AA–P309A–TV

Rainbow™ 100

CP/M–86/80 Software Design and Maintenance Manual

digital equipment corporation

First Printing, April 1983

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

CP/M®, CP/M®–80 and CP/M®–86 are registered trademarks of Digital Research Inc.

Z80® is a registered trademark of Zilog, Inc.

8088® is a registered trademark of Intel Corporation.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC	MASSBUS	UNIBUS
DECmate	PDP	VAX
DECsystem–10	P/OS	VMS
DECSYSTEM–20	Professional	VT
DECUS	Rainbow	Work Processor
DECwriter	RSTS	
DIBOL	RSX	

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Printed in U.S.A.

CONTENTS

CONTENTS (Cont.)

FIGURES

1.0 SYSTEM ORGANIZATION

The CP/M-86/80 operating system (Version 1.0) is an enhanced version of CP/M-86 V1.1 The enhancements allow the operating system to execute both Z80 processor and 8088 processor transients.

The system is initially configured to provide the largest available transient program area (TPA) to 8088 transients (03D00H to top of RAM). The CP/M-86/80 operating system is therefore placed at 400H, above the 8088 interrupt vectors. A data block needed for Z80 interfacing is located just above the CP/M-86/80 operating system. See Figure 1-1.

When a Z80 transient is requested, the system must be reconfigured to allow the transient to start executing at 100H. Therefore, all needed modules must be placed as high in memory as possible to make the largest TPA available to Z80 transients. See Figures 1-2 and 1-3.

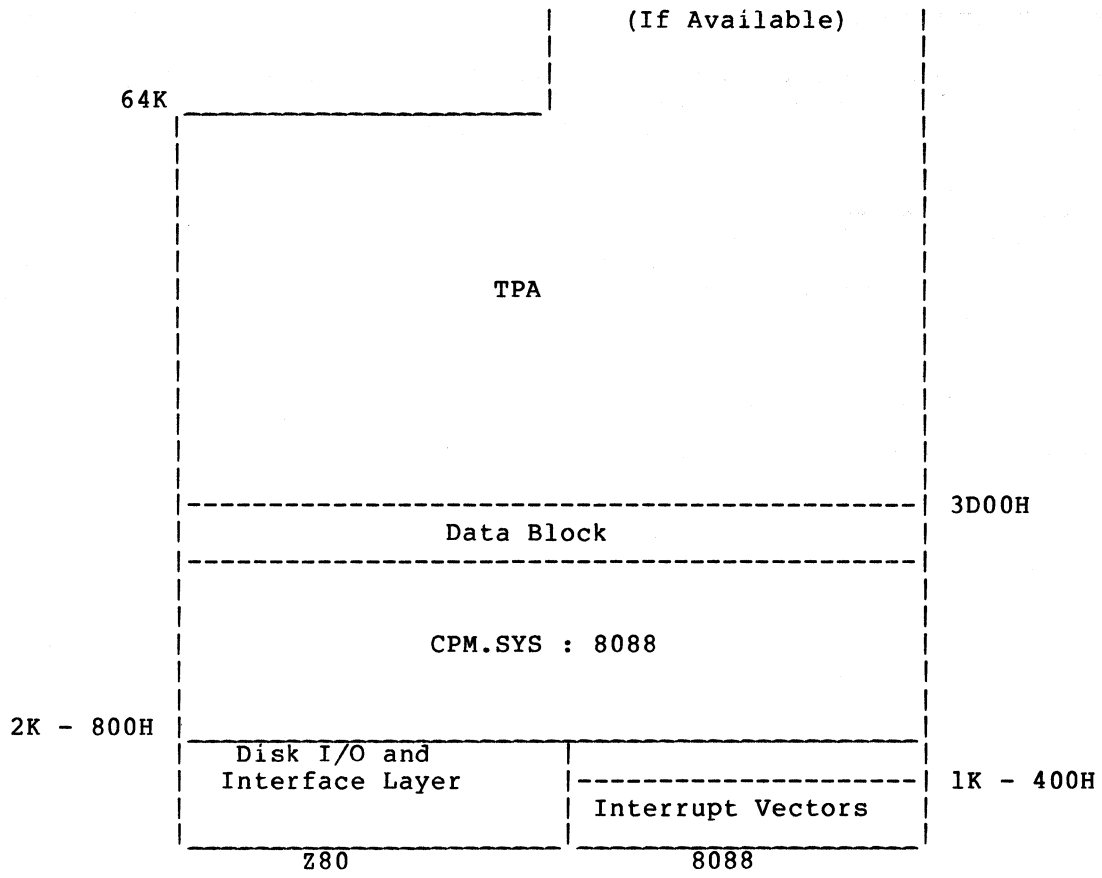

Figure 1-1 CP/M-86/80 Initial Configuration

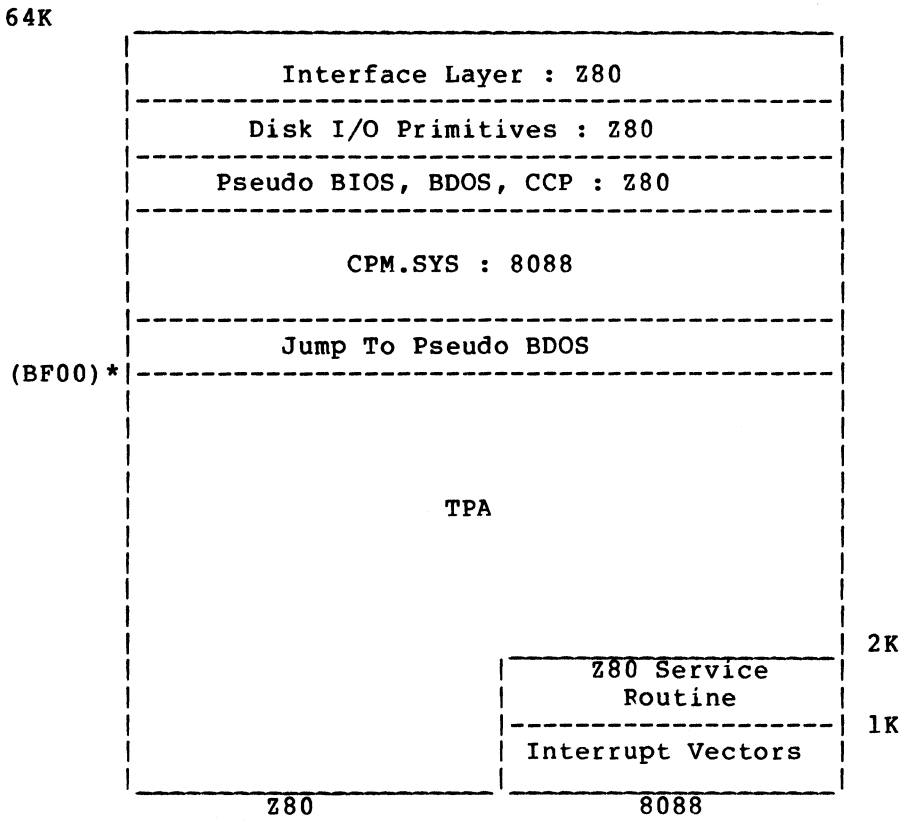

Figure 1-2 Z80 Transient Execution (64K System)

* Location 5 = JMP BF06. Therefore, the maximum TPA is BF40(Hex)-100(Hex) = BE00 which is approximately equal to 47.5K bytes.

