Banked and Portable Basic I/O System

(B/P Bios)

Harold F. Bower
7914 Redglobe Ct.
Severn, MD 21144-1048

Cameron W. Cotrill
2935 Manhattan Ave
La Crescenta, CA 91214

Both Authors maintain accounts and may be contacted on the Ladera Z-Node, Al Hawley, SYSOP, at (310) 670-9465. Al Hawley has been gracious in offering Ladera services as the central repository for the official versions, and support clearinghouse for the latest B/P Bios support information and utilities.

B/P Bios is an original work architected by Cameron W. Cotrill, with support methodology, routines, and "grunt work" by Harold F. Bower.

This manual was edited on several 8-bit machines using the operating system described herein with both ZDE and WordStar 4.0. It was formatted using WordStar 7.0 on a 386 clone and printed on an Epson LQ-570 printer.

Acknowledgements: Many thanks to the testers of preliminary versions of B/P Bios for their faith, feedback and suggestions. These hearty souls include; Paul Chidley, designer of the YASBEC, Ian Cottrell, Jim Thale, and Larry Moore who all participated in banked testing of the YASBEC Version; and Terry Hazen, designer of the MDISK memory expansion board for the Ampro Little Board who validated B/P Bios on that platform.

TABLE OF CONTENTS

1.0 Introduction.

The Banked and Portable (B/P) Basic I/O System (BIOS) is an effort to standardize many of the logical to physical mapping mechanisms on Microcomputers running Z-Systems with ZSDOS. In expanding the capabilities of such systems, it became apparent that standard BIOSes do not contain the functionality necessary, adequate standardization in extended BIOS calls, nor an internal structure to fully support external determination of system parameters. B/P Bios provides a method of achieving these goals, while also possessing the flexibility to operate on a wide range of hardware systems with a much smaller level of systems programming than previously required.

1.1 About This Manual.

Documentation on B/P Bios consists of this manual plus the latest addendum on the distribution disk in the file **README.2ND**. This manual is divided into the following sections:

o *The Features of B/P Bios* summarizes the significant features of B/P Bios in general, highlighting advantages and the few limitations in the system.

o *Tailoring B/P Bios* contains details on altering the many options to generate a customized .REL file tailored to your system.

o *Installing a B/P Bios* details the installation of B/P Bios in both *Unbanked* and *Banked* configurations in a "how to" fashion.

o *Programming for B/P Bios* describes the interfaces, data structures and recommended programming practices to insure the maximum benefit and performance from systems with B/P Bios.

o *The B/P Bios Utilities* describes the purpose, operation, and customization of all supplied B/P Bios utilities and support routines.

o *Appendices* which summarize various technical information.

o A *glossary* defining many technical terms used in this Manual.

o An *index* of key words and phrases used in this Manual.

For those not interested in the technical details, or who want to bring the system up with a pre-configured version as quickly as possible, Section 4, *Installing a B/P Bios*, will lead you through the installation steps needed to perform the final tailoring to your specific computer. Other chapters cover details of the individual software modules comprising the B/P Bios, and specifics on the utilities provided to ease you use of this product.

1.2 Notational Conventions

Various shorthand terms and notations are used throughout this manual. Terms are listed in the Glossary at the end of this manual.

Though the symbols seem cryptic at first, they are a consistent way of briefly summarizing program syntax. Once you learn to read them you can tell at a glance how to enter even the most complicated commands.

Several *special symbols* are used in program syntax descriptions. By convention, square brackets ([]) indicate optional command line items. You may or may not include items shown between brackets in your command, but if you do not, programs usually substitute a default value of their own. If items between brackets are used in a command, all other items between the brackets must also be used, unless these items are themselves bracketed.

All of the support utilities developed to support the B/P Bios system contain built-in help screens which use the above conventions to display helpful syntax summaries. Help is always invoked by following the command with two slashes (//). So for example,

```
ZXD //
```

invokes help for ZXD, the ZSDOS extended directory program. Interactive ZSDOS programs such as BPCNFG2 also contain more detailed help messages which appear as a session progresses.

