 015000      LXI   B,LWID       ; SCREEN WIDTH
1770 060A C00E06      CALL  CLEARSP     ; CLEAR THE LINE
1771 060D C9          RET
1772
1773 PAGE

```

048 (PBIOS)

```

1774
1775
1776 ;*****
1777 ;*
1778 ;* CLEARSP CLEAR BC - BYTES IN CRT-RAM
1779 ;* HL - START ADDRESS IN CRT RAM
1780 ;*
1781 ;*****
1782
1783
1784 CLEARSP:
1785 060E E5 PUSH H
1786 060F C5 PUSH B ;
1787 0610 0E20 MVI C, ' ' ;
1788 0612 CDAB02 CALL TRNCHR ; TRANSLATE SPACE
1789 0615 59 MOV E,C ;
1790 0616 C1 POP B ;
1791 0617 E1 POP H
1792 0618 CD5C01 CALL SETCUR ; SET CURSOR
1793 061B CD1401 CALL SPCLEAR1 ; GDC CLEAR ROUTINE
1794 061E CD0D06 CALL WRCUPO
1795 0621 C9 RET
1796 PAGE
    
```

049 (PBIOS)

```

1797
1798 ;*****
1799 ;*
1800 ;* COLUMO SET X-POSITION = 0
1801 ;*
1802 ;*****
1803
1804 COLUMO:
1805 0622 AF XRA A ;
1806 0623 324607 STA CURCOL ; SET X-POSITION = 0
1807 0626 CD0D06 CALL WRCUPO ; SET CURSOR
1808 0629 C9 RET
1809
1810
1811
1812
1813 PAGE
    
```


050 (PBIOS)

```

1814
1815 ;*****
1816 ;*
1817 ;* SCROLLUP MOVES SCREEN ONE LINE UP
1818 ;* A - SECIFIED LINE
1819 ;*
1820 ;*****
1821
1822
1823 SCROLLUP:
1824 062A B7 ORA A
1825 062B CA5A06 JZ SCLUP3 ; SCROLL ALL
1826 062E 47 MOV B,A
1827 062F 3E17 MVI A,LINES-1
1828 0631 90 SUB B ; A - NUMBER OF LINES TO MOVE
1829 0632 CA4406 JZ SCLUP2 ; JUMP IF NO LINE TO MOVE
1830 0635 4F MOV C,A
1831 0636 CD0802 CALL CUROFF ; CURSOR OFF
1832 SCLUP1:
1833 0639 CDCA06 CALL MUROW ; MOVE ONE LINE UP
1834 063C 04 INR B
1835 063D 00 DCR C
1836 063E C23906 JNZ SCLUP1 ; NEXT LINE
1837 0641 CD1302 CALL CURON ; CURSOR ON
1838 ;
1839 SCLUP2:
1840 0644 3E17 MVI A,LINES-1 ; CLEAR LINE 25
1841 0646 CD0000 CALL CLR LIN
1842 0649 AF XRA A
1843 064A 324607 STA CURCOL ; X - POS = 0
1844 064D CD0E06 CALL WRCUPO ; SET CURSOR
1845 0650 C9 RET
1846
1847 SCLUP4:
1848 0651 AF XRA A
1849 0652 324607 STA CURCOL
1850 0655 3E17 MVI A,23 ; CURRENT ROW = 24
1851 0657 324707 STA CURROW
1852 SCLUP3:
1853 065A CD0802 CALL CUROFF
1854 065D 218007 LXI H,24*80
1855 0660 E5 PUSH H
1856 0661 CD5C01 CALL SETCUR ; READ AND SAVE LINE 25
1857 0664 CD9501 CALL RDLIN
1858 0667 E1 POP M
1859 0668 015000 LXI B,80
1860 0668 CD0E06 CALL CLEARSP ; CLEAR ERROR LINE
1861 066E CD2802 CALL SCROLL ; SCROLL WHOLE SCREEN
1862 0671 218007 LXI H,24*80
1863 0674 CD5C01 CALL SETCUR ; RESTORE LINE 25
1864 0677 CDC001 CALL WRLIN ; AND DISPLAY IT AGAIN
1865 067A CD1302 CALL CURON
1866 067D CD0E06 CALL WRCUPO ; SET CURSOR
1867 0680 C9 RET
1868
1869 PAGE

```

051 (PBIOS)

```

1870
1871 ;*****
1872 ;*
1873 ;* SCROLLDN MOVES SCREEN ONE LINE DOWN *
1874 ;* A - SPECIFIED LINE *
1875 ;*
1876 ;*****
1877
1878
1879 SCROLLDN:
1880 0681 F5 PUSH PSW ;
1881 0682 47 MOV B,A ;
1882 0683 3E17 MVI A,LINES-1 ;
1883 0685 90 SUB B ; A - NUMBER OF LINES TO MOVE
1884 0686 CA9A06 JZ SCLDN2 ; JUMP IF NO LINE TO MOVE
1885 0687 4F MOV C,A ;
1886 068A 0616 MVI B,LINES-2 ; START WITH LINE BEFORE LAST
1887 068C C00802 CALL CUROFF ; CURSOR OFF
1888
1889 068F CDA606 CALL MDROW ; MOVE ONE LINE DOWN
1890 0692 05 DCR B
1891 0693 00 DCR C
1892 0694 C28F06 JNZ SCLDN1 ; NEXT LINE
1893 0697 C01302 CALL CURON ; CURSOR ON
1894 ;
1895 SCLDN1:
1896 069A F1 POP PSW ; RESTORE LINE NUMBER
1897 069B C00000 CALL CLRLIN ; CLEAR THIS LINE
1898 069E AF XRA A
1899 069F 324607 STA CURCOL ; X - POS = 0
1900 06A2 C0ED06 CALL WRCUPO ; SET CURSOR
1901 06A5 C9 RET
1902
1903 PAGE

```

```

1904
1905 ;*****
1906 ;*
1907 ;* MDROW MOVE LINE B TO LINE B+1
1908 ;*
1909 ;*****
1910
1911 MDROW:
1912 06A6 C5 PUSH B ;
1913 06A7 D5 PUSH D ;
1914 06A8 E5 PUSH H ; LINE
1915 06A9 78 MOV A,B ; NUMBER TO A
1916 06AA 2600 MVI H,0
1917 06AC 2E50 MVI L,LWID
1918 06AE C00000 CALL MULT ; HL - LINE B * 80
1919 06B1 EB XCHG
1920 06B2 215000 LXI H,LWID
1921 06B5 19 DAD D ; HL = LINE B+1 DE = LINE B
1922 06B6 0E50 MVI C,LWID ; C - ÷ OF CHARACTERS / LINES
1923 MDROW1:
1924 06B8 EB XCHG
1925 06B9 C05C01 CALL SETCUR ; SET CURSOR
1926 06BC EB XCHG
1927 06BD C09501 CALL RDLIN ; GET CHARACTER AND ATTRIBUTE
1928 06C0 C05C01 CALL SETCUR ; SET CURSOR TO DESTINATION
1929 06C3 C0C001 CALL WRLIN ; WRITE CHARACTER AND ATTRIBUTE
1930 06C6 E1 POP H ;
1931 06C7 D1 POP D ;
1932 06C8 C1 POP B ; RESTORE REGS
1933 06C9 C9 RET
1934
1935
1936
1937 PAGE

```

053 (PBIOS)

```

1938
1939 ;*****
1940 ;*
1941 ;* MUROW MOVE LINE IN B+1 TO LINE B *
1942 ;*
1943 ;*****
1944
1945 MUROW:
1946 06CA C5 PUSH B ;
1947 06CB D5 PUSH D ;
1948 06CC E5 PUSH H ;
1949 06CD 78 MOV A,B ; LINE NUMBER TO A
1950 06CE 2600 MVI H,0
1951 06D0 2E50 MVI L,LWID
1952 06D2 CD0000 CALL MULT
1953 06D5 EB XCHG
1954 06D6 215000 LXI H,LWID
1955 06D9 19 DAD D ; HL = LINE B+1 DE = LINE B
1956 06DA 0E50 MVI C,LWID ; C - # OF CHARACTERS / LINE
1957 MUROW1:
1958 06DC CD5C01 CALL SETCUR ; SET CURSOR
1959 06DF EB XCHG
1960 06E0 CD9501 CALL RDLIN ; GET CHARACTER AND ATTRIBUTE
1961 06E3 CD5C01 CALL SETCUR ; SET CURSOR TO DESTINATION
1962 06E6 CDC001 CALL WRLIN ; WRITE CHARACTER
1963 06E9 E1 POP H ;
1964 06EA D1 POP D ;
1965 06EB C1 POP B ; RESTORE REGS
1966 06EC C9 RET
1967
1968 PAGE

```

054 (PBIOS)

```

1969
1970 ;*****
1971 ;*
1972 ;*   WRCUPO   WRITE CURRENT CURSOR POSITION
1973 ;*
1974 ;*****
1975
1976
1977 WRCUPO:
1978   06ED F5   PUSH   PSW
1979   06EE E5   PUSH   H
1980   06EF 2A4607 LHL   CURCOL   ;L = CURRENT COLUMN
1981                                     ;H = CURRENT ROW
1982   06F2 CDFB06 CALL   WRHLPOS  ;CALCULATE CURSOR POSITION
1983   06F5 C05C01 CALL   SETCUR   ;SET CURSOR
1984   06F8 E1   POP    H
1985   06F9 F1   POP    PSW
1986   06FA C9   RET
1987
1988
1989
1990 ;*****
1991 ;*
1992 ;*   WRHLPOS   COMPUTE ADDRESS WITHIN CRT-BUFFER
1993 ;*           H - LINE
1994 ;*           L - COLUMN
1995 ;*
1996 ;*****
1997
1998
1999 WRHLPOS:
2000   06FB F5   PUSH   PSW
2001   06FC E5   PUSH   H
2002   06FD 7C   MOV    A,H
2003   06FE 2600 MVI   H,0
2004   0700 2E50 MVI   L,LWID
2005   0702 C00000 CALL  MULT      ; HL - RELATIVE ADDRESS OF ROW
2006   0705 EB   XCHG
2007   0706 E1   POP    H
2008   0707 2600 MVI   H,0      ; L - COLUMN
2009   0709 19   DAD   D        ; HL - ROW * 80 + COLUMN
2010   070A F1   POP    PSW
2011   070B C9   RET
2012
2013
2014
2015
2016
2017 PAGE

```

055 (PBIOS)