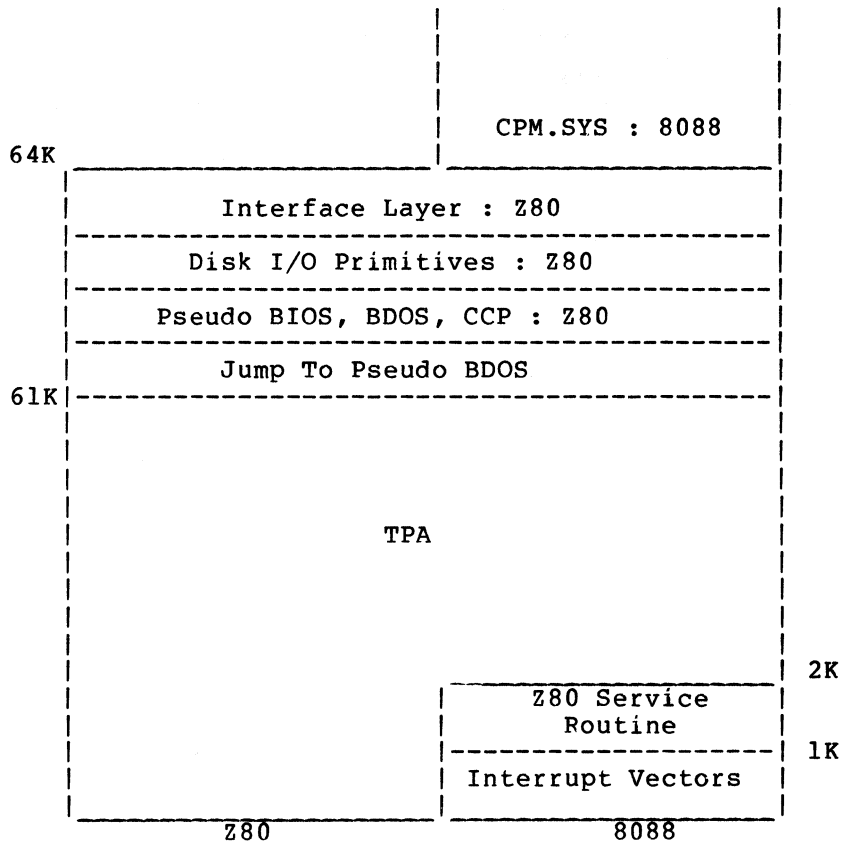

Figure 1-3 Z80 Transient Execution (128K or greater System)

2.0 SYSTEM OVERVIEW

Patches made to the CCP and BDOS within the CPM.SYS file allow the CP/M-86/80 operating system to execute both Z80 and 8088 transients. Patches released by Digital Research Inc. have also been applied (see Sections 4.1, 4.2 and 4.3. The BIOS portion of CPM.SYS is written to support the operating system facilities of CP/M on the Rainbow 100's dual CPU.

A Z80 executable Interface Layer module handles the cross-CPU communication for the Z80. This module is physically combined with the Primitive Routines which perform functions the 8088 cannot, as follows:

1. Disk I/O
2. Moving to/from private Z80 memory from/to shared memory
3. Executing Z80 code at a specific location

A pseudo BDOS and pseudo BIOS (Z80 executable) module stores a Z80 transient's BDOS/BIOS parameters in shared RAM and via the Interface Layer, transfers control to the waiting Z80 Service Routine in private 8088 memory.

The system must be reconfigured prior to the execution of a Z80 transient. To accomplish this, the system loads Z80CCP.SYS (8088 executable). This module is also used to reconfigure the system before resuming the execution of 8088 transients.

While the Z80 transient is executing, the 8088 waits in its Z80 Service Routine (in private 8088 memory) for BDOS and BIOS requests from the Z80. When the Z80 Service Routine gets control, it analyzes the parameters and then performs the actual BDOS call.

Buffers and data areas necessary for cross-CPU communication and system reconfiguration are contained in the Pointers/Buffers Data Block. This data block must reside in shared memory.

2.1 Utilities

2.1.1 **LDCOPY (8088 executable)** - This program creates the system tracks on a Rainbow 100 formatted disk.

2.1.2 **SAVE (8088 executable)** - This program saves a specified number of pages of Z80 memory into a specified file.

2.2 Bootstrap Programs (Z80 executable)

2.2.1 **BOOT100** - This bootstrap program resides on the first sector of the first track and is loaded by the Read Only Memory (ROM). When BOOT100 receives control, it reads the remainder of the first track and the entire second track by doing direct disk I/O. These tracks contain the loader program (8088 executable) and the interface and primitive routines (Z80 executable) needed for loading CPM.SYS and for executing the CP/M-86/80 operating system.

2.2.2 **BOOT101** - The BOOT101 program resides on the first sector of the second track. It is essentially the same as BOOT100, but includes preface information required by the RX50 bootstrap specification. (This file is not included in the CP/M-86/80 operating system BIOS listings. The code is not executed.)

3.0 CROSS-CPU COMMUNICATION

The 8088 and the Z80 communicate via message packets. When either CPU requests service from the other, the address of the message packet (packet pointer) is stored into an agreed upon location and the "master" CPU interrupts the "slave" CPU. The "master" waits while the "slave" CPU retrieves the packet pointer, performs the requested function, and indicates completion by interrupting the "master." See Figures 3-1 and 3-2.

The 8088 and the Z80 reverse master and slave roles when the 8088 requests the Z80 to start executing at a specific address -- the Z80 is now the "master" and the 8088 is the "slave." When the Z80 is finished, it does a BDOS call 0 to make the 8088 the "master" CPU again and the Z80 the "slave".

All message packets and the packet pointers must be in shared Random Access Memory (RAM). Message packets are constructed for the following functions:

FUNCTION CODE (HEX)	FUNCTION	MASTER CPU
13	Reads one sector on a specified drive	8088
14	Writes one sector on a specified drive	8088
15	Check media on a specified drive	8088
21	Start a Z80 program at a specified address	8088 to Z80
22	Move blocks of data in Z80 RAM space	8088
>40	BDOS/BIOS (for BDOS call 0) ----->	Z80 Z80 to 8088

See Sections 7.1 through 7.3 for message packet formats.

```
8088 REQUESTING Z80 SERVICE (disk/move)
=======================================
(Code exists in the CP/M-86/80 operating system BIOS)
     clear done flag
     store packet address (in data block)
     interrupt Z80
     wait for Z80 to clear interrupt
     wait for done flag to be set <---(interrupt 39)
     get packet address (in BIOS)


8088 REQUESTING Z80 SERVICE (start)
===================================
(Code exists in Z80CCP.SYS in private 8088 memory)
     clear done flag
     store packet address (in data block)
     interrupt Z80
     wait for Z80 to clear interrupt
     go to 8088 Service Loop (see below)


8088 SERVICE LOOP FOR Z80 REQUESTS (BDOS/BIOS)
==============================================
(Code exists in Z80CCP.SYS in private 8088 memory)
     wait for done flag to be set  <---(interrupt 39)
     get packet address (in BIOS)
     perform BDOS/BIOS function
          (if function is BDOS call 0, control is not returned
          to this loop but to the ccp)
     set status in packet
     clear done flag
     store packet address (in data block)
     interrupt Z80
     wait for Z80 to clear interrupt


8088 INTERRUPT 39 SERVICE ROUTINE
=================================
(Code exists in the CP/M-86/80 operating system BIOS)
     clear interrupt
     retrieve packet address (in data block)
     save packet address (in BIOS)
     set done flag
     interrupt return
```

Figure 3-1 8088 Code Guide for Cross-CPU Communication

```
Z80 REQUESTING 8088 SERVICE (BDOS/BIOS)
=======================================
     clear done flag
     store packet address (in data block)
     interrupt 8088
     wait for 8088 to clear interrupt
     wait for done flag to be set  <---(RST 6)
         (If the routine called returns data)


Z80 SERVICE LOOP FOR 8088 REQUESTS (disk/move/start)
====================================================
     halt (wait for interrupt from 8088)


Z80 RST 6 SERVICE ROUTINE
=========================
     clear interrupt
     retrieve packet address (in data block)
     get packet function code
     if   function is a Z80 function (disk/move/start)
          perform function
               (if function is start, control is not returned
               to this routine; a BDOS call 0 is eventually done
               instead)
          set status in packet
          store packet address (in data block)
          interrupt 8088
          wait for 8088 to clear interrupt
      else
          set done flag
      endif
      interrupt return
```

Figure 3-2 Z80 Code Guide for Cross-CPU Communication

4.0 CPM.SYS

4.1 CCP Patches

The following changes are applied as patches against the CP/M-86 operating system V1.1. They exist in the file CPLPATCH.A86 with the BDOS patches. The patch space is ORG'd after the BIOS.

4.1.1 **Saving the Boot Drive Number** - The system loads system files from the boot disk, so it must save the boot drive number to use in the File Control Blocks (FCB's). This boot drive number is passed from the boot routine to the loader to the CP/M-86/80 cold start routine (in BIOS) and then to the CCP where it is saved.

MODIFICATIONS TO CODE

New code is marked by an asterisk (*) in the left margin.

```
     CCPSTART:
                .
                .
                .
                MOV       CDISK,AL
*               MOV       BOOTDRV,AL          ;SAVE BOOT DRIVE#
                CALL      SELECT
                .
                .
                .
*    BOOTDRV    RB        1
                .
                .
                .
```

4.1.2 **Allowing Z80 Transients** - When a transient is requested, a test is made for the existence of a .CMD file. If it exists, and if it is not SAVE.CMD, and if the system is currently configured for 8088 transients, the .CMD file is loaded and executed as in CP/M-86. If the system must be reconfigured, Z80CCP.SYS is loaded from the boot disk and executed. Upon return, the .CMD file is loaded and executed.

If a .CMD file does not exist when a transient is requested, a test is made for the existence of a .COM file. If a .COM file exists, Z80CCP.SYS is loaded and executed. Z80CCP.SYS reconfigures the system if necessary, loads the .COM file, and turns control over to the Z80 for execution of the .COM file.

Note that if SAVE(.CMD) is requested, it runs only if the most recent transient was a .COM file. SAVE.CMD is created using GENCMD SAVE 8080 CODE[A40]. This insures that SAVE is loaded in private 8088 space so that the data in Z80 private space and shared memory can be saved undisturbed.

If the most recent transient was a .CMD file, the CP/M-86/80 operating system occupies the space where SAVE is to be loaded and CP/M returns with a 'MEMORY NOT AVAILABLE' message.

See Section 9.0 for Z80CCP parameters, return codes, and error conditions.

MODIFICATIONS TO CODE

Code is modified where the CCP tests for a .CMD transient:

New code is marked by an asterisk (*) in the left margin; deleted code is marked by a semicolon (;) in the left margin.

```
     USERF1:
                CALL      SETDSK
                .
                .
                .
                MOV       SI,OFFSET CMDTYPE
                .
                .
                .
                CALL      OPENC
;               JNZ       LOADUSER
*               (new code to load Z80CCP.SYS if necessary)
                CALL      RESETDISK
                .
                .
                .
```

MODIFICATIONS TO CODE

A new label is added after the user FCB information has been determined before the information is moved to the user's page 0. This label is needed to execute the .CMD file after the system is reconfigured.

```
     GOUSER:
                .
                .
                .
                MOV       COMFCB+16,AL
                MOV       BYTE PTR COMREC,0H
*    GOUSER1:
                MOV       DI,USERFCB
                .
                .
                .
```

MODIFICATIONS TO CODE

Code is also modified just before the registers are reset prior to doing a CALLF to execute the .CMD transient:

```
     BMOVE3:
                .
                .
                .
                MOV       BPTR,0H
                MOV       COMLEN,00H
*               (new code to run Z80CCP.SYS if necessary)
                MOV       ES,PAG0
                MOV       DS,PAG0
                .
                .
                .
```

4.2 BDOS Patches

The following changes are applied as patches against the CP/M-86 operating system V1.1. They exist in the file CPLPATCH.A86 with the CCP patches. The patch space is ORG'd after the BIOS.

4.2.1 BDOS Function 10 -- delete-character modification - A

modification has been made to BDOS to make the delete character behave like a backspace character.

MODIFICATIONS TO CODE

New code is marked by an asterisk (*) in the left margin; deleted code is marked by a semicolon (;) in the left margin.

```
     L19:
                .
                .
                .
                JNZ         NOTH
*    DOBACKSP:
                OR          CH,CH
                .
                .
                .
     NOTH:
                CMP         AL,RUBOUT
                JNZ         NOTRUB
*               JMPS        DOBACKSP
;               OR          CH,CH
;               JZ          READNX
;               MOV         AL,ES:[BX]
;               DEC         CH
;               DEC         BX
;               JMPS        RDECH1
```

4.2.2 MOVE Routine - call to new BIOS function - Whenever

data is moved to a user's data area, a test must be made to determine if the data to be moved is/will be in private Z80 space. This is accomplished by calling a new BIOS function which performs the test and data move.

MODIFICATIONS TO CODE

New code is marked by an asterisk (*) in the left margin; deleted code is marked by a semicolon (;) in the left margin.

```
*       ZMOVEF      EQU         2500+(3*21)
                    .
                    .
                    .
        MOVE:
                    .
                    .
                    .
                    MOV         SI,DX
                    MOV         DI,BX
;                   CLD
;           REP MOVS            AL,AL
*                   CALL        ZMOVEF
                    POP         CX
                    .
                    .
                    .
```

4.3 Digital Research Patches

Digital Research Inc. patches to CP/M-86 V1.1 CCP and BDOS have been applied to the CP/M-86/80 operating system. They are contained in the DRIPATCH.A86 file.

5.0 BIOS

The source code of the BIOS (in file CPLBIOS.A86) is divided into seven sections:

1. The jump table at the beginning of the BIOS provides entry points to the 21 functions specified in Digital Research Inc. documentation and to 2 functions which have been added for the CP/M-86/80 operating system.

2. The initialization routines (in INCLUDEd file CPLBIOS1.A86) are described in Section 5.1.

3. The character input/output routines (in INCLUDEd file CPLBIOS1.A86) handle single character I/O functions for the console, list, reader, and punch devices. These routines are described in Section 5.2.

4. The interrupt handlers (in INCLUDEd file CPLBIOS1.A86) process the serial port receive and Z80 interrupts. These handlers are described in Section 5.3.

5. The disk input/output routines (in INCLUDEd files CPLBLOK.LIB and CAT.LIB) perform all CP/M disk functions, including blocking and deblocking. They are described in Section 5.4.

6. The utility subroutines, described in Section 5.5, are used to display error messages.

7. The data area is in INCLUDEd file CPLBIOS2.A86.

5.1 Initialization

The cold boot routine, INIT, performs the following initialization functions:

- Sets the segment register values and the stack
- Sets the IOBYTE to its default setting
- Indicates that the Z80 is not running
- Initializes the interrupt vectors
- Displays the signon message
- Transmits the default disk drive identity to the CCP
- Initializes the serial I/O ports

INIT clears the screen and sets the cursor at line 3 before displaying the signon message.

The warm boot routine, WBOOT, performs the following initialization functions:

- Indicates that the Z80 is not running
- Initializes the interrupt vectors
- Initializes the serial I/O ports

5.2 BIOS Character I/O Routines

The character I/O routines handle all input and output for the CP/M console, list, reader, and punch devices. The routines operate at three levels: the logical level (which CP/M sees), the physical level (which CP/M uses for I/O redirection), and the hardware level. These are described below.

5.2.1 **Hardware Device Drivers** - The hardware device drivers are at the lowest level of the character I/O hierarchy. All console input/output is handled by the ROM via software interrupt 40. The serial port drivers for the communications port and the printer are described below.

There is also a set of drivers for null devices. The null input driver, PNULIN, returns end-of-file (control-Z) at each call. The null output driver, PNULOUT, returns immediately. The input and output null status test drivers, PNULSTI and PNULSTO, always return a "ready" status.

5.2.1.1 **Console I/O** - All console input and output is handled by the ROM. The parameters are passed in the registers as follows:

```
CONSOLE OUT (CRTOUT)
     DI = function code = 0
     AL = ASCII character code

CONSOLE IN (CRTIN)
     DI = function code = 2
     AL = returned character
     CL = returned status
          0 = no character available
         FF = character available in AL

CONSOLE IN STATUS (CRTSTI)
     DI = function code = 4
     CL = returned status
          0 = no character available
         FF = character available
```

The BIOS uses these ROM routines for Console In, Console Out, and Console Status. Console Out simply invokes ROM function 0 and Console Status invokes ROM function 2. It repeats this until a character becomes available. Therefore, it always returns with a character. The BIOS also contains a CRTSTO routine which returns a "ready" status.

5.2.1.2 **Control Blocks** - Each serial port (Communications and Printer) is associated with a control block which contains the port address, input processing information, and an input buffer for the port. This control block is used by all communication routines that access the physical device.

The format of the control block is:

OFFSET	NAME	CONTENTS
0	QTPORT	Control port address
1	QTFLAGS	Processing flags Bit 0 set to suspend output Bit 1 set for XON/XOFF Bit 2 set if initialization required
2	QTNRCHR	Number of characters currently in buffer
3	QTCAP	Buffer capacity in bytes
4	QTINPTR	Buffer input pointer (offset of next character to be stored in buffer)
5	QTOTPTR	Buffer output pointer (offset of next character to be removed from buffer)
6	QTDEND	Offset of last data byte position in buffer
7	QTDEVID	Physical device identification for timeout error message
10	QTDATA	Buffer data area

All offset data contains offsets relative to the beginning of the control block.

5.2.1.3 **Serial Port Initialization** - The subroutine P232INIT performs initialization of all the serial ports. Each of the serial ports is set to the proper configuration by loading the BX register with the port control block offset, loading the SI register with the offset of the initialization sequence, and calling P232IPR which initializes one port. The printer and standard communications ports are initialized by the firmware, thus the control blocks are set not to initialize them (interrupts are initialized). The optional communication port control block is set to not perform the initialization.

5.2.1.4 **Input and Output** - The hardware input/output drivers are called by the physical I/O routines to directly access the serial port USARTs. Four subroutines perform the hardware I/O tasks.

P232IN retrieves an input character from the circular buffer in the port control block (whose offset is in the BX register). If the buffer is empty, the routine waits until a character is available from the buffer. The characters are placed into the buffer by the serial port interrupt handler I232RX.

P232OUT writes a character to a serial port. Before writing the character, it makes sure that the USART transmitter is ready and that output is not suspended. If the port is a printer port, XOFF characters are not transmitted.

P232STI determines whether an input character is ready by checking the number of characters in the circular buffer. If the character count is zero, then "not ready" status is returned.

P232STO determines whether an output port is ready to receive a character. If the USART transmitter is ready and the output is not inhibited, "ready" status is returned. Otherwise, "not ready" status is returned.

5.2.2 **Physical Device Drivers** - There are eleven physical devices defined by CP/M. Those not associated with actual hardware devices are normally regarded as null devices.

5.2.2.1 **Mapping of Device Names to Ports** - Each of the physical device names is associated with a hardware input/output device or is regarded as a null device. A null device returns an end-of-file indication (control Z) on each input call, returns immediately on each output call, and returns a "ready" status at all times. The correspondence of I/O devices to CP/M physical devices in this BIOS is as follows:

NAME	DEVICE
TTY:	Printer
CRT:	Console
UC1:	Optional communications port
PTR:	Communications port
UR1:	Null device
UR2:	Null device
PTP:	Communications port
UP1:	Null device
UP2:	Null device
LPT:	(Same as TTY:)
UL1:	Null device

5.2.2.2 **Relation to Hardware Device Drivers** - Except for the CRT, which has no control block, the physical device drivers for serial I/O are implemented in their simplest form by code which loads the BX register with the address of the appropriate hardware device control block and branches to the applicable hardware driver. In the case of null devices, the physical device drivers are equated directly to the null device drivers.

5.2.3 **Logical Device Drivers** - In CP/M there are four logical devices used for character input/output:

CON: The console device

AXI: The auxillary input device

AXO: The auxillary output device

LST: The list device

These devices are referred to implicitly by BIOS calls. They, in turn, call physical device drivers to perform the input, output, or status testing function.

5.2.3.1 **I/O Redirection (IOBYTE)** - The CP/M-86/80 operating system supports redirection of character input/output from logical device drivers to various physical devices. The redirection is specified by setting bits in a location called the IOBYTE. This BIOS implements the full redirection of all four logical devices.

This redirection is accomplished by code in the logical drivers that fetches the IOBYTE, isolates the 2-bit field of interest, and branches to a physical driver according to the value in that field.

5.2.3.2 **Default I/O Assignments** - The IOBYTE is set during the cold boot process to associate the four logical device drivers with default physical drivers. These default settings are:

Logical	to	Physical
CON:		CRT:
AXI:		PTR:
AXO:		PTP:
LST:		LPT:

5.3 BIOS Interrupt Handlers

The 8088 generates five interrupts of concern to the BIOS. BDOS calls are type-224. A "receive ready" condition at either of the two standard serial ports causes a type-36 interrupt. Interruption by the Z80 causes a type-39 interrupt. A "receive ready" condition at the optional communications port causes a type-37 interrupt. Once every line cycle, a type-44 interrupt is generated by the hardware. This interrupt is intercepted by BIOS, which in turn generates a type-100 interrupt, which can be intercepted by a user program. The interrupt vectors are initialized during cold and warm boot procedures.

5.3.1 **RS-232 Receive Ready** - The serial port interrupt handler, I232RX, processes interrupts from any or both of the standard serial port receive functions. The I232RPT subroutine performs interrupt processing for a given port.

I232RX first saves all the registers. The address of each port control block is placed in the BX register and I232RPT is called. After processing each port, the registers are restored and an interrupt return operation is performed. The optional communication port interrupt handler, I232RX2, operates similarly for the optional communication port.

The subroutine I232RPT tests to see if the port (whose control block address is in the BX register) has a receiver ready condition; if not, it tests for a break condition. If neither condition exists, an immediate exit is made. If a break condition exists on the line, the line is reset and the data port is cleared. If a character is ready, it is read and processed. If an overrun error occurs, it is assumed that an XON character may have been lost, and the output is unconditionally enabled.

If the port is the communications port, the character is stored in the control block buffer. If the port is the printer port, special processing is necessary for XON and XOFF control characters, which are used to enable and disable output. This feature is controlled by a flag in QTFLAGS. The following table shows the processing logic.

OUTPUT IS	CHARACTER	ACTION
Enabled	XOFF	Suspend output; discard character
	XON	Place into buffer
	any other	Place into buffer
Suspended	XOFF	Discard character
	XON	Enable output; discard character
	any other	Place into buffer

The output is suspended or resumed by changing the state of a flag in QTFLAGS.

If the printer's buffer becomes half full, an XOFF character is transmitted. Similarly, should the printer's buffer become entirely full an XOFF and a BEL character are transmitted. When the buffer returns to being less than half full, an XON character is transmitted to the printer port.

If any overrun error condition exists at the port and the received character is an XON, it is assumed that the lost character was an XOFF. The character is discarded and output is enabled.

The code at label I232RP20 stores a character in the circular buffer. If a parity error has occurred, an ASCII "SUB" character is substituted for the received character. The code at label I232RP30 sends a BEL character to the device when the circular buffer is full. Label I232RP40 contains code to process a break condition.

5.3.2 **Z80 Interrupts** - The Z80 interrupt handler, TYPE_39_SERV, processes hardware interrupts from the Z80. The handler saves the packet pointer (Z80PKT) and indicates that a Z80 interrupt was received by setting a done flag (ZOT).

Note that Z80FLAG and ZOT must be positioned immediately before TYPE_39_SERV so that Z80CCP.SYS and SAVE.CMD can access these flags.

When the BIOS wishes to communicate with the Z80, it calls the PACKER routine which takes items off the stack and places them into a message packet buffer. The packet pointer is stored and the Z80 is interrupted (SENDPKT routine). PACKER then waits for the Z80 to finish (WAITZ80) by testing the done flag (ZOT).