Many utilities may be invoked from the command line with *options* which command the programs to behave in slightly different ways. By convention, options are given after other command parameters. For example, the P option in the command

```
ZXD *.* P
```

causes the ZXD directory utility to list all files (*.*) and send its output to the printer (P). For convenience, a single slash character (/) can often be used in place of leading parameters to signify that the rest of the command line consists of option characters. Therefore, the command

```
ZXD /P
```

is identical in meaning to the previous example (see 6.23 for more on ZXD).

1.3 What is B/P Bios?

B/P Bios is a set of software subroutines which directly control the chips and other hardware in your computer and present a standard software interface to the Operating System such as our ZSDOS/ZDDOS, Echelon's ZRDOS, or even Digital Research's CP/M 2.2. These routines comply with the CP/M 2.2 standards for a Basic IO System (BIOS) with many extensions; some based on CP/M 3.x (aka CP/M

Plus), and others developed to provide necessary capabilities of modern software. When properly coded, the modules comprising a B/P Bios perform with all the standard support utilities, nearly all Z-System utilities, and most application programs without alteration.

The ability to operate Banked, Non-banked and Boot System versions of the Bios with a single suite of software, across a number of different hardware machines, plus the maximization of Transient Program Area for application programs in banked systems are features which are offered by no other system of which we are aware.

1.3 The History of B/P Bios.

Our earlier work developing ZSDOS convinced us that we needed to attack the machine-dependent software in Z80-compatible computers and develop some standard enhancements in order to exercise the full potential of our machines. This premise is even more true today with large Hard Disks (over 100 Megabytes) being very common, needs for large RAM Drives, and an ever shrinking Transient Program Area. Attempts to gain flexibility with normal operating systems were constrained by the 64k addressable memory range in Z80-compatible systems, and forced frequent operating system changes exemplified by NZCOM and NZBLITZ where different operating configurations could be quickly changed to accommodate application program needs.

In the mid to late 1980's, several efforts had been made to bank portions of CP/M 2.2 "type" systems. XBIOS was a banked Bios for only the HD64180-based MicroMint SB-180 family. While it displayed an excellent and flexible interface and the ability to operate with a variety of peripherals, it had several quirks and noticeably degraded the computer performance. A banked Bios was also produced for the XLM-180 single board S-100 computer, but required special versions of many Z-System utilities, and was not produced in any significant quantity. Other spinoffs, such as the Epson portable, attempted banking of the Bios, but most failed to achieve our comprehensive goals of compatibility with the existing software base, high performance, and portability.

In 1989, Cam developed the first prototype of B/P Bios in a Non-banked mode on his TeleTek while Hal concentrated on extending ZSDOS and the Command Processor. As of mid-1992, B/P Bios has been installed on:

YASBEC -	Z180 CPU, FD1772 FDC, DP8490 SCSI, 1MB RAM
Ampro LB w/MDISK -	Z80 CPU, FD1770 FDC, MDISK 1MB RAM
MicroMint SB-180 -	HD64180 CPU, SMS9266 FDC, 256KB RAM
Compu/Time S-100 -	Z80 CPU, FD1795 FDC, 1 MB RAM
Teletek -	Z80 CPU, NEC765 FDC, 64KB RAM

2 Features of B/P Bios.

B/P BIOS is designed to be completely compatible with the CP/M 2.2 standards for a Basic IO System, as well as to provide many extensions needed for banked memory which is becoming so prevalent with newer systems and processors. Additionally, strict coding standards used in the various modules forming the BIOS ease interface problems with applications programs and provide a more robust framework for future development. The extensions added to the basic CP/M 2.2 foundation include many elements from Digital Research's CP/M 3 (aka CP/M Plus), but in a more logically consistent manner. Also included in banked versions are provisions for managing up to 8 MB of extended memory for banked applications, RAM Drives and potentially multitasking in future versions. To provide insight into the methodology used, let us now examine some of the features in a generic B/P Bios.