```

2018
2019                                GDCIOM
2020      +      ;      GDC I/O MACRO 27/10/82 10:00 WF
2021      +      ;      =====
2022      +      OUTCMD:
2023 070C+D1807 1      CALL  FIFRDY      ;PUT A COMMAND TO GDC
2024 070F+D3A1 1      OUT    GDCCOM
2025 0711+C9    1      RET
2026      +      ;
2027      +      OUTPAR:
2028 0712+D1807 1      CALL  FIFRDY      ;PUT A PARAMETER TO GDC
2029 0715+D3A0 1      OUT    GDCCPAR
2030 0717+C9    1      RET
2031      +      ;
2032      +      FIFRDY:
2033 0718+F5    1      PUSH  PSW          ;CHECK AND WAIT FOR FIFO READY
2034      +      FIFO10:
2035 0719+D8A0 1      IN     GDCSTA      ;READ GDC STATUS
2036 071B+E602 1      ANI    FIFULL
2037 071D+C21907 1     JNZ    FIFO10      ;WAIT IF FIFO FULL
2038 0720+F1    1      POP    PSW
2039 0721+C9    1      RET
2040      +      ;
2041      +      CKDATR:
2042 0722+F5    1      PUSH  PSW          ;CHECK AND WAIT FOR DATA READY
2043      +      CKDA10:
2044 0723+D8A0 1      IN     GDCSTA      ;READ GDC STATUS
2045 0725+E601 1      ANI    DATRDY
2046 0727+CA2307 1     JZ     CKDA10      ;WAIT IF FIFO FULL
2047 072A+F1    1      POP    PSW
2048 072B+C9    1      RET
2049      +      ;
2050      +      INPAR:
2051 072C+D2207 1      CALL  CKDATR      ;READ FROM FIFO
2052 072F+D8A1 1      IN     FIFO
2053 0731+C9    1      RET
2054      +      ;
2055      +      SENPAR:
2056 0732+7E    1      MOV    A,M          ;LOAD PARAMETER
2057 0733+D1207 1      CALL  OUTPAR      ;PUT PARAMETER TO GDC
2058 0736+23    1      INX    H          ;POINTER TO NEXT PARAMETER
2059 0737+05    1      DCR    B          ;PARAMETER COUNTER - 1
2060 0738+C23207 1     JNZ    SENPAR      ;JUMP IF NOT AT END
2061 0738+C9    1      RET
2062      +      ;
2063      +      CHKDIP:
2064 073C+F5    1      PUSH  PSW          ;CHECK FOR DRAWING IN PROCESS
2065      +      CKDP10:
2066 073D+D8A0 1      IN     GDCSTA      ;READ GDC STATUS
2067 073F+E608 1      ANI    DRWINP
2068 0741+C23D07 1     JNZ    CKDP10      ;JUMP IF DRAWING IN PROCESS
2069 0744+F1    1      POP    PSW
2070 0745+C9    1      RET
2071      +      PAGE

```

056 (FBIOS)

```
2072 +
2073 + ;
2074 +      ENDM
2075
2076 ;*****
2077 ;*                                     *
2078 ;*          VIDEO DATA AREA          *
2079 ;*                                     *
2080 ;*****
2081
2082
2083 0746 00      CURCOL: DB      0      ; CURRENT COLUMN NUMBER
2084 0747 00      CURROW: DB      0      ; CURRENT ROW NUMBER
2085
2086 0748 =      TMPXY  EQU      $      ; TEMPORARY ROW/COL STORAGE
2087 0748 00      TMPCOL: DB      0      ; TEMPORARY STORAGE FOR COLUMN
2088 0749 00      TMPROW: DB      0      ; TEMPORARY STORAGE FOR ROW
2089
2090
2091 074A 0000     CURPOS: DW      0      ; CURSOR POSITION IN RAM
2092 074C 00      DRB:   DB      0      ; DATA REQUEST FLAG
2093 074D 00      ESCFLG: DB      0      ; ESCAPE FLAG
2094 074E 0000     DRQADR: DW      0      ; DATA REQUEST ADDRESS POINTER
2095 0750 E8      INVFLG: DB      0E8H   ; ATTRIBUTE BYTE
2096                                     ; BACKGROUND - BLACK
2097 0751 00      REVVID: DB      0      ; REVERSE VIDEO ON OFF FLAG 1=ON 0 = OFF
2098 0752 00      SCROW:  DB      0      ; SCROLLING ROW NUMBER
2099
2100 0753 00      YPOS:   DB      0      ; CURSOR CONTROL DATA
2101
2102 0754 00      COLORI: DB      0      ; COLOR INDEX BYTE 'C'-COLOR
2103
2104 0755          END
```

		057	(PB105)					
ALV0	FA80	181	182					
ALV1	FAE0	183	184					
ALV2	FB71	185	187					
ALV3	FBC2	187	189					
ALV4	FC13	189	191					
ALV5	FC64	191	193					
ALV6	FC85	193	195					
ALV7	FD06	195	197					
ALV8	FD57	197	199					
ALV9	FDA8	199	201					
BACKSP	055F	1076	1613					
BELL	0433	1074	1332					
BELL1	0443	1342	1344					
BELL2	043A	1340	1346					
BELLOUT	0439	1333	1338	1367	1372	1381		
BKSP1	057A	1616	1622	1626				
BKSP2	0575	1619	1623					
BS	0008	162						
CARRET	0559	1084	1601					
CCHAR	0048	518	838	846				
CHAMOD	0020	567						
CHKDIP	073C	2063						
CINIT	0000	103	267					
CKDA10	0723	2043	2046					
CKDATR	0722	2041	2051					
CKDP10	073D	2065	2068					
CLEAR	05A3	1088	1109	1111	1683			
CLEARSP	060E	860	1689	1725	1753	1770	1784	1860
CLREOL	05E8	1536	1728	1739				
CLREOS	058E	1117	1119	1707				
CLREOS1	05E2	1712	1727					
CLRLI1	0601	102	104	1765				
CLRLIN	0000	83	1841	1897				
CLRSCN	001A	166						
COAD	0026	233						
COLORI	0754	277	1200	1220	1260	1302	2102	
COLUMO	0622	1554	1602	1804				
COMPDEHL	0000	80	690	735				
COMTBL	02EA	940	1072					
COMIN	003A	103	321					
CONOUT	0048	104	338					
CONST	002C	103	304					
COTC	0027	234						
COUNTRY	0001	127						
CPMIRAM	0000	138						
CPMVER	0002	139						
CR	0000	160						
CRTOUT	00A7	342	343	366	368	410	592	
CRTTBL	0000	80	1002					
CSV0	FAA0	182	183					
CSV1	FB31	184	185					
CSV2	0000	186						
CSV3	0000	188						
CSV4	0000	190						
CSV5	0000	192						
CSV6	0000	194						

058 (PBLOS)

CSV7	0000	1964									
CSV8	0000	1984									
CSV9	0000	2004									
CURCOL	0746	102	634	902	906	914	920	1178	1427	1429	1437
		1483	1485	1549	1551	1614	1627	1670	1740	1806	1843
		1849	1899	1930	20834						
CURD	00E0	5274									
CURDISK	0004	1594									
CURHOM	0593	1092	16574								
CUROFF	0208	8374	1441	1494	1684	1724	1831	1853	1887		
CURON	0213	8454	1456	1510	1691	1726	1837	1865	1893		
CURPOS	074A	938	1048	20914							
CURROW	0747	911	921	923	1183	1398	1412	1569	1573	1617	1621
		1624	1640	1643	1708	1749	1851	20844			
CURS	0049	5214	742								
DATRDY	0001	5034	2045								
DCOMD	0051	2214									
DECHA1	04EF	14964	1509								
DECHA2	050A	1489	15124								
DECHAR	04DA	1131	14824								
DELIN	0482	1127	14114								
DESLCT	0000	84	271								
DIRBUF	FA00	1804	181								
DISFD	0007	2444									
DMACOM	0028	2354									
DMAEXC	0010	5074									
DMAMB	002A	2384									
DMANO	002B	2374									
DMAR	00A4	5504									
DMAREAD	0008	2414									
DMAST	0028	2364									
DMAW	0024	5544									
DMAWRT	0004	2404									
DWBRR	0001	1444									
DWBRS	0000	1454									
DP80	FE99	2024	203								
DP81	FEA8	2034	204								
DP82	FEB7	2044	205								
DP83	FEC6	2054	206								
DP84	FED5	2064	207								
DP85	FEE4	2074	208								
DP86	FEF3	2084	209								
DP87	FF02	2094	210								
DP88	FF11	2104	211								
DP89	FF20	2114									
DRQ	074C	102	598	1156	1181	1246	1319	1363	1383	20924	
DRQADR	074E	102	601	1158	1170	1248	1365	1374	20944		
DRWALL	0000	5764									
DRWIMP	0008	5064	2067								
DRWRET	0010	5774									
DSTAT	0050	2224									
DYHRAM	0004	5744									
EWFD	0003	2434									
EREOL	0522	1086	1121	1123	15364						
ESC	0018	1654									
ESCAPE	0203	597	10454								

	059	(PBIO5)																		
ESCFLG	074D	595	1046	1058	2093															
ESCTBL	030E	1050	1108																	
FDCRA	0051	223																		
FF	000C	163																		
FIFEMP	0004	505																		
FIFO	00A1	484	2052																	
FIF010	0719	2034	2037																	
FIFRDY	0718	2023	2028	2032																
FIFULL	0002	504	2036																	
FIGD	006C	525																		
FIGS	004C	524	669	706	769	796	827													
FINDAD	0295	959	968	1051																
FINDCH	028E	608	937																	
FSPACE	0528	1082	1548																	
FSPACE1	053A	1553	1556																	
FUKCCH	0000	80	1137																	
FWVER	0FF7	146	276																	
GCHRD	0068	526																		
GDCCOM	00A1	489	2024																	
GDCPAR	00A0	488	2029																	
GDCRES	0000	515																		
GDCSTA	00A0	483	2035	2044	2066															
GETFRQ	0461	1364	1370																	
GETLEN	046E	1373	1379																	
GETLPOS	0000	83	1003																	
GETX	0360	1169	1174																	
GETX1	0368	1177	1179																	
GETY	034C	1157	1162																	
GETY1	0357	1166	1168																	
GRAEMD	1FFF	496																		
GRAMOD	0002	566																		
HALFRES	0391	1115	1218																	
HALFSET	0378	1113	1198																	
HOME	0599	1658	1668	1690																
HORETR	0040	509																		
HSTBUF	F600	179	180																	
INCHA1	04A7	1442	1455																	
INCHA2	04C2	1433	1458																	
INCHAR	0488	1129	1426																	
INIT10	01E9	817	867	1685																
INLINE	0479	1125	1398																	
INLRPF	0008	570																		
INPAR	072C	679	681	782	2050															
INPBFF	0002	123	1345																	
INTLAC	0009	571																		
INVFLG	0750	640	718	1205	1207	1225	1227	1257	1259	1269	1290									
		1296	1305	1316	1465	1519	2095													
INVR1	039D	1222	1224																	
INVS1	0387	1202	1204																	
IOBSET	0000	81	268																	
IOBYTE	0003	158	269	282	305	322	339	359	380	405	425									
KBDAT	0001	126																		
KBDINIT	0000	83	270																	
KBELL	0041	120	1348																	
KCOUNT	0041	121																		
KEYBASE	0040	115	117	118	119	120	121													

060 (PERIODS)

KEYIN	0000	81	325	326	325	386					
KEYST	0000	81	308	309	430						
LANGUAGE	0000	83	398								
LF	000A	1614									
LFEED	0540	1555	15684	1589							
LFEED1	0550	1572	15764								
LGDAT	0030	1254									
LINBUF	F0F9	2014	202	779	805						
LINEFD	0553	1078	15884								
LINES	0018	5804	912	1571	1688	1710	1827	1840	1882	1886	
LIPDET	0030	5104									
LIST	0078	103	4034								
LISTST	0088	103	4244								
LOOKE	02A4	961	964	9694							
LF11	025F	8834									
LF12	0260	826	865	869	8844						
LF21	0263	8864									
LF22	0264	824	870	872	8874						
LPC	000A	4994									
LPRO	00C0	5284									
LWID	0050	5794	635	904	1431	1438	1487	1552	1625	1688	1688
		1719	1743	1769	1917	1920	1922	1951	1954	1956	2004
MASK	004A	5234	758								
MDROW	06A6	1889	19114								
MDROW1	0688	19234									
MIXGAC	0000	5654									
MOCOMP	0001	5424									
MODCRT	0002	104	612	6324							
MOREPL	0000	5414	655	714	803						
MORES	0002	5434									
MOSET	0003	5444									
MOTOROFF	0001	2254									
MOTORON	0014	2324									
MULT	0000	80	1720	1918	1952	2005					
MUR0W	06CA	1833	19454								
MUR0W1	06DC	19574									
MUSIC	044E	1135	13624								
MUSICC	0006	1284	1366								
MCRVER	0004	1434									
MCRVERH	0000	1424									
MOINTL	0000	5694									
MRWAPL	0048	4974	498								
NXTCR1	0272	905	9104								
NXTCR2	027F	913	9194								
NXTCUR	0265	613	9014								
OUTCMD	070C	656	670	678	707	715	743	759	770	778	797
		804	828	839	847	875	20224				
OUTPAR	0712	658	660	672	674	676	709	711	713	717	719
		745	747	749	761	762	772	774	776	799	801
		802	809	830	841	849	851	853	20274	2057	
OUTPBFF	0001	1244	1341								
PI0INIT	0000	80	288								
PITCH	0047	5224									
POPREG	00CE	102	6154	1059	1094	1141	1159	1171	1185	1208	1228
		1249	1320	1334	1368	1375	1384	1400	1414	1469	1523
		1537	1558	1590	1603	1629	1646	1659	1692	1729	

	061	(PB105)																		
PUSCUB	033E	1133	1154																	
PRAM	0070	530	874																	
PRAMSA	0000	531																		
PRTOUT	0000	82	367	411																
PRTSTAT	0000	82	431																	
PUNCH	0056	103	359																	
PUTCHR	00C4	609																		
RANSEL	0010	228	278																	
ROAT	00A0	546	677	777																
RDGCHR	00F3	668	1445	1499																
ROKEY	0040	118	1343																	
RDLIN	0195	766	1857	1927	1960															
RDLIN1	0184	781	786																	
READER	0068	103	379																	
RELEASE	0004	141																		
RELEASEH	0000	140																		
RESMOP	0000	562																		
REVERSE	03A7	1139	1244																	
REVVID	0751	1263	1267	1309	1313	2097														
RLFEED	0583	1080	1639																	
RLFEED1	0580	1642	1644																	
ROMSEL	0011	229	275																	
ROUT1	00A2	451	453																	
ROUTE	0098	102	285	306	323	340	364	383	408	428	445									
RSKEY	0041	119	1340	1344																
RVO	0385	1247	1253																	
RV1	03F9	1256	1293																	
RV2	03F1	1262	1289																	
RV3	0306	1268	1314																	
RV4	041B	1304	1308																	
RV5	0406	1295	1299																	
RVE	0429	1288	1292	1298	1307	1315														
RVE1	042C	1265	1301	1311	1317															
SACTIVE	0000	83	281																	
SATRM	0000	573																		
SCLDN1	068F	1888	1892																	
SCLDN2	069A	1884	1895																	
SCLUP1	0639	1832	1836																	
SCLUP2	0644	1829	1839																	
SCLUP3	065A	1577	1825	1852																
SCLUP4	0651	637	1847																	
SCROL1	024F	868	873	1686																
SCROLL	0228	857	1861																	
SCROLLDN	0681	1399	1879																	
SCROLLUP	062A	1413	1823																	
SCROW	0752	2098																		
SEMPAR	0732	878	2055	2060																
SETCUR	015C	728	1444	1449	1464	1498	1503	1518	1792	1856	1863									
		1925	1928	1958	1961	1983														
SETCUR1	0170	702	740																	
SETCUR2	016F	736	738																	
SETESC	02E2	1057	1090																	
SETMSK	0187	705	750	757																
SETTC	0012	230																		
SIOINIT	0000	80	286	287	289															
SF1	0250	731	820	861	864	877	882													

062 (FBIOS)

SP2	0261	821	885								
SPCLEAR1	0114	686	1793								
SPCLEAR2	0136	692	693	698	703						
SRLCOUT	0000	81	341	344	365						
SRLIN	0000	81	324	327	384	387					
SRLINST	0000	81	307	310							
SRLOUT	0000	82	409	412							
SRLSTAT	0000	82	429	432							
START	0068	519									
SUBHLDE	0000	82	696	737							
SYSSTA	0013	231									
TMPCOL	0748	102	2087								
TMPROW	0749	2088									
TPXY	0748	2086									
TOF	0000	164									
TRANS	0000	84	994								
TRNCHR	02A8	102	610	993	1461	1515	1788				
TRNCHR1	028C	1009	1017								
TRNCHR2	02C8	996	1021								
TRNSPC	0282	102	997								
TYHIBY	0018	539									
TYLOBY	0010	538									
TYWORD	0000	537	655	677	714	803					
VERETR	0020	508									
VID1	008C	600	605								
VSYNCH	006F	517									
VSYNCS	006E	516									
WDAT	0020	534	655	714	803						
WDKEY	0040	117									
WINIT	0016	103	279								
WPL	0024	498									
WRCUPO	06E0	104	924	1184	1468	1522	1557	1574	1628	1645	1671
		1794	1807	1844	1866	1900	1977				
WRGCHR	00E5	642	654	1450	1467	1504	1521				
WRHLPOS	06FB	1428	1440	1484	1717	1751	1768	1982	1999		
WRLIN	01C0	793	1864	1929	1962						
WRLIN1	01D0	807	812								
YPOS	0753	1167	1182	2100							
ZOOM	0046	520									

003 (QB IOS)