5.4 BIOS Disk I/O Routines

The discussion of disk input/output routines is divided into four components: logical I/O, blocking/deblocking, error message display, and physical I/O. Because CP/M operates with reference to 128-byte logical disk sectors and the Rainbow 100 physical disk contains 512-byte sectors, the logical and physical I/O levels are separate, and special routines are used to block and deblock logical records to and from physical records. Error message display provides diagnostic information for the user.

The standard CP/M disk parameter tables are used to provide information on disk characteristics. The disk parameter headers (at DPE0 through DPE3) are generated using the DISKS macro from DISKDEF.LIB. The disk parameter blocks are generated using the GENDEF utility with the following parameters:

First physical sector	0
Last physical sector	39
Skew factor	1
Data allocation block size	2048
Disk size (in blocks)	195
Directory entries	128
Checked directory entries	128
System tracks	2

The disk sector translation table generated at XLT0 is overlaid with a special table providing a blocking factor of four and a physical skew factor of two.

A dummy disk parameter header is generated for VT180 media reference:

First physical sector	0
Last physical sector	35
Skew factor	1
Data allocation block size	1024
Disk size (in blocks)	171
Directory entries	64
Checked directory entries	64
System tracks	2

5.4.1 **Logical I/O Routines** - The nine logical disk I/O routines perform the functions specified in the Digital Research Inc.'s CP/M-86 System Guide. The implementation of each function is briefly described below.

HOME sets the desired track to zero. The physical seek is deferred until a read or write operation is performed.

SELDSK stores the desired drive number and returns the address of the disk parameter header for the desired drive. The actual physical select of the drive is performed only when a read or write operation is performed. If the drive selected is above the allowable range ("A" throught "D"), the routine returns with an error condition. For example, if the BDOS calls and an error condition is returned, the message "BDOS ERR ON x: SELECT: is displayed. If an unallocated disk is selected, SELDSK performs a media check to determine if it is VT180 media. If necessary, the pointer to the correct disk parameter block (that is, DPB0 or DPB4) is updated in the selected disk's parameter header. This routine also checks the BDOS log-in vector to determine if a disk reset has occurred.

SETTRK stores the desired track number of use when a read or write operation is performed.

SETSEC stores the desired sector for use when a read or write operation is performed.

SECTRAN performs a simple translation of the passed sector number (based at zero) in register CX using the translation table address passed in register DX.

SETDMA stores the desired memory offset for use when a read or write operation is performed.

SETDMAB stores the desired memory segment for use when a read or write operation is performed.

READ moves one logical record from the physical sector buffer to the memory (DMAB:DMA) address specified by the user.

WRITE moves one logical record from the DMA address to the physical sector buffer. The flag HSTWRT is set to indicate that the buffer has been changed. If an attempt is made to write to a ROBIN disk, the message "Cannot write on VT180 disk on drive x" is displayed.

5.4.2 **Blocking and Deblocking** - Because the logical disk I/O routines deal with 128-byte sectors and the physical disk format contains 512 bytes, auxiliary routines are necessary to map the logical sector numbers onto physical sector numbers, assist in extracting the desired data from the physical sector buffer, and manage the contents of the buffer.

READ and WRITE use RWOPER to assist in the blocking of logical records into 512-byte physical sectors. Using the logical sector number, RWOPER computes the physical sector number and the location within the buffer for the logical sector data. The identity of the buffer contents is maintained in the variables HSTDSK, HSTTRK, and HSTSEC, which contain the drive, track, and physical sector associated with the current buffer contents. If the desired drive, track, and (physical) sector are the same as those associated with the buffer contents, the appropriate data is moved to/from the buffer. If, however, the buffer has been changed (as indicated by the value of HSTWRT) it is written to disk. The desired sector is read from disk unless the BDOS has specified that this is a write to a previously unallocated block. Because BDOS signals a write to only the first sector of an unallocated block, RWOPER must check each non-directory write to see if the drive, track, and sector are still within the previously unallocated block. RWOPER calls READHST and WRITEHST to read and write physical sectors.

READHST sets up a message packet for the Z80 to read into the sector buffer. An error message is displayed if a hard error condition occurs.

WRITEHST sets up a message packet for the Z80 to write from the sector buffer. An error message is displayed if a hard error condition occurs.

5.4.3 **Error Message Display** - READHST and WRITEHST display messages on the logical console when disk error conditions occur.

In the message formats shown below, the "d" identifies the drive, "tt" the track (decimal), and "ss" the physical sector (decimal).

```
Drive not ready -- d:
Press CTRL-C to restart,
space bar to retry, or any other key to continue.

Drive write protected -- d:
Press CTRL-C to restart,
space bar to retry, or any other key to continue.

Seek error on drive d:, track tt
Press CTRL-C to restart,
space bar to retry, or any other key to continue.

Read error on drive d:, track tt, sector ss
Press return to ignore error, any other key to continue

Write error on drive d:, track tt, sector ss
Press return to ignore error, any other key to continue
```

5.4.4 **Physical I/O Routines** - Physical I/O is controlled by the Z80. See Section 7.1.

5.5 BIOS Utility Subroutines

Two utility subroutines are used for displaying messages directly from the BIOS routines. These messages consists of error or warning notifications requiring action by the console operator.

5.5.1 **Message Display** - The message display routine, PMSG, is called with register BX pointing to the message text. A zero byte terminates the message text.

5.5.2 **Decimal Number Display** - The decimal display routine, DECPRT, is used to display track and sector numbers in disk I/O error messages.

6.0 INTERFACE LAYER

The Interface Layer handles cross-CPU communication for the Z80. Two copies of these routines exist on the system disk: one is ORG'd at 100H and is used when the CP/M-86/80 operating system is executing .CMD transients; the other is ORG'd at the high end of shared memory and is used when the CP/M-86/80 operating system is executing .COM transients.

The Interface Layer code is in the Z80CODE.ASM file. PRIVATE.ASM and SHARED.ASM contain different assembly time conditional values to control the assembly of the appropriate addresses and routines. This code is physically combined with the Primitive Routines (see Section 7.0).

6.1 Interface Layer Initialization

The boot routine loads the Interface Layer (and its associated Primitive Routines) into Z80 private memory. When the boot routine has completed its functions, the Interface Layer's initialization routine initializes the RST6 vectors and appropriate data areas; enables interrupts; and then halts at RST4 until it is interrupted by the 8088.

When a new copy of the Interface Layer (and its associated Primitive Routines) is loaded, control is passed to it by using the "controlling" copy of the Interface Layer to execute a Start-Z80 function at the "new" interface's initialization routine. The "new" copy now has control and the RST6 vectors and data areas are reinitialized. A return from the start function never occurs; instead, the Z80 halts at RST4 waiting for an 8088 interrupt.

6.2 Z80 Interruption

When the Z80 is interrupted, control is given to the RST6 address (30H). A jump is taken to a routine (PKTPRO) which clears the interrupt, retrieves the packet pointer (I88PKT), determines the function number and jumps to the specific function handler routine.

Upon return from the disk and move functions, the packet pointer is stored (Z80PKT), the Z80 interrupts the 8088, and the interrupt return is taken.

A BDOS/BIOS call is initiated by the Z80 (in I88SVC, see 6.3 below). The RST6 routine is entered because the 8088 has indicated that the call was performed. The waiting Z80 routine (I88SVC) is therefore signalled that the 8088 is finished by setting a flag (DONEFL), placing the packet pointer in register HL, and performing an interrupt return.

No interrupt return is taken if the function is Start-Z80.

All registers are preserved upon entry. Interrupts are re-enabled after saving the registers and after a return from any disk I/O routine.

6.3 Z80 Request Interrupts

When the pseudo BDOS/BIOS routines wish to interrupt the 8088 for actual BDOS/BIOS service, it calls I88SVC, which expects a pointer to the message packet in register HL. This packet pointer is stored (Z80PKT) and a done flag is cleared (DONEFL). The 8088 is interrupted and interrupts are re-enabled.