2.1 Character IO.

As defined by Digital Research in their CP/M 2.2 standards, character IO consisted of logical devices referred to as TTY, CRT, UC1, CON, etc. B/P Bios extends and generalizes these interfaces using the IOBYTE to define four physical devices called COM1, COM2, PIO and NUL. The first two, COM1 and COM2, are serial ports; PIO is a Parallel port, while NUL is a "bit-bucket" which can be replaced by a customized driver, or used in lieu of an actual device. Digital Research provided only a limited interface capability to the character devices in CP/M 2.2, consisting of a Console (CON), an auxiliary Input and Output (RDR/PUN), and a Printer (LST). The ability to sense Input and Output Status with these devices was extremely limited and was enhanced in CP/M 3. These enhanced capabilities are completely incorporated into B/P Bios with the addition of strict register usage so that only relevant registers may be altered in the respective routines. By manipulating the IOBYTE, any of the four physical devices may be used in the three logical devices of CONsole, AUXiliary, and Printer (LST).

Also featured in B/P Bios are modifications of CP/M 3 functions to initialize (or re-initialize) all devices and parameters, and return the address of a table which contains names and parameters of the defined character devices. While not totally compatible with CP/M 3 equivalents, these functions are consistent with the spirit and functionality needed with this advanced system. Included in the device table are; flags defining whether the device is capable of Input, Output or Both, Data rates for serial devices (Maximum and Set), Serial data format where applicable, and Handshaking method (CTS/RTS, XON/XOFF or None), as well as Input and Output Data masks for stripping unneeded bits from characters during IO.

2.2 Mass Storage IO.

All versions of Digital Research's CP/M BIOSes define only a generic Disk driver with implementations of Floppy, Hard, RAM and Tape drives left to the user or developer. In B/P Bios, we went several steps further to ease many problems. First, we retained all standard CP/M 2.2 functions and parameters. added CP/M 3 features for returning the Disk Parameter Header (DPH) table address, and flushing of the software deblocking code segment, and added a new vector to the BIOS jump table to provide a standard method of directly addressing low-level device functions. Several standard low-level Floppy Disk functions are supported and used by the standard utilities, including a function to return the type of Disk Controller in use which permits a single support utility to adapt to a wide variety of hardware platforms. In a like manner, low-level functions are provided for SCSI/SASI Hard Disk drives, and provisions for RAM Disk drives in the event special hardware is implemented. The methods used to implement these access mechanisms may be logically extended to handle Tape Drives or Network Interfaces.

2.3 Clock Support for Time and Date.

Many Hardware vendors have added provisions for Time and Date as non-standard extensions to CP/M 2.2 BIOSes, and more have incorporated such support into CP/M 3 BIOSes. We opted to define the CP/M 3 clock vector as a ZSDOS-standard clock building on our previous Operating System work. This entry point into the Bios completely complies with our ZSDOS standards and can completely replace the separate clock driver when used with ZSDOS. For systems capable of returning tenths-of-seconds, such as the YASBEC and SB-180, the standard has been enhanced to support this capability as well.

2.4 Banked Memory Support.

While Digital Research added banked memory support to their CP/M 3, it was in a manner incompatible with Bios interface standards defined for earlier CP/M standards. The method used in B/P Bios is compliant with CP/M 2.2 in direct accessing of Bios functions with only one minor exception when using the Banked ZSDOS2, and contains many of the CP/M 3 extensions added for banked memory support, with some being modified to be consistent with standards adopted for Z-System software. The exception to CP/M 2.2 accesses occurs when the Operating System can access certain buffers in the System Memory Bank. With ZSDOS 2, Allocation Bit Buffers (ALV), Check Buffers (CSV), and the Disk Host Buffer are all contained in the System Bank and not directly accessible