```

68
69 ;*****
70 ;*****
71 ;*****
72 ;***** EXTERNAL DEFINITIONS *****
73 ;*****
74 ;*****
75 ;*****
76 ;
77 ;
78 ;
79 EXTRN  POPREG,MODCRT,TRNSPC,DRQ,DRQADR,CURCOL,TRPCOL
80 EXTRN  CONFIGFL,KEYTBL,FUNTBL,FNTBLND,FNERRM
81 EXTRN  DISPERR,CONIN,CLRLIN,OFST
82 EXTRN  MOVE,ROUTE,SUBHLDE,COMPDEHL,MOVEL
83 EXTRN  M1RS232,M2RS232,PVRS232
84
85
86 PAGE
    
```

004 (QB IOS)

```

87
88
89
90 ;*****
91 ;*****
92 ;*****
93 ;***** ENTRY DEFINITIONS *****
94 ;*****
95 ;*****
96 ;*****
97 ;
98 ;
99 ;
100
101 PUBLIC LANGUAGE,KEYIN,KEYST,GETLPOS,FUNCCH,KBDINIT,TRANS
102
103 PUBLIC PRTSTAT,PRTOUT,PIOINIT
104
105 PUBLIC SRLIN,SRLINST,SRLOUT,SRLSTAT,SRLCOUT,SACTIVE,SIOINIT
106
107
108 PAGE
    
```

```

109
110 ;*****
111 ;*****
112 ;*****
113 ;*****          KDB ROUTINES          *****
114 ;*****          *****
115 ;*****
116 ;*****
117
118 KEYEQU
119 + ;*****
120 + ;*
121 + ;*          KEYBOARD EQUATES 27/10/82 10:27 WF
122 + ;*
123 + ;*****
124 0040+= KEYBASE EQU    00040H      ; BASE ADDRESS OF KEYBOARD
125 +
126 0040+= WKEY EQU     KEYBASE      ; WRITE DATA TO KEYBOARD
127 0040+= RKEY EQU     KEYBASE      ; READ DATA FROM KEYBOARD
128 0041+= RSKEY EQU    KEYBASE+1    ; READ KEYBOARD STATUS
129 0041+= KBELL EQU    KEYBASE+1    ; RING BELL
130 0041+= KCOUNT EQU   KEYBASE+1    ; COUNTRY CODE PORT
131 +
132 0002+= INPBUF EQU    02H         ; INPUTBUFFER FULL
133 0001+= OUTBUF EQU    01H         ; OUTPUTBUFFER FULL
134 0080+= LGDAT EQU     80H         ; LANGUAGE CODE READY FLAG
135 0001+= KBDAT EQU     01H         ; DATA READY BIT
136 0001+= COUNTRY EQU   01H         ; COMMAND TO GET COUNTRY CODE
137 0006+= MUSICC EQU    06H         ; MUSIC COMMAND
138 + ;
139 +          ENOM
140
141
142
143
144 0000 00  LANGUAGE:  DB    0      ; LANGUAGE CODE
145                                     ; 0 - US/HEBREW
146                                     ; 1 - UK
147                                     ; 2 - FRANCE
148                                     ; 3 - GERMANY
149                                     ; 4 - SWEDEN
150                                     ; 5 - DANSK
151                                     ; 6 - SPAIN
152                                     ; 7 - ITALY
153                                     ; 8 - SWISS
154                                     ; 9 - CANADA 1
155                                     ;10 - CANADA 2
156                                     ;11 - SOUTH AFRICA
157                                     ;12 - PORTUGAL
158                                     ;13 - YUGUSLAVIA
159
160
161 0001 00  TRANS:  DB    0          ;SOFTWARE SWITCH
162          PAGE

```

006 (QB10S)

163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205

Page

```

;*****
;#
;#   KBDINIT GET COUNTRY CODE
;#
;*****
    
```

```

KBDINIT:
    MVI   A,COUNTRY
    OUT  KCOUNT

KBDIN1:
    IN   RSKEY
    ANI  KBDAT
    JZ   KBDIN1      ; WAIT UNTIL A CHARACTER IS READY
    IN   RSKEY
    ANI  LGDAT
    JNZ  KBDIN2      ; LANGUAGE CODE READY
    IN   RDKEY
    JMP  KBDIN1      ; GET LANGUAGE CODE

KBDIN2:
    IN   RDKEY
    ; GET LANGUAGE CODE
    MOV  B,A
    ; SAVE LANGUAGE CODE
    CMA
    ANI  0FH
    ; MASK HIGH BYTE TO GET KBD VERSION
    RRC
    MOV  C,A
    MOV  A,B
    ; GET ORIGINAL VALUE
    ANI  07H
    ; MASK OUT LANGUAGE VERSION
    ADD  C
    ; COMBINE IT WITH KBD VERSION
    LXI  H,LTAB
    CALL OFST
    MOV  A,M
    ; GET LANGUAGE CODE OUT OF TABLE
    STA  LANGUAGE
    ; SET LANGUAGE CODE
    RET
    