While the 8088 is processing the Z80 request, the Z80 is halted. When execution resumes, an interrupt has already been received and processed. The done flag is tested to see if the Z80's service request has been completed. If not, the Z80 halts again. (The 8088 might request disk I/O or move service from the Z80 before it can complete the Z80's BDOS/BIOS service request.) If the requested service has been performed, register HL contains the packet pointer. Interrupts are re-enabled and control is returned to the pseudo BDOS/BIOS routines.

If service for BDOS call 0 was requested, the Z80 will remain halted (not RST4) until another start-Z80 function occurs or until the system is reconfigured. Control does not return to the pseudo BDOS/BIOS routines.

7.0 PRIMITIVE ROUTINES

Code for the Primitive Routines is contained in the Z80CODE.ASM file. PRIVATE.ASM and SHARED.ASM contain different assembly time conditional values to control the assembly of the appropriate addresses and routines. This code is physically combined with the Interface Layer (see Section 6.0).

The following routines receive control from the RST6 routine PKTPRO with the packet pointer in register IX.

7.1 Disk I/O

7.1.1 **Check Media** - Called by the BIOS in the 8088 whenever a drive is selected for the first time after a system reset, this function attempts to determine whether a Rainbow 100 or VT180 disk is mounted on the specified drive.

The check media function Message Packet uses the following format:

Byte 0 Function code = 15H

Byte 1 Returned status 0 = RAINBOW
2 = VT-180

Byte 2 Bits 0-4 not used
Bits 5-7 drive number (binary value)

A disk restore operation and then a read of sector 10 is first performed. If a record is not found, VT180 media is assumed. If the record is successfully read or if any other error occurs, Rainbow 100 media is assumed. Appropriate data areas are initialized to reflect media type.

7.1.2 **Read/Write Sector(s)** - This Primitive Routine uses the following Message Packet format:

Byte 0 function code = 13H for read, 14H for write

Byte 1 returned controller status

Byte 2 bits 0-4 sector number to begin reading/writing (binary)
bits 5-7 drive number (binary)

Byte 3 bits 0-6 track number (binary value)
bit 7 not used

Byte 4 LSB of data address in shared RAM

Byte 5 MSB of data address in shared RAM

Byte 6 Number of sectors to read/write

7.1.2.1 **Read/Write Sector(s) Processing** - The drive is selected before reading or writing to a sector. The first access of the drive causes it to be homed. If necessary, a seek of the appropriate track is performed. Appropriate error recovery operations are performed with the read or write.

7.1.2.2 **Drive Ready** - The drive is considered ready if the controller reports ready status and if the disk is determined to be right side up on the first access of the drive. A change in the index pulse status determines correct disk placement.

7.1.2.3 **Disk Home** - Two attempts are made to home a disk. If the first attempt fails, the controller performs five "step-in's" and then tries again.

7.1.2.4 **Disk Seek** - Two attempts are made to locate a track. If the first attempt fails, the drive is homed and the seek is attempted again.

For a VT180 disk, the desired track number is multiplied by two to get the seek track number. The seek is performed with no verification. The track number is then read and compared to the desired track number.

7.1.2.5 **Disk I/O Loop** - The disk I/O loop used is ORGd at Z80 location 40 (see Z80BASE.ASM). The appropriate I/O instruction is stored at location 46 before this loop is executed. The low-memory location prevents 8088-induced wait states from affecting disk I/O performance.

7.1.2.6 **Read/Write Error Recovery** - Five attempts are made to read or write a sector. If these five attempts fail, and the original attempt failed because of a CRC error or a Record Not Found error, an advanced error recovery procedure is performed. This procedure consists of:

- Homing the drive
- Retrying the seek

Then four more attempts are made to read or write the sector. If this still fails, a seek to track 79 is made, followed by a seek to the desired track and four more attempts. If this fails still, the original error is reported.

7.2 MOVE

This function is called to move data within Z80 memory. The following Message Packet format is used:

Byte 0 Function code = 22H

Byte 1 Not used

Byte 2 LSB of source address

Byte 3 MSB of source address

Byte 4 LSB of destination address

Byte 5 MSB of destination address

Byte 6 LSB of number of bytes to move

Byte 7 MSB of number of bytes to move

The source and destination addresses specify addresses in Z80 memory. They are moved to registers HL and DE respectively. The byte count is placed in register BC and a string move is performed. Control is returned to the RST6 routine.

7.3 Start Z80

This function is called to start a Z80 transient (and also to turn control over to a new copy of the Interface layer -- see Section 6.1). The following Message Packet format is used:

Byte 0 Function code = 21H

Byte 1 Not used

Byte 2 LSB of Z80 start address

Byte 3 MSB of Z80 start address

Interrupts are disabled and the stack pointer is reset to the base of the stack. A return address of '00' is placed on the stack (in case the Z80 transient does a 'RET' to the operating system). The start address is obtained, interrupts are re-enabled, and the Z80 starts executing the transient.

Control does not return to the RST6 routine. A BDOS call 0 is made via RST3 and control is returned to the CP/M-86/80 operating system when the Z80 transient terminates.

8.0 PSEUDO BDOS AND PSEUDO BIOS

The Pseudo BDOS/BIOS routines receive control when a Z80 transient requests BDOS or BIOS service. The code for this module is in the Z80CODE.ASM file. It is assembled when combined with the SHARED.ASM file (see Section 10.3.3).

These routines place the parameters (in the registers) into a buffer and set the appropriate function code. A pointer to the buffer is placed in register HL and the 8088 is interrupted by calling I88SVC. When control is returned, the parameters are replaced in the registers and the Z80 transient continues execution.

These routines use their own stack.

Z80/8088 register mapping for the Pseudo BDOS/BIOS routines is shown below.

Message Packet Byte	Z80	8088
0	Function code*	Register AH
1	Register A	Register AL
2	Register C	Register CL
3	Register B	Register CH
4	Register E	Register DL
5	Register D	Register DH
6	Register L	Register BL
7	Register H	Register BH
8	IOBYTE	IOBYTE

* function code = 90H for BDOS, 40H to 90H for BIOS

8.1 IOBYTE

Since Z80 transients may change the IOBYTE value directly, it is a parameter on every BDOS/BIOS call (so that the 8088's BIOS has the same value). On return, it is replaced (possibly with a new value).

8.2 Cold and Warm Boot

The BIOS BOOT and WBOOT calls are not expected to return to the Pseudo BDOS/BIOS routines.

8.3 BIOS SECTRAN

The BIOS SECTRAN call is not serviced by the 8088. Since no CP/M-86/80 BIOS data fields are updated nor any I/O takes place, the Pseudo BIOS handles this call directly to reduce disk latency time.

8.4 Reply Latency

For both BIOS and BDOS there are functions which do not wait for reply, such as PRINT and CONOUT. In these cases, the Z80 signals the 8088 to perform the function while the Z80 resumes processing. This reduces Z80 reply latency.

8.5 Console Status Checks

Console status checks are stored in shared memory, so that BIOS and BDOS functions that need this information can access it without calling the 8088.

9.0 Z80CCP.SYS

Z80CCP is loaded by the CP/M-86/80 CCP whenever a Z80 transient is requested and when the system must be reconfigured before loading and executing an 8088 transient. The code for this module is contained in the Z80CCP.ASM file and in the INCLUDEd file Z80SVC.LIB.