```

PAGE

007 (QB10S)

206
207
208
209
210
211
212
213
214
215
216
217

```

;
;
;   LTAB TRANSLATES KBD/LANGUAGE VERSION TO PHYSICAL
;   TRANSLATION TABLE ENTRY
;
;
;*****
    DB   00H,01H,02H,03H,04H,05H,06H,07H ; KBD VERSION 1 OLD
    DB   00H,01H,05H,03H,04H,05H,06H,07H ; KBD VERSION 1 NEW
    DB   02H,02H,02H,09H,0AH,0BH,0CH,0CH ; KBD VERSION 2
    DB   00H,00H,00H,00H,00H,00H,00H,00H ; KBD VERSION 3
    
```

PAGE

008 (QB10S)

```

218
219 ;*****
220 ;#
221 ;* KEYST KEYBOARD SYSTEM STATUS #
222 ;# RETURN REGISTER A = 0FFH IF CHARACTER READY #
223 ;# A = 0 IF CHARACTER NOT READY #
224 ;#
225 ;*****
226
227
228 KEYST:
229 0050 3AE201 LDA FACTIVE
230 0053 B7 ORA A
231 0054 C0 RNZ ; RETURN IF FUNCTION KEY ACTIVE
232 0055 C0A500 CALL PKEYST ; GET KBD STATUS
233 0058 C9 RET
234
235
236
237 PAGE

```

009 (QB10S)

```

238
239 ;*****
240 ;#
241 ;* KEYIN KEYIN WITH EXPANDING FUNCTIONS #
242 ;#
243 ;*****
244
245
246 0059 3AE201 KEYIN: LDA FACTIVE
247 005C B7 ORA A
248 005D C27A00 JNZ KEYSY1 ; JUMP IF FUNCTION KEY ACTIVE
249 KEYSY2:
250 0060 C0AE00 CALL PKEYIN ; GET KEY BUFFER --> A
251 0063 FE80 CPI 80H
252 0065 D8 RC ; RETURN ASCII CHARACTER
253 0066 FEAD CPI 0A0H
254 0068 DA8200 JC KBDTRN ; JUMP IF CHARACTER BETWEEN 80H AND 9FH
255 006B 4F MOV C,A ; SAVE CHARACTER --> C
256 006C 3A0000 LDA CONFIGFL
257 006F B7 ORA A
258 0070 C28000 JNZ KEYSY4 ; JUMP IF FLAG SET (RET. FUNCT CHAR)
259 0073 79 MOV A,C ; RESTORE CHARACTER
260 0074 C0B700 KEYSY3: CALL FUNSET ; SET UP FUNCTION KEY HANDLING
261 0077 CA6000 JZ KEYSY2 ; JUMP IF AN INACTIVE FUNCTION
262 007A C0E400 KEYSY1: CALL FUNGETH ; GET NEXT CHAR
263 007D CCF400 CZ FUNCLOS ; DEACTIVATE FUNCTION KEY HANDLING
264 0080 79 KEYSY4: MOV A,C ; CHARACTER --> A
265 0081 C9 RET
266 PAGE

```

010 (QB IOS)

```

267
268 ;*****
269 ;*
270 ;* KBDTRN TRANSLATE KEY TO ASCII CHARACTER *
271 ;*
272 ;*****
273
274
275 KBDTRN:
276 0082 FE9E CPI 09EH
277 0084 DA9300 JC KBDTR1 ;JUMP IF = < THEN 9EH
278 0087 3EFF MVI A,OFFH
279 0089 CA8000 JZ KBDTR2 ;JUMP IF SOFTWARE SWITCH ON
280 008C AF XRA A ;SOFTWARE SWITCH OFF
281
282 KBDTR2:
283 008D 320100 STA TRANS ;SET SOFTWARE SWITCH
284 0090 C36000 JMP KEYSY2 ;GET NEXT CHARACTER
285
286 KBDTR1:
287 0093 4F MOV C,A
288 0094 C00000 CALL TRNSPC ; TRANSLATE CHARACTER
289 0097 79 MOV A,C
290 0098 FE80 CPI 80H
291 009A D8 RC ; RETURN IF TRANSLATED
292 009B E61F ANI 01FH ; CLEAR BIT 5,6,7
293 009D 210000 LXI H,KEYTBL ; POINT TO START OF TRANSLATE TABLE
294 00A0 C00000 CALL OFST ; HL = HL + A
295 00A3 7E MOV A,M
296 00A4 C9 RET
297 PAGE

```

011 (00105)

```

298
299 ;*****
300 ;*
301 ;* PKEYST GET KBD STATUS
302 ;* Z SET IF DATA READY
303 ;*
304 ;*****
305
306 PKEYST:
307 00A5 D841 IN RSKEY ; GET STATUS
308 00A7 E601 ANI KBDAT ;
309 00A9 C8 RZ ; NO DATA READY
310 00AA 3EFF MVI A,CFFH ;
311 00AC B7 ORA A ; DATA READY
312 00AD C9 RET
313
314
315 ;*****
316 ;*
317 ;* PKEYIN GET KEYBOARD CHARACTER
318 ;* A - CHARACTER
319 ;*
320 ;*****
321
322
323 PKEYIN:
324 00AE C0A500 CALL PKEYST ; TEST KEYBOARD STATUS
325 00B1 CAAE00 JZ PKEYIN ; WAIT UNTIL A CHARACTER READY
326 00B4 D840 IN RDKY ; GET KBD-CHARACTER
327 00B6 C9 RET
328
329 PAGE

```

012 (QBIOS)

```

330
331 ;*****
332 ;*****
333 ;*****
334 ;***** FUNCTION KEY HANDLING *****
335 ;*****
336 ;*****
337 ;*****
338 ;*****
339 ;*
340 ;* FUNSET SETUP FUNCTION KEY RUE STRINGS RETURN WITH *
341 ;* FCHARCNT, THE CHAR COUNT. AND FPOINTER *
342 ;* THE POINTER INTO FUNCTION STRING. *
343 ;* FUNCTION ARE DEFINED AS: *
344 ;* A0 - B3 *
345 ;* C0 - D3 *
346 ;* E0 - F3 *
347 ;*
348 ;*
349 ;*****
350
351
352 FUNSET:
353 0087 E61F ANI 01FH ; MASK FUNCTION NUMBER
354 0089 FE14 F34F CPI 20
355 008B D2D400 JNC FUNSET1 ; JUMP IF FUNCTION CODE > 20
356 00BE CDD600 CALL GETFPOS ; GET POSITION OF FUNCTION
357 00C1 7E MOV A,M ; CHAR COUNT IN [A]
358 00C2 3D DCR A ; CORRECT FUNCTION LENGTH
359 00C3 CAD400 JZ FUNSET1 ; JUMP IF FUNCTION LENGTH = 0
360 00C6 23 INX H ; POINT TO 1ST CHARACTER
361 00C7 22E401 SHLD FPOINTER ; SAVE AS POINTER TO BEGING OF STRING
362 00CA 32E301 STA FCHARCNT ; SAVE [A] AS CHAR COUNT
363 00CD 3EFF MVI A,OFFH
364 00CF 32E201 STA FACTIVE ; INDICATOR THAT FUN IS ACT
365 00D2 87 ORA A ; INDICATE ACTIVE FUNCTION
366 00D3 C9 RET
367
368 FUNSET1:
369 00D4 AF XRA A ; RETURN INVALID FUNCTION
370 00D5 C9 RET
371 PAGE

```

```

372
373 ;*****
374 ;*
375 ;* GETFPOS  SET POSITION OF REQUESTED FUNCTION
376 ;*          A - FUNCTION #
377 ;*
378 ;*          HL - ADDRESS OF FUNCTION
379 ;*****
380 ;
381 ;
382 GETFPOS:
383 0006 210000      LXI    H,FUNTBEL      ; START OF FUNCTION TABLE
384
385 ;*****
386 ;*
387 ;* GETLPOS  GET POSITION OF REQUESTED TABLE ENTRY
388 ;*          Entry:  A - TABLE ENTRY NUMBER
389 ;*                  HL - TABLE START ADDRESS
390 ;*
391 ;*          Exit:   HL - ADDRESS IN TABLE
392 ;*****
393 ;
394 GETLPOS:
395 0009 47          MOV    B,A           ;
396 00DA 04          INR    B
397 0008 05      GETF1: DCR    B           ; DECREMENT TABLE ENTRY COUNTER
398 00DC C8          RZ                   ; RETURN IF FUNCTION REACHED
399 00DD 7E          MOV    A,M           ; GET FUNCTION LENGTH
400 00DE C0000      CALL   OFST          ; ADD FUNCTION LENGTH TO ADDRESS
401 00E1 C30B00      JMP    GETF1
402                PAGE

```

014 (QBIOS)

```

403
404 ;*****
405 ;*
406 ;* FUNGETN GET NEXT CHAR FROM THE FUNCTION TABLE *
407 ;* RETURNS CHAR IN [C] , COUNT REMAINING IN [A] *
408 ;*
409 ;*****
410
411
412 FUNGETN:
413 00E4 2AE401    LJLD    FPOINTER    ; FPOINTER POINTS TO NEXT CHAR
414 00E7 4E        MOV     C,M          ; GET CHAR TO [C]
415 00E8 23        INX     H            ; BUMP POINTER
416 00E9 22E401    SHLD   FPOINTER    ; SAVE BUMPED POINTER
417 00EC 3AE301    LDA    FCHARCNT    ; GET CHAR COUNT
418 00EF 3D        DCR     A           ; DEC TO ZERO
419 00F0 32E301    STA    FCHARCNT    ; SAVE OLD CHAR
420 00F3 C9        RET              ; RETURN WITH COUNT FLAGS SET
421
422
423 ;*****
424 ;*
425 ;* FUNCLOS DEACTIVATE FUNCTION KEY HANDLING *
426 ;*
427 ;*****
428
429
430 FUNCLOS:
431 00F4 AF        XRA    A
432 00F5 32E201    STA    FACTIVE    ; RESET FUNCTION KEY ACTIVE FLAG
433 00F8 C9        RET
434 PAGE

```

015 (QBIOS)

```

435
436 ;*****
437 ;*
438 ;* FUNCCH  CHANGE FUNCTION VALUE *
439 ;*
440 ;*****
441 ;
442 ;
443 FUNCCH:
444 00F9 3EFF      MVI    A,OFFH
445 00FB 320000    STA    DRQ        ; SET DATA REQUEST BYTE
446 00FE 32EE01    STA    FNERR      ; SET NO ERROR OCCURED BYTE
447 0101 210A01    LXI    H,GETFCHR
448 0104 220000    SHLD   DRQADR     ; SET ADDRESS OF 'GET FUNC. CHR' ROUTINE
449 0107 C30000    JMP    POPREG
450
451
452
453 PAGE

```

```

454
455 ;*****
456 ;*
457 ;* GETFCHR GET FUN. CHR. WHICH SHOULD CHANGED
458 ;*
459 ;*****
460
461
462 GETFCHR:
463 010A 79      MOV    A,C
464 010B 32E601  STA    FNCCHAR ; SAVE FUNCTION
465 010E E61F    ANI    1FH
466 0110 C00600  CALL   GETFPOS
467 0113 7E     MOV    A,M ; GET FUNCTION LENGTH
468 0114 23     INX    H
469 0115 22E701 SHLD   FNSTR ; SET START ADDRESS OF NEW FUNC.
470 0118 22E901 SHLD   FNACT ; SET ACTUAL FNC-ADDRESS
471 011B C00000 CALL   OFST ; COMPUTE ADDRESS OF NEXT FUNCTION
472 011E 2B     DCX    H
473 011F C00701 CALL   GFNLN ; COMPUTE REFSIZE OF FOLLOWING FUNC.
474 0122 22E801 SHLD   RSTLEN ; SAVE RESTLENGTH
475 0125 44     MOV    B,H
476 0126 4D     MOV    C,L
477 0127 2AE701 LHLD   FNSTR ; GET FUNCTION START ADDRESS
478 012A EB     XCHG
479 012B C00000 CALL   MOVE ; MOVE REST OF FUNC. TO START OF NEW
480 012E EB     XCHG
481 012F 2B     DCX    H ; LAST FUNCTION CHARACTER
482 0130 22EF01 SHLD   FNEND ; SAVE LAST CHARACTER ADDRESS
483 0133 3E01   MVI    A,1
484 0135 32ED01 STA    FNLEN ; SET LENGTH OF FUNC = 0
485 0138 214101 LXI    H,CHCHR
486 013B 220000 SHLD   DRQADR ; SET ADDRESS OF 'GET NEW FUNC. CHR.'
487 013E C30000 JMP    POPREG
488 PAGE

```

017 (QB IOS)

```

489
490 ;*****
491 ;*
492 ;* CHCHR CHANGE ONE CHARACTER UND UPDATE POINTERS *
493 ;*
494 ;*****
495
496
497 CHCHR:
498 0141 3AE601 LDA FNCCHAR
499 0144 B9 CMP C
500 0145 CA7E01 JZ CHCHEND ; JUMP IF END OF FUNC. CHANGE
501 0148 3AEEO1 LDA FNERR
502 014B B7 ORA A
503 014C CA0000 JZ POPREG ; DOESN'T ACCEPT CHARACTER (ERROR)
504 014F C5 PUSH B ; SAVE NEW CHARACTER
505 0150 21ED01 LXI H, FNLEN
506 0153 34 INR M ; FUNCTION LENGTH + 1
507 0154 21E801 LXI H, RSTLEN
508 0157 4E MOV C, M
509 0158 23 INX H
510 0159 46 MOV B, M ; BC = RESTLENGTH
511 015A 2AEF01 LHLD FNEND
512 015D 23 INX H
513 015E 22EF01 SHLD FNEND ; SET NEW END ADDRESS
514 0161 EB XCHG
515 0162 210000 LXI H, FMTBLND ; TABLE END
516 0165 C00000 CALL COMPDEHL
517 0168 CA8801 JZ FUNERR ; JUMP IF END OF TABLE REACHED
518 016B 2AEF01 LHLD FNEND ; ACTUAL POSITION
519 016E 2B DCX H ; START OF MOVE
520 016F C00000 CALL MOVE1 ; MOVE REST OF FUNC ONE CHAR
521 0172 C1 POP B
522 0173 2AE901 LHLD FNACT
523 0176 71 MOV M, C ; SET NEW CHARACTER INTO FUNC. TABLE
524 0177 23 INX H
525 0178 22E901 SHLD FNACT ; SET NEW ACTUAL FN ADDRESS
526 017B C30000 JMP POPREG
527 PAGE

```


018 (QB10S)

```

528
529 ;*****
530 ;*
531 ;* CHCHEMD LAST FUNCTION CHAR ENTERED CLOSE FUNC. CHANGE *
532 ;*
533 ;*****
534
535
536 CHCHEMD:
537 017E 3AEE01 LDA FNERR
538 0181 87 ORA A
539 0182 CA9401 JZ FNERRD ; DISPLAY ERROR MESSAGE
540 0185 3AED01 LDA FNLEN
541 0188 2AE701 LHLD FNSTR ;
542 018B 28 DCX H
543 018C 77 MOV M,A ; SET FUNCTION LENGTH
544 018D AF XRA A
545 018E 320000 STA DRQ ; RESET REQUEST DATA BYTE
546 0191 C30G00 JMP POPREG
547 PAGE

```

019 (QB10S)

```

548
549 ;*****
550 ;*
551 ;* FNERRD DISPLAY ERROR MESSAGE IF FUNCTION TABLE END *
552 ;* REACHED *
553 ;*
554 ;*****
555
556
557
558 FNERRD:
559 0194 AF XRA A
560 0195 320000 STA DRQ ; RESET DATA REQUEST BYTE
561 0198 2A0000 LHLD CURCOL
562 019B 220000 SHLD TMPCOL ; SAVE ORIGINAL CURSOR
563 019E 210018 FNERR1: LXI H,01800H
564 01A1 220000 SHLD CURCOL
565 01A4 210000 LXI H,FNERRM
566 01A7 C00000 CALL DISPERR ; DISPLAY ERROR MESSAGE
567 01AA C00000 CALL CONIN ; INPUT A CHARACTER
568 01AD 2A0000 LHLD TMPCOL
569 01B0 220000 SHLD CURCOL
570 01B3 3E18 MVI A,24
571 01B5 C00000 CALL CLRLLN ; CLEAR ERROR LINE
572 01B8 C30000 JMP POPREG
573
574
575
576 PAGE

```

020 (QBIOS)

```

577
578 ;*****
579 ;*
580 ;* FUNERR      FUNCTION TO LONG (END OF FNTABLE REACHED) *
581 ;*
582 ;*****
583
584
585
586
587 FUNERR:
588 0188 C1      POP      B          ; CLEAR STACK
589 018C AF      XRA      A
590 018D 32EE01  STA      FNERR      ; SET ERROR FLAG
591 01C0 21EB01  LXI      H,RSTLEN
592 01C3 4E      MOV      C,M
593 01C4 23      INX      H
594 01C5 46      MOV      B,M          ; BC = RESTLENGTH
595 01C6 2AE701  LHL     FNSTR
596 01C9 2B      DCX      H
597 01CA 3601    MVI      M,1          ; SET FUNCTION LENGTH = 0
598 01CC 23      INX      H
599 01CD EB      XCHG
600 01CE 2AE901  LHL     FNACT          ; HL = NXT FNCT , DE = STRT OF NEW FNCT
601 01D1 C00000  CALL    MOVE          ; REMOVE REST OF FUNCTIONS
602 01D4 C30000  JMP     POPREG
603
604
605
606 ;*****
607 ;*
608 ;* GFNLN      COMPUTE REST OF FUNCTION LENGTH *
609 ;*
610 ;*****
611
612
613
614 GFNLN:
615 01D7 E5      PUSH    H
616 01D8 3E14    MVI    A,20
617 01DA C0D600  CALL   GETFPOS        ; COMPUTE FUNCTION TABLE END ADDRESS
618 01E0 23      INX    H
619 01E2 D1      POP    D
620 01E4 C30000  JMP    SUBHLDE
621
622
623
624 PAGE

```

```

625
626 ;*****
627 ;*
628 ;* : : : FUNCTION-KEY DATA AREA
629 ;*
630 ;*****
631
632
633 01E2 00 FACTIVE: DB 0 ; FUNCTION EXPANSION ACTIVE FLAG
634 01E3 00 FCHARCNT: DB 0 ; FUNCTION CHAR COUNT
635 01E4 0000 FPOINTER: DW 0 ; POINTER INTO CHARSTRING
636 01E6 00 FMCCHAR: DB 0 ; FUNCTION CHARACTER
637 01E7 0000 FNSTR: DW 0 ; START OF OLD FUNCTION
638 01E9 0000 FMACT: DW 0 ; ACTUAL ADDRESS
639 01EB 0000 RSTLEN: DW 0 ; RESTLENGTH OF FOLLOWING FUNCTIONS
640 01ED 00 FNLEN: DB 0 ; LENGTH OF NEW FUNCTION
641 01EE 00 FNERR: DB 0 ; ERROR FLAG
642 01EF 0000 FMEND: DW 0 ; ACTUAL END ADDRESS WITHIN TABLE
643
644
645
646 PAGE

```

022 (QEIOS)

```

647
648 ;*****
649 ;*****
650 ;*****
651 ;***** SERIAL INTERFACE (RS232) *****
652 ;*****
653 ;*****
654 ;*****
655
656
657 R232PAEQ
658 + ;
659 + ; PORT ADDRESSES FOR RS232 (2651) 28.10.82 PH
660 + ;
661 0060+= SPRDATA EQU 60H ; READ DATA
662 0061+= SPRSTAT EQU 61H ; READ STATUS
663 0063+= SPRCOM EQU 63H ; READ COMMAND
664 0064+= SPWDATA EQU 64H ; WRITE DATA
665 0066+= SPWMODE EQU 66H ; WRITE MODE
666 0067+= SPWCOM EQU 67H ; WRITE COMMAND
667 + ;
668 + ENDM
669 R232STEQ
670 + ;
671 + ; STATUS EQUATES FOR RS232 (2651) 28.10.82 PH
672 + ;
673 0001+= TXRDY EQU 0000001B ; TRANSMIT HOLDING REGISTER EMPTY
674 0002+= RXRDY EQU 0000010B ; RECEIVE HOLDING REGISTER EMPTY
675 0004+= TXENT EQU 0000100B ; CHANGE IN DSR OR DCD OR TRANSMIT
676 0008+= PARITY EQU 0001000B ; PARITY ERROR
677 0010+= OVERRUN EQU 0010000B ; OVERRUN ERROR
678 0020+= FRAMING EQU 0010000B ; FRAMING ERROR
679 0040+= DCD EQU 0100000B ; DATA CARRIER DETECT
680 0080+= DSR EQU 1000000B ; DATA SET READY
681 + ;
682 + ENDM
683
684
685 0011 = XON EQU 11H ; XON VALUE
686 0013 = XOFF EQU 13H ; XOFF VALUE
687 PAGE

```

023 (QB IOS)

```

688
689 ;*****
690 ;*
691 ;*      SIOINIT INITIALIZE THE SERIAL I/O
692 ;*
693 ;*****
694
695
696 SIOINIT:
707 707 01F1 3A0000      LDA      MIRS232  E010
708 708 01F4 0366        OUT      SPWMODE      ; OUT MODE 1 BYTE      = 66H
709 709 01F6 3A0000      LDA      M2RS232  E011
710 710 01F9 0366        OUT      SPWMODE      ; OUT MODE 2 BYTE
711 711 01FB 3E37        MVI      A,37H
712 712 01FD 0367        OUT      SPWCOM      ; ENABLE TRANSMITTER AND RECEIVER
713 713                ; SET DTR AND RTS , RESET ERROR
714
715 705 01FF 21FF00      LXI      H,000FFH
716 706 0202 22A402      SHLD    SACTIVE 75EE ; ENABLE SERIAL INTERFACE
717 707                ; DISABLE PARALLEL INTERFACE
718 708 0205 C9         RET
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739

```

024 (QB IOS)