Z80CCP is passed the following parameters:

```
ES = Segment base of CP/M-86/80
BX = Offset to parameter block
```

Where parameter block =

```
bytes 0-1  Offset to FCB of requested transient

      2    Current disk drive number

      3    Boot drive number
```

Return codes are:

AX = 0 System has been reconfigured; run .CMD file
AX = 0 Error

Possible error conditions are:

1. The File PRMTVPVT.SYS Not Found on Boot Disk

2. The File Z80.SYS Not Found on Boot Disk

3. CANNOT LOAD .COM FILE (.COM file is too large for available TPA)

9.1 Z80 Transient System Configuration

The system must be reconfigured if the Pseudo BDOS/BIOS routines are not in memory (PCPMADR_OFFSET). These routines, with the Interface Layer and Primitive Routines, are loaded into high Z80 memory from Z80.SYS. Next, the pointer/buffers data block and the CP/M-86/80 operating system are moved. All appropriate tables and pointers are reset. Control is then transferred to the new Interface Layer (by the Start-Z80 function -- see Section 6.1) and to the new CP/M (by resetting the interrupt vector segments, the CALLF return segment value, and the stack segment). The system is now configured as shown in Figure 1-2 or Figure 1-3. The Z80 transient must now be loaded. See Section 9.3.

9.2 8088 Transient System Configuration

A new copy of the Interface Layer/Primitive Routines is loaded in private Z80 memory from PRMTVPVT.SYS. Next, the pointers/buffers data block and the CP/M-86/80 operating system are moved. All appropriate tables and pointers are reset and control is transferred to the new modules. (See Section 9.1.) The system is now configured as shown in Figure 1-1. Control is returned to the CCP for the loading and execution of the .CMD transient.

9.3 Executing a Z80 Transient

Before executing a Z80 transient, the base page must be initialized and the transient must be loaded. After execution begins (under control of the Z80), the 8088 must wait in a service loop for BDOS/BIOS calls.

9.3.1 **Initialization for Z80 Transients** - The following base page locations are initialized:

0 - 2	Jump to Pseudo BIOS warm boot
3	IOBYTE
4	Current disk drive
5 - 7	Jump to location containing a "Jump to Pseudo BDOS"
5C - 7C	File Control Block

In addition, the "jump to Pseudo BDOS" location is initialized.

Locations 5 - 7 do not contain a direct jump to Pseudo BDOS because some transients use the jump address to determine the size of the TPA. A direct jump to BDOS could terminate CP/M on a 64K bytes system. Therefore, a jump is taken to the end of the TPA and from there a jump is taken to the Pseudo BDOS.

A dummy disk parameter header area (PSDPH) is also initialized with absolute pointers to the translate table area (PSTRN), disk parameter block area (PSDPB), and the allocation vector (PSALV).

9.3.2 **Code Relocation** - Before the Z80 transient can be loaded, the code that performs the loading and the code for the service routine must be relocated into private 8088 memory. Therefore, this code cannot exceed 400H bytes.

9.3.3 **Loading and Executing the .COM File** - Before reading the .COM file, a Z80-running-flag is set so that it will appear to the BIOS that the Z80 is requesting the read. This causes the records to be read directly into Z80 memory. After the Z80 file is loaded, a Start-Z80 function is performed and the 8088 falls through to the waiting service routine.

9.3.4 **Z80 Service Routine** - This routine begins execution when a Z80 transient program calls a BIOS or BDOS routine through the Pseudo BIOS/BDOS. It determines if any "pre-conditioning" of the BDOS parameters is necessary (BDJMPS table), does the actual BDOS call, and then performs any necessary "post-conditioning" (BDYJMPS table). Z80 BIOS calls are serviced through the 8088 BDOS Direct BIOS call (function #50). Upon completion of the BDOS call, the Z80 is interrupted as an indication that the request has been serviced. The service routine then waits for another BDOS/BIOS call from the Z80.

Special consideration is given to the following:

IOBYTE

If the IOBYTE value has changed, a BDOS Set I/O Byte call (function 8) is made.

BDOS System Reset (Function 0)
BIOS Init (Function 0)

The Z80-running-flag is reset and BDOS call 0 is made with memory released. Control is returned to the CCP.

BDOS Direct Console I/O (Function 6)

CP/M-80 uses X'FF' for status and input (instead of X'FE' and X'FF'). If a character is ready, it is read. If not, 0 is returned.

BDOS Set I/O Byte (Function 8)

The returned IOBYTE value is saved and returned to the Z80.

BDOS Get Console Status (Function 11)

CP/M-80 uses a non-zero value of X'FF' instead of X'01'.

BDOS Return Login Vector (Function 24)

CP/M has register A = register L for backward compatability.

BDOS Calls using Buffer Addresses (Functions 9, 10, 15, 16, 17, 19, 20, 21 22, 23, 30, 33, 34, 35, 36, and 40)

If necessary, data in the buffer is moved to a shared memory buffer. The offset of the shared memory buffer is given to BDOS. Upon return, the data is moved into the Z80 buffer. FCB buffers are 36 bytes long; string buffers are defaulted to 128 bytes. The Read Console Buffer call (Function #10) is limited to 126 input bytes.

BDOS calls returning Allocation Vector pointer (Function 27)

The Allocation Vector data is moved into a shared memory buffer (PSALV) and a pointer to the shared memory buffer is returned to the Z80.

BDOS calls returning Disk Parameter Block pointer (Function 31)

The Disk Parameter Block data is moved into a shared memory buffer (PSDPB) and a pointer to the shared memory buffer is returned to the Z80.

BIOS calls returning Disk Parameter Header pointer (Function 50)

Upon return from the BDOS Direct BIOS call, the translate table data, the disk parameter block data, and the allocation vector data are moved into a shared memory buffer (PSTRN, PSDPB, PSALV) and a pointer to the shared memory buffer (PSDPH) is returned to the Z80.

BIOS SECTRAN (Function 16)

This call is handled by the Pseudo BIOS (see Section 8.4).

10.0 SYSTEM DISK GENERATION

This description of CP/M-86/80 system disk generation for the Rainbow 100 computer is divided into five parts:

1. Tools and files required for system disk generation
2. Preparation and placement of files on system tracks
3. Preparation and placement of System files on data tracks
4. Tools required for the generation of Rainbow 100 utilities
5. Utility file preparation

10.1 System Disk Generation Requirements

The following tools and source/hex files are required to generate a system disk:

Tools

ASM86.CMD	LOAD.COM
DDT.COM	MAC.COM
DDT86.CMD	PIP.CMD
GENCMD.CMD	SAVE.CMD
LDCOPY.CMD	Z80.LIB

Source/Hex Files

BOOT100.ASM	DEFBUF.LIB
CAT.LIB	DRIPATCH.A86
CPLBIOS.A86	LDBDOS.H86
CPLBIOS1.A86	PRIVATE.ASM
CPLBIOS2.A86	SHARED.ASM
CPLBLOK.LIB	Z80BASE.ASM
CPLDBIOS.A86	Z80CCP.A86
CPLLDCPM.A86	Z80CODE.ASM
CPLPATCH.A86	Z80SVC.LIB
CPM.H86	

10.2 Preparation of System Files

The appropriate disk drive is indicated by "d."

10.2.1 **BOOT100.SYS -**

1. Use the BOOT100.ASM file

2. MAC BOOT100 $AdHdPd+S

3. d:DDT BOOT100.HEX
   -M1000,11FF,100
   ^C
   SAVE 2 d:BOOT100.SYS

NOTE

BOOT101.HEX must be on the logged-in drive.

10.2.2 **BOOT101.SYS -**

1. Use the BOOT101.ASM file

2. MAC BOOT101 $AdHdPd+S

3. d:DDT BOOT101.HEX
   -M1000,11FF,100
   ^C
   SAVE 2 d:BOOT101.SYS

NOTE

BOOT101.HEX must be on the logged-in drive.