```

714
715 ;*****
716 ;*
717 ;*      SERIAL PROTOCOL HANDLER FOR FUTURE EXPANSION
718 ;*
719 ;*****
720
721
722 SRLOUT:
723 723 0206 3A0000      LDA      PVRS232      ; GET PROTOCOL VECTOR,
724 724 0209 C00000      CALL    ROUTE        ; AND DISPATCH APPROPRIATELY
725 725 020C 7102        DW      SRLCOUT
726 726 020E 7102        DW      SRLCOUT
727 727 0210 7102        DW      SRLCOUT
728 728 0212 7102        DW      SRLCOUT      ; DUMMY FOR INTERRUPT SERVICE ON
729
730
731 SRLSTAT:
732 732 0214 3A0000      LDA      PVRS232      ; GET PROTOCOL VECTOR,
733 733 0217 C00000      CALL    ROUTE        ; AND DISPATCH
734 734 021A 4202        DW      SRLPRST
735 735 021C 4202        DW      SRLPRST
736 736 021E 4202        DW      SRLPRST
737 737 0220 4202        DW      SRLPRST      ; DUMMY FOR INTERRUPT SERVICE ON
738
739

```

025 (QB IOS)

```

740
741 ;*****
742 ;*                                     *
743 ;*          XON/XOFF SERIAL PROTOCOL          *
744 ;*                                     *
745 ;*****
746
747
748
749 ;*****
750 ;*                                     *
751 ;*          SRLINST GET PRINTER INPUT STATUS          *
752 ;*          Z - SET IF NO CHARACTER RECEIVED          *
753 ;*          OFFH - IF A CHARACTER RECEIVED          *
754 ;*                                     *
755 ;*****
756
757
758
759 SRLINST:
760 0222 3AA402      LDA      SACTIVE
761 0225 B7          ORA      A
762 0226 CCF101     CZ        SIOINIT          ; INIT SERIAL INTERFACE IF REQUIRED
763 0229 D861      IN        SPRSTAT
764 022B E638      ANI       OVERRUN OR PARITY OR FRAMING
765 022D C43902   CNZ       TRERR          ; CALL IF ERROR IN RECEIVER
766 0230 D861      IN        SPRSTAT          ; GET INPUT STATUS
767 0232 E602      ANI       RXRDY
768 0234 C8        RZ        ; RETURN IF NO CHARACTER RECEIVED
769 0235 3EFF     MVI       A,OFFH
770 0237 B7        ORA      A          ; RETURN CHAR. RECEIVED STATUS
771 0238 C9        RET
772
773 TRERR:
774 0239 D860      IN        SPRDATA          ; DUMMY READ
775 023B D863      IN        SPRCOM          ; READ COMMAND BYTE
776 023D F610     ORI       10H          ; RESET ERROR
777 023F D367     OUT       SPWCOM          ; RESET ERROR
778 0241 C9        RET
779
780
781 PAGE

```

782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812

```

;*****
;*
;*      SRLPRST      GET PRINTER STATUS
;*                      ZERO SET PRINTER NOT READY
;*
;*****

```

F58d

```

SRLPRST:
LDA      SACTIVE      F5EE
ORA      A
CZ      SIOINIT      F5JB ; INIT SERIAL INTERFACE IF REQUIRED
CALL    SRLINST      ; CHECK INPUT STATUS
JZ      SP1          ; JUMP IF NO INPUT
CALL    SRLIN        ; GET INPUT CHARACTER
STA      XOFFFLG     ; SAVE FLAG
SP1:    LDA      XOFFFLG
CPI      XOFF
JZ      SP2          ; JUMP IF XOFF FLAG SET PRINTER NOT READY
IN      SPRSTAT
ANI      TXRDY
RZ      ; RETURN IF TRANSMITTER NOT READY
MVI     A,OFFH
ORA      A          ; TRANSMITTER READY
RET

SP2:    XRA      A          ; PRINTER NOT READY
RET
PAGE

```

027 (QB10S)

```

813
814
815
816 ;*****
817 ;*
818 ;*   SRLIN   GET CHARACTER FROM INTERFACE
819 ;*
820 ;*****
821
822
823 SRLIN:
824 0268 CD2202      CALL   SRLINST      ; CHECK INPUT STATUS
825 0268 CA6802      JZ     SRLIN        ; WAIT ON INPUT CHARACTER
826 026E D860       IN     SPRDATA     ; GET CHARACTER
827 0270 C9         RET
828
829
830
831 ;*****
832 ;*
833 ;*   SRLCOUT  OUTPUT CHARACTER
834 ;*           [C] - OUTPUT CHARACTER
835 ;*
836 ;*****
837
838
839
840
841 SRLCOUT:
842 0271 CD4202      CALL   SRLPRST 758c' ; CHECK OUTPUT STATUS
843 0274 CA7102      JZ     SRLCOUT     ; WAIT UNTIL READY TO TRANSMIT
844 0277 79         MOV    A,C         ; GET CHARACTER
845 0278 D364       OUT    SPWDATA
846 027A C9         RET
847
848 PAGE

```


028 (QB10S)

```

849
850
851 ;*****
852 ;*****
853 ;*****
854 ;***** PARALLEL INTERFACE (CENTRONICS) *****
855 ;*****
856 ;*****
857 ;*****
858
859
860 0060 = PBDA EQU 60H ; DATA PORT
861 0061 = PBSTA EQU 61H ; STATUS PORT
862 0063 = PBCOM EQU 63H ; CONTROL PORT
863
864
865
866 0020 = BUSY EQU 20H ; PRINTER BUSY
867 0002 = POBF EQU 02H ; OUTPUT BUFFER FULL
868
869 PAGE

```

029 (QB10S)

```

870
871
872 ;*****
873 ;*
874 ;* PIOINIT INIT PARALLEL INTERFACE
875 ;*
876 ;*****
877
878
879 PIOINIT:
880 0278 3EAA MVI A,0AAH
881 027D 0363 OUT PBCOM ; INIT INTERFACE
882
883 027F 2100FF LXI H,OFF00H
884 0282 22A402 SHLD SACTIVE ; ENABLE PARALLEL INTERFACE
885 ; DISABLE SERIAL INTERFACE
886 0285 C9 RET
887
888 PAGE

```

030 (QB10S)

```

889
890 ;*****
891 ;*
892 ;* PRTOUT OUTPUT CHARACTER
893 ;* [C] - OUTPUT CHARACTER
894 ;*
895 ;*****
896
897
898 PRTOUT:
75D0 899 0286 C09002 CALL PRTSTAT ; CHECK INTERFACE STATUS
900 0289 CA8602 JZ PRTOUT ; JUMP IF PRINTER NOT READY
75D7 901 028C 79 MOV A,C ; OUTPUT CHARACTER -> A
902 028D D360 OUT PBDA ; OUT CHARACTER
903 028F C9 RET
904
905
906
907 ;*****
908 ;*
909 ;* PRTSTAT GET PRINTER STATUS
910 ;* ZERO SET PRINTER NOT READY
911 ;*
912 ;*****
913
914
915 PRTSTAT:
75D8 916 0290 3AA502 LDA PACTIVE
917 0293 B7 ORA A
918 0294 CC7B02 CZ PIOINIT ; INIT PARALLEL INTERFACE IF REQUIRED
75E1 919 0297 0861 IN PBSTA
920 0299 E622 ANI BUSY OR POBF
921 029B CAA002 JZ P1STATX ; JUMP IF PRINTER READY TO RECEIVE A BYTE
922 029E AF XRA A ; PRINTER NOT READY
923 029F C9 RET
924
925
926 P1STATX:
75EA 927 02A0 3EFF MVI A,OFFH
75ED 928 02A2 B7 ORA A ; PRINTER READY
929 02A3 C9 RET
930
931 PAGE

```

031 (QB10S)

```

932
933
934 ;*****
935 ;*
936 ;* PRINTER DATA AREA
937 ;*
938 ;*****
939
940
941
75EE 942 02A4 00 SACTIVE:DB 0 ; SERIAL INTERFACE ACTIVE FLAG
943 02A5 00 PACTIVE:DB 0 ; PARALLEL INTERFACE ACTIVE FLAG
944
945
946 02A6 00 XOFFFLG:DB 0 ; XOFF FLAG
947 02A7 END

```

032 (QB10S)

BUSY	0020	866‡	920				
CHCHEND	017E	500	536‡				
CHCHR	0141	485	497‡				
CLRLIN	0000	81	571				
COMPDEHL	0000	82	516				
CONFIGFL	0000	80	256				
CONIN	0000	81	567				
COUNTRY	0001	136‡	176				
CURCOL	0000	79	561	564	569		
DCD	0040	679‡					
DISPERR	0000	81	566				
DRQ	0000	79	445	545	560		
DRQADR	0000	79	448	486			
DSR	0080	680‡					
FACTIVE	01E2	229	246	364	432	633‡	
FCHARCNT	01E3	362	417	419	634‡		
FHACT	01E9	470	522	525	600	638‡	
FHCCHAR	01E6	464	498	636‡			
FNEND	01EF	482	511	513	518	642‡	
FNERR	01EE	446	501	537	590	641‡	
FNERR1	019E	563‡					
FNERRD	0194	539	558‡				
FNERRM	0000	80	565				
FNLEN	01ED	484	505	540	640‡		
FNSTR	01E7	469	477	541	595	637‡	
FNTBLMD	0000	80	515				
FPOINTER	01E4	361	413	416	635‡		
FRAMING	0020	678‡	764				
FUNCCH	00F9	101	443‡				
FUNCLOS	00F4	263	430‡				
FUNERR	0188	517	587‡				
FUNGETN	00E4	262	412‡				
FUNSET	0087	260	352‡				
FUNSET1	0004	355	359	368‡			
FUNTEL	0000	80	383				
GETF1	0008	397‡	401				
GETFCHR	010A	447	462‡				
GETFPOS	0006	356	382‡	466	617		
GETLPOS	0009	101	394‡				
GFMLN	0107	473	614‡				
IMPBFF	0002	132‡					
KBOAT	0001	135‡	180	308			
KBDIN1	0006	178‡	181	186			
KBDIN2	0019	184	187‡				
KBDINIT	0002	101	175‡				
KBDTR1	0093	277	285‡				
KBDTR2	0080	279	281‡				
KBDTRN	0082	254	275‡				
KBELL	0041	129‡					
KCOUNT	0041	130‡	177				
KEYBASE	0040	124‡	126	127	128	129	130
KEYIN	0059	101	246‡				
KEYST	0050	101	228‡				
KEYSY1	007A	248	262‡				

		033	(QB105)							
KEYSY2	0060	249	261	283						
KEYSY3	0074	260								
KEYSY4	0080	258	264							
KEYTBL	0000	80	292							
LANGUAGE	0000	101	144	200						
LGDAT	0080	134	183							
LTAB	0030	197	213							
M1RS232	0000	83	697							
M2RS232	0000	83	699							
MOOCRT	0000	79								
MOVE	0000	82	479	601						
MOVEL	0000	82	520							
MUSICC	0006	137								
OFST	0000	81	198	293	400	471				
OUTPBFF	0001	133								
OVERRUN	0010	677	764							
P1STATX	02A0	921	926							
PACTIVE	02A5	916	943							
PARITY	0008	676	764							
PBCOM	0063	862	881							
PBDA	0060	860	902							
PBSTA	0061	861	919							
PIOINIT	0278	103	879	918						
PKEYIN	00AE	250	323	325						
PKEYST	00A5	232	306	324						
POBF	0002	867	920							
POPREG	0000	79	449	487	503	526	546	572	602	
PRTOUT	0286	103	898	900						
PRTSTAT	0290	103	899	915						
PVRS232	0000	83	723	732						
RKEY	0040	127	185	188	326					
ROUTE	0000	82	724	733						
RSKEY	0041	128	179	182	307					
RSTLEN	01EB	474	507	591	639					
RXROY	0002	674	767							
SACTIVE	02A4	105	706	760	792	884	942			
SIOINIT	01F1	105	696	762	794					
SP1	0255	796	799							
SP2	0266	801	809							
SPRCOM	0063	663	775							
SPRDATA	0060	661	774	826						
SPRSTAT	0061	662	763	766	802					
SPWCOM	0067	666	702	777						
SPWDATA	0064	664	845							
SPWMODE	0066	665	698	700						
SRLCOUT	0271	105	725	726	727	728	841	843		
SRLIN	0268	105	797	823	825					
SRLINST	0222	105	759	795	824					
SRLOUT	0206	105	722							
SRLPRST	0242	734	735	736	737	791	842			
SRLSTAT	0214	105	731							
SUHLDE	0000	82	620							
TMPCOL	0000	79	562	568						
TRANS	0001	101	161	282						

		034	(0810S)	
TRERR	0239	765	773	
TRNSPC	0000	79	287	
TXENT	0004	675		
TXRDY	0001	673	803	
WQKEY	0040	126		
XOFF	0013	686	800	
XOFFLG	02A6	798	799	946
XOM	0011	685		

003 (RBIOS)

```

66
67 ;*****
68 ;*****
69 ;*****
70 ;***** EXTERNAL DEFINITIONS *****
71 ;*****
72 ;*****
73 ;*****
74 ;
75 ;
76 ;
77
78
79 EXTRN HSTDSK,HSTTRK,HSTSEC
80 EXTRN MULT,PMAADR,NERFLEX,DSKCMD
81 EXTRN ERRBUF
82 PAGE
    
```

004 (RBIOS)

```

83
84 ;*****
85 ;*****
86 ;*****
87 ;***** ENTRY DEFINITIONS *****
88 ;*****
89 ;*****
90 ;*****
91 ;
92 ;
93 ;
94
95
96 PUBLIC WINDRW,WINDRE,WINDFM,DESLCT
97
98
99
100 PAGE
    
```

```

101
102
103 ; WINCHESTER DISK DEFINITIONS
104 ; =====
105 ;
106 ;
107 00C0 = HBASE EQU 0C0H ;CONTROLLER BASE ADDR.
108 00C0 = DATA EQU HBASE ; R/W DATA REGISTER
109 00C1 = ERROR EQU HBASE+1 ; R ERROR REGISTER
110 00C1 = WPC EQU HBASE+1 ; W WRITE PRECOMPENSATION REGISTER
111 00C2 = SECNT EQU HBASE+2 ; R/W SECTOR COUNT REGISTER FOR FORMAT COMM.ONLY
112 00C3 = SECNO EQU HBASE+3 ; R/W SECTOR NUMBER REGISTER
113 00C4 = CYLLO EQU HBASE+4 ; R/W CYLINDER LOW REGISTER
114 00C5 = CYLHI EQU HBASE+5 ; R/W CYLINDER HIGH REGISTER
115 00C6 = SDH EQU HBASE+6 ; R/W BYTES PRE SECTOR / DRIVE ↕ / HEAD ↕ /
116 00C7 = STATUS EQU HBASE+7 ; R STATUS REGISTER
117 00C7 = COMMD EQU HBASE+7 ; W COMMAND REGISTER
118 ;
119 0000 = STRATE EQU 0 ;STEPIING RATE TRACK TO TRACK = BUFFERED STEP
120 ;
121 ;
122 0010 = REST EQU 10H OR STRATE ;RESTORE COMMAND WITH STRATE
123 0070 = SEEK EQU 70H OR STRATE ;SEEK COMMAND WITH STRATE
124 0020 = READ EQU 20H ;READ COMMAND
125 0022 = READL EQU 22H ;READ LONG
126 0030 = WRITE EQU 30H ;WRITE COMMAND
127 0032 = WRITEL EQU 32H ;WRITE LONG
128 0050 = FORMAT EQU 50H ;FORMAT COMMAND
129 ;
130 ;
131 ; *****
132 ; *** ERROR REGISTER EQUATES ***
133 ; *****
134 ;
135 0001 = DAMNFD EQU 01H ; ADDR. MARK NOT FOUND
136 0002 = TRO EQU 02H ; TRACK 0 ERROR
137 0004 = ABC EQU 04H ; ABORTED COMMAND
138 0010 = IDNFD EQU 10H
139 0020 = CRCID EQU 20H ; CRC-ERROR ID-FIELD
140 0040 = UMCOR EQU 40H ; UNCORRECTED DATA IN DATA FIELD
141 0080 = BBD EQU 80H ; BAD BLOCK DETECTED
142 ;
143 ; *****
144 ; *** STATUS REGISTER EQUATES ***
145 ; *****
146 ;
147 0001 = CERR EQU 01H ; CONTROLLER ERROR
148 0004 = CORR0 EQU 04H ;DATA CORRECTED IN DATA FIELD (ECC)
149 0008 = CDRQ EQU 08H ; CONTROLLER DATA REQUEST
150 0010 = DSEEC EQU 10H ; DRIVE SEEK COMPLETE
151 0020 = DWRFA EQU 20H ; DRIVE WRITE FAULT
152 0040 = DREADY EQU 40H ; DRIVE READY
153 0080 = CBUSY EQU 80H ; CONTROLLER BUSY
154 ;
155 ; ERROR CODES
156 ;

```

```

006 (RBIOS)

157 0068 = NTRDY EQU 00068H ; NOT READY
158 95C0 = FATER EQU 095C0H ; FATAL ERROR
159 20C0 = IOER EQU 020C0H ; I/O ERROR
160 ;
161 ; MISC EQU
162 ;
163 0002 = UNIT1 EQU 02H ; UNIT SELECT
164 0080 = ECC EQU 80H ; ECC INDICATOR
165 0020 = SECSIZ EQU 20H ; SECTOR SIZE = 512 BYTES/SECTOR
166 0001 = HEAD1 EQU 01H ; HEAD INDICATOR
167 0018 = DRIVE4 EQU 18H ; DRIVE 4 INDICATOR
168 PAGE
    
```

```

007 (RBIOS)

169
170
171 ;*****
172 ;*****
173 ;***** *****
174 ;***** HARD DISK DRIVER *****
175 ;***** *****
176 ;*****
177 ;*****
178 ;
179 ;
180 ;
181 ;*****
182 ;* *
183 ;* WINDRE RESTORE HARD DISK *
184 ;* *
185 ;*****
186
187 FSAF
188 WINDRE:
189 0000 3E10 MVI A,REST
190 0002 C5 PUSH B ; DUMMY PUSH
191 0003 C31700 JMP WINMAIN ; GO TO HARD DISK MAIN ROUTINE
192
193
194
195
196 ;*****
197 ;* *
198 ;* WINDFM FORMAT TRACK *
199 ;* *
200 ;*****
201
202 WINDFM:
203 0006 3E50 MVI A,FORMAT
204 0008 C5 PUSH B ; DUMMY PUSH
205 0009 C31700 JMP WINMAIN ; GOTO MAIN ROUTINE
206
207
208 PAGE
    
```


008 (RBIOS)

```
209
210
211 ;*****
212 ;#
213 ;* WINDRW READ OR WRITE ENTRY POINT
214 ;#
215 ;*****
216
217 WINDRW:
218 000C 3A0000 LDA DSKCMD
219 000F B7 ORA A
220 0010 3E30 MVI A,WRITE ;WRITE COMMAND
221 0012 CA1700 JZ WINMAIN ;JUMP IF A WRITE COMMAND SEQ.
222 0015 3E20 MVI A,READ ;READ COMMAND
223 WINMAIN:
224 0017 32E800 STA FUNCT ; SAVE FUNCTION
225 001A 219000 LXI H,0
226 001D 220000 SHLD ERRBUF ;RESET ERROR STATUS
227
228
229 PAGE
```

009 (RBIOS)

```
230
231 ;
232 ; ** CHECK IF DRIVE READY
233 ;
234 FIXD1:
235 0020 3EAA MVI A,0AAH
236 0022 D3C4 OUT CYLLO ;CHECK IF DRIVE IS UNDER POWER
237 0024 3E55 MVI A,055H
238 0026 D3C3 OUT SECNO
239 0028 D8C4 IN CYLLO
240 002A FEAA CPI 0AAH
241 002C C23600 JNZ FIXNDR ;JUMP IF NOT READY
242 002F D8C3 IN SECNO
243 0031 FE55 CPI 055H
244 0033 CA3C00 JZ FIXD2 ; JUMP IF READY
245 FIXNDR:
246 0036 216800 LXI H,NTROY ; HL - ERROR MESSAGE
247 0039 C3C000 JMP ER2
248 PAGE
```

010 (RBI05)

```

249
250          FIXD2:
251 003C 3A0000      LDA  HSTSEC      ; GET SECTOR NUMBER
252 003F 3D          DCR  A              ; START SECTOR = 0
253 0040 D3C3        OUT  SECHD        ; SET SECTOR NUMBER
254 0042 3A0000      LDA  HSTDSK
255 0045 210000      LXI  H,NBRFLEX
256 0048 96          SUB  M
257 0049 47          MOV  B,A          ; SAVE UNIT NUMBER
258 004A 07          RLC
259 004B 07          RLC
260 004C E618        ANI  18H
261 004E 4F          MOV  C,A          ; SAVE DRIVE NUMBER
262 004F 78          MOV  A,B          ; GET UNIT NUMBER
263 0050 1F          RAR
264 0051 3E00        MVI  A,0
265 0053 D25800      JNC  FIXD3      ; JUMP IF EVEN DRIVE NUMBER REQUESTED
266 0056 3E02        MVI  A,UNIT1    ; DRIVE 1 REQ. HEAD 2 + 3
267          FIXD3:
268 0058 B1          ORA  C              ; COMBINE WITH DRIVE NUMBER
269 0059 F6A0        ORI  ECC OR SECSIZ ; SET ECC AND SECTOR SIZE
270 005B 47          MOV  B,A
271 005C 2A0000      LHL  HSTTRK      ; GET TRACK
272 005F 7D          MOV  A,L
273 0060 1F          RAR
274 0061 3E00        MVI  A,0
275 0063 D26900      JNC  FIXD4      ; JUMP IF TRACK NUMBER EVEN (HEAD = 0)
276 0066 28          DCX  H              ; CORRECT TRACK NUMBER FOR LATER
277 0067 3E01        MVI  A,HEAD1
278          FIXD4:
279 0069 80          ORA  B              ; COMBINE SDH
280 006A D3C6        OUT  SDH          ; SET HEAD DRIVE SEC. SIZE AND ECC
281 006C D8C7        IN   STATUS
282 006E E650        ANI  DREADY OR DSEEC
283 0070 FE50        CPI  DREADY OR DSEEC
284 0072 C23600      JNZ  FIXNDR      ; JUMP IF DRIVE NOT READY
285 0075 7C          MOV  A,H
286 0076 1F          RAR
287 0077 D3C5        OUT  CYLHI      ; SET CYLINDER HIGH
288 0079 7D          MOV  A,L
289 007A 1F          RAR
290 007B D3C4        OUT  CYLLO      ; SET CYLINDER LOW
291 007D 3AE800      LDA  FUNCT        ; GET FUNCTION
292 0080 D3C7        OUT  COMND      ; SET FUNCTION
293 0082 E6F0        ANI  OFOH
294 0084 FE20        CPI  READ          ; READ
295 0086 CA9600      JZ   RD
296 0089 FE30        CPI  WRITE        ; WRITE
297 008B CACF00      JZ   WR
298 008E FE50        CPI  FORMAT      ; FORMAT
299 0090 CACB00      JZ   WR0
300 0093 C30A00      JMP  WR2          ; SEEK OR RESTORE
301          PAGE

```

011 (RB10S)