10.2.3 **LOADER.CMD -**

1. Use CPLLDCPM.A86 with DEFBUF.LIB (on the same drive)
   CPLDBIOS.A86 with DEFBUF.LIB (all on the same drive)
   CPLBIOS1.A86
   CPLBLOK.LIB
   CPLBIOS2.A86
   CAT.LIB
   LDBDOS.H86

2. ASM86 CPLLDCPM $Ad Hd Pd Sd

3. ASM86 CPLDBIOS $Ad Hd Pd Sd

4. PIP d:LOADER.H86=d:CPLLDCPM.H86,d:LDBDOS.H86,
   d:CPLDBIOS.H86

5. GENCMD d:LOADER 8080

6. ERA d:LOADER.H86

10.2.4 **Z80BASE.COM -**

1. Use the Z80BASE.ASM file
2. MAC Z80BASE $AdHdPd+S
3. LOAD d:Z80BASE

10.2.5 **PRMTVPVT.SYS -**

1. Use the PRIVATE.ASM and
   Z80CODE.ASM files
2. PIP d:PRMTVPVT.ASM=d:PRIVATE.ASM,d:Z80CODE.ASM
3. AMC PRMTVPVT $AdHdPd+S
4. ERA d:PRMTVPVT.ASM
5. LOAD PRMTVPVT
6. REN d:PRMTVPVT.SYS=PRMTVPVT.COM
   or PIP d:PRMTVPVT.SYS=d:PRMTVPVT.COM

10.2.6 **File Placement: System Tracks -**

1. d:DDT86
   -F400:0,27FF,1A
   -WDUMMY.LDX,400:0,400:27FF

NOTE

All files read and written with the 'R' and 'W' commands must be on the logged-in drive.

```
-RDUMMY.LDX
   START      END
 [S0]:0000 [S0]:27FF
-RBOOT100.SYS
   START      END
 [S1]:0000 [S1]:01FF
-RLOADER.CMD
   START      END
 [S2]:0000 [S2]:[L2]
-RZ80BASE.COM
   START      END
 [S3]:0000 [S3]:[L3]
-RPRMTVPVT.SYS
   START      END
 [S4]:0000 [S4]:[L4]
-RBOOT101.SYS
   START      END
[S5]:0000 [S5]:[L5]
```

NOTE

S0, S1, S2, S3, S4, L2, L3, L4 are supplied by the DDT86 'R' commands above.

```
-M[S1]:0,1FF,[S0]:0
-M[S2]:80,127F,[S0]:200
-M[S5]:0,[L5],[S0]:1400
-M[S2]:1280,[L2],[S0]:1600
-M[S3]:0,[L3],[S0]:2200
-M[S4]:0,[L4],[S0]:2300
-WLOADFILE.LDX,[S0]:0,[S0]:27FF
^C
```

2. ERA d:DUMMY.LDX

3. LDCOPY d:LOADFILE.LDX

10.3 Preparation of System Files for Data Tracks

The appropriate disk drive is indicated by "d."

10.3.1 CPM.SYS -

1. Use DRIPATCH.A86
   CPLBIOS.A86 with DEFBUF.LIB (all on the same drive)
   CPLBIOS1.A86
   CPLBLOK.LIB
   CPLBIOS2.A86
   CAT.LIB
   CPLPATCH.A86 with DEFBUF.LIB (on the same drive)
   CPM.H86

2. ASM86 DRIPATCH $Ad Hd Pd Sd

3. ASM86 CPLBIOS $AD Hd Pd Sd

4. ASM86 CPLPATCH $Ad Hd Pd Sd

5. PIP d:CPMSYS.H86=d:CPM.H86,d:DRIPATCH.H86,
   d:CPLBIOS.H86,d:CPLPATCH.H86

6. GENCMD d:CPMSYS 8080 CODE[A40]

7. ERA d:CPMSYS.H86

8. REN d:CPM.SYS=CPMSYS.CMD
   or PIP d:CPM.SYS=D:CPMSYS.CMD

10.3.2 **Z80CCP.SYS -**

1. Use Z80CCP.A86 with Z80SVC.LIB (all on the same drive)
   DEFBUF.LIB

2. ASM86 Z80CCP $Ad Hd Pd Sd

3. GENCMD d:Z80CCP CODE[A3D0] DATA[A450]

4. REN d:Z80CCP.SYS=Z80CCP.CMD

10.3.3 **Z80.SYS -**

1. Use the SHARED.ASM and
   Z80CODE.ASM files

2. PIP d:Z80SYS.ASM=d:SHARED.ASM,d:Z80CODE.ASM

3. MAC Z80SYS $AdHdPd+S

4. ERA d:Z80SYS.ASM

NOTE

Z80SYS.HEX must be on the logged-in drive.

5. d:DDT
   -IZ80SYS.HEX
   -R0B00
   ^C
   SAVE 7 d:Z80.SYS

10.3.4 **PRMTVPVT.SYS** - See Section 10.2.4 above.

10.3.5 **Placement of Files: Data Tracks** - CPM.SYS must be the first file following the directory (required for the SERIAL utility). Z80CCP.SYS, Z80.SYS, and PRMTVPVT.SYS are placed after CPM.SYS (order is unimportant).

10.4 Utility File Preparation

10.4.1 **LDCOPY.CMD -**

1. Use the LDCOPY.A86 file

2. ASM86 LDCOPY $Ad Hd Pd Sd

3. GENCMD d:LDCOPY

10.4.2 **SAVE.CMD -**

1. Use the SAVE.A86 file
2. ASM86 SAVE $Ad Hd Pd Sd
3. GENCMD d:SAVE 8080 CODE[A40]

11.0 **SUBMIT.CMD MODIFICATION**

A patch has been applied to the SUBMIT.CMD file supplied by Digital Research Inc. using the DDT86 utility. This modification causes the SUBMIT utility to accept the user-selected Rainbow 100 boot drive as the controlling drive instead of automatically defaulting to drive A.

The following scenario depicts the patch being made to SUBMIT.CMD with DDT86.

```
A>DDT86

-RSUBMIT.CMD

     START           END
[SEG]: 0000     [SEG]:  0F7F

-SA
[SEG]: 000A  AD      AF
[SEG]: 000B  00      .
-SE
[SEG]: 000E  AD      AF
[SEG]: 000F  00      .
-A97
[SEG]: 0097  JMP     F60
[SEG]: 009A  .
-AF60
[SEG]: F60   PUSH    DS
[SEG]: F61   XOR     AX,AX
[SEG]: F63   MOV     DS,AX
[SEG]: F65   MOV     DS,[382]
[SEG]: F69   MOV     AL,[24B7]
[SEG]: F6C   POP     DS
[SEG]: F6D   INC     AL
[SEG]: F6F   MOV     [139],al
[SEG]: F72   CALL    F0
[SEG]: F75   JMP     9A
[SEG]: F78   .
-W SUBMIT.CMD
```

12.0 LISTINGS

The following PIP command lines may be used to obtain listings (the appropriate disk drive is indicated by "d:"):

```
PIP<cr>

*LST:=d:BOOT100.PRN
*LST:=d:CPLLDCPM.LST
*LST:=d:CPLLDCPM.SYM[P
*LST:=d:CPLDBIOS.LST
*LST:=d:CPLDBIOS.SYM[P
*LST:=d:Z80BASE.PRN
*LST:=d:PRMTVPVT.PRN
*LST:=d:DRIPATCH.LST
*LST:=d:DRIPATCH.SYM[P
*LST:=d:CPLBIOS.LST
*LST:=d:CPLBIOS.SYM[P
*LST:=d:CPLPATCH.LST
*LST:=d:CPLPATCH.SYM[P
*LST:=d:Z80CCP.LST
*LST:=d:Z80CCP.SYM[P
*LST:=d:Z80SYS.PRN
*LST:=d:COPYDSK.PRN
*LST:=d:FORMAT.PRN
*LST:=d:LDCOPY.LST
*LST:=d:LDCOPY.SYM[P
*LST:=d:MAINT.PRN
*LST:=d:BDOS86.PRN
*LST:=d:R100VID.PRN
*LST:=d:MAINT.MAP
*LST:=d:SAVE.LST
*LST:=d:SAVE.SYM[P
*LST:=d:SERIAL.PRN
^C
```

Rainbow™ 100
CP/M–86/80 Software Design
and Maintenance Manual
AA–P309A–TV

READER'S COMMENTS

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

☐ First-time computer user
☐ Experienced computer user
☐ Application package user
☐ Programmer
☐ Other (please specify)

Name

Date

Organization

Street

City

State

Zip Code
or Country

Do Not Tear – Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

Do Not Tear – Fold Here and Tape