```
302
303 ;*****
304 ;READ ROUTINE
305 ;*****
306 RD:
307 0096 0E00 MVI C,DATA ; PORT ADDRESS
308 0098 0600 MVI B,00H ; TRANSFER LENGTH =256
309 009A 2A0000 LHLD PMAADR ; GET BUFFER ADDRESS
310 009D CDA000 CALL WAIT ; WAIT UNTIL CONTROLLER FINISHED
311 RD2:
312 INIR
313 00A0+E0B2 DB 0EDH,0B2H
314 + ENDM
315 INIR
316 00A2+E0B2 DB 0EDH,0B2H
317 + ENDM
318 00A4 CDE300 CALL DESLCT ; DESELECT REQ. DRIVE
319 00A7 C1 POP B ; RESTORE RETRY COUNTER
320 00A8 AF XRA A ; RETURN GOOD STATUS
321 00A9 C9 RET
322 PAGE
```

012 (RB10S)

```
323
324 ;*****
325 ;WAIT ROUTINE
326 ;*****
327 00AA DBC7 WAIT: IN STATUS
328 00AC A7 ANA A
329 00AD FAAA00 JM WAIT ;JUMP IF CONTROLLER BUSY
330 00B0 1F RAR
331 00B1 D0 RNC ;RETURN IF NO ERROR
332 ;
333 ;
334 00B2 DBC1 ER1: IN ERROR
335 00B4 E1 POP M ;RESTORE RETURN ADDRESS
336 00B5 21C020 LXI H,I0ER
337 00B8 E660 ANI CRCID OR UMCOR
338 00BA C2C000 JNZ ER2 ;JUMP IF I/O ERROR
339 00BD 21C095 LXI H,FATER ;FATAL ERROR
340 ER2:
341 00C0 220000 SHLD ERRBUF ;SET ERROR FOR ERROR-HANDLER
342 00C3 CDE300 CALL DESLCT ;DESELECT REQ. DRIVE
343 00C6 C1 POP B ;RESTORE RETRY COUNTER
344 00C7 3EFF MVI A,OFFH
345 00C9 B7 ORA A ;SET ERROR FLAG
346 00CA C9 RET
347 PAGE
```

013 (REIOS)

```

348
349
350 ;*****
351 ;WRITE ROUTINE
352 ;*****
352 00C8 3E11 WRO: MVI A,17
353 00CD D3C2 OUT SECNT ; SET SECTOR COUNT FOR FORMAT
354 WR:
355 00CF 0E00 MVI C,DATA ; PORT ADDRESS
356 0001 0600 MVI B,00H ; TRANSFER LENGTH = 256
357 0003 2A0000 LHL PMAADR ; BUFFER ADDR.
358 WR1:
359 OTIR
360 00D6+ED83 DB 0EDH,0E3H
361 + ENDM
362 OTIR
363 00D8+ED83 DB 0EDH,0E3H
364 + ENDM
365 00DA CDA00 WR2: CALL WAIT ; WAIT UNTIL CONTROLLER READY
366 00DD CDE300 CALL DESLCT ; DESELECT REQ. DRIVE
367 00E0 C1 POP B ; RESTORE RETRY COUNTERS
368 00E1 AF XRA A ; SET GOOD STATUS
369 00E2 C9 RET
370
371 PAGE

```

014 (REIOS)

```

372
373 ;*****
374 ;*
375 ;* DESLCT DESELECT REQ. DRIVE AND SELECT DRIVE 4 *
376 ;*
377 ;*****
378
379
380 DESLCT:
381 00E3 3EB8 MVI A,ECC OR SECSIZ OR DRIVE4
382 00E5 D3C6 OUT SDH ; SELECT DRIVE 4
383 00E7 C9 RET
384
385
386
387
388
389
390
391 00E8 00 FUNCT: DB 0 ;FUNCTION BYTE
392 00E9 END

```

015 (RBIOS)

ABC	0004	137																			
BBD	0020	141																			
BC	0000																				
CBUSY	0020	153																			
CDRQ	0008	149																			
CERR	0001	147																			
COMND	00C7	117	292																		
CORRD	0004	148																			
CRCID	0020	139	337																		
CYLHI	00C5	114	287																		
CYLLO	00C4	113	236	239	290																
DAMNFD	0001	135																			
DATA	00C0	108	307	355																	
DE	00C2																				
DESLCT	0CF3	96	318	342	366	380															
DREADY	0040	152	282	283																	
DRIVE4	0018	167	381																		
DSEEC	0010	150	282	283																	
DSKCMD	0000	80	218																		
DWRFA	0020	151																			
ECC	0080	164	269	381																	
ER1	00B2	334																			
ER2	00C0	247	338	340																	
ERRBUF	0000	81	226	341																	
ERROR	00C1	109	334																		
FATER	95C0	158	339																		
FIXD1	0020	234																			
FIXD2	003C	244	250																		
FIXD3	0058	265	267																		
FIXD4	0069	275	278																		
FIXNDR	0036	241	245	284																	
FORMAT	0050	122	203	298																	
FUNCT	00E8	224	291	391																	
HBASE	00C0	107	108	109	110	111	112	113	114	115	116										
		117																			
HEAD1	0001	166	277																		
HL	0004																				
HSTOSK	0000	79	254																		
HSTSEC	0000	79	251																		
HSTRK	0000	79	271																		
IDMFD	0010	138																			
IOER	20C0	159	336																		
IX	0004																				
IY	0004																				
MULT	0000	80																			
MBRFLEX	0000	80	255																		
NTRDY	0068	157	246																		
PMAADR	0000	80	309	357																	
RD	0096	295	306																		
RD2	00A0	311																			
READ	0020	124	222	294																	
READL	0022	125																			
REST	0010	122	189																		
SDH	00C6	115	280	382																	
SECNO	00C3	112	238	242	253																
SECNT	00C2	111	353																		

		016	(RB IOS)			
SECSIZ	0020	165 +	269	381		
SEEK	0070	123 +				
STATUS	00C7	116 +	281	327		
STRATE	0000	119 +	122	123		
TRO	0002	136 +				
UNCOR	0040	140 +	337			
UNIT1	0002	163 +	266			
WAIT	00AA	310	327 +	329	365	
WINDFM	0906	96	202 +			
WINDRE	0000	96	188 +			
WINDRW	000C	96	217 +			
WINMAIN	0017	191	205	221	223 +	
WPC	00C1	110 +				
WR	00CF	297	354 +			
WRO	00CB	299	352 +			
WR1	0006	358 +				
WR2	00DA	300	365 +			
WRITE	0030	126 +	220	296		
WRITEL	0032	127 +				
!						

CP/M-80 BIOS LINKLIST

BTRECS	EAFD	DSKSLC	E589	LISTST	EC7D	SEKDSK	EAD8
CHECKO	E8E9	DSKTRK	E605	M1RS23	E010	SECSET	E73C
CINIT	EBF5	DSKTYP	E631	M2RS23	E011	SEKHST	EB09
CLOSE	E4FB	DTRKSE	E741	MODCRT	ECC7	SEKSEC	EADB
CLRLI1	F1F6	ERFLAG	EB0A	MOVE	DFE4	SEKTRK	EAD9
CLRLIN	DE3C	ERRBUF	EAED	MOVEL	DFEA	SIOINI	F53B
CMPRE	E65D	FATERR	E288	MULT	DFC2	SRLCOV	F5BB
COMBLO	EAE4	FIXINI	E562	NBRFLE	E00A	SRLIN	F5B2
COMPDE	DFE4	FLUSH	E4F0	NOTRDY	E236	SRLINS	F56C
CONFIG	E00F	FNERRM	E29F	NRDYRX	E247	SRLOUT	F550
CONIN	EC2F	FNTBLN	E115	OFST	E64C	SRLSTA	F55E
CONOUT	EC3D	FORMAT	E6B3	PIOINI	F5C5	STBIOS	DE00
CONST	EC21	FREAD	E6AB	PMAADR	EB02	SUBHLD	DFE0
CREAD	E69C	FUNNCH	F443	POPREG	ECC3	TMPCOL	F33D
CRTTBL	E116	FUNTBL	E015	PRTOUT	F5D0	TRANS	F34B
CURCOL	F33B	FWRITE	E6A4	PRTSTA	F5DA	TRKSET	E735
CWRITE	E695	GETLPO	F423	PUNCH	EC4B	TRNCHR	EEA0
DESLCT	EBEF	HSTACT	EB07	PVRS23	E013	TRNSPC	EEA7
DHOME	E83E	HSTDSK	EAE0	RDTRAK	E6EC	UNACNT	EAFF
DISKIO	E671	HSTSEC	EAE3	READER	EC5D	UNADSK	EADC
DISPER	DE39	HSTTRK	EAE1	READID	E87E	UNASEC	EADF
DMAADR	EB00	HSTWRT	EB08	READOP	E805	UNATRK	EADD
DMASET	E717	IOBSET	E00B	READSE	E897	WBOTE	DE03
DRQ	F341	IOERRM	E273	RESET	E9F0	WINDFM	EB12
DRQADR	F343	KBDINI	F34C	RETRYC	E00C	WINDRE	EB0C
DSKCMP	EAFE	KEYST	F39A	ROUTE	EC8D	WINDRW	EB18
DSKERR	EA01	KEYTBL	E1EE	RSFLAG	EB04	WINIT	EC0B
DSKGET	E552	KEYIN	F3A3	RSTC	E00D	WRCUPO	F2E2
DSKMSC	E60D	LANGUA	F34A	SACTIV	F5EE	WRPRCT	E25A
DSKSET	E71C	LIST	EC6D	SAVESE	EB08	WRTYPE	EB06
DSKSID	E5B5						

In the special version of the operating system for Arabic, all BIOS sections except MBIOS are located 4KB lower in memory than indicated by the BIOS linklist. The same is true of the version of the operating system supporting the Dead-Key facility, with the difference that these BIOS sections are located 2KB lower than indicated by the linklist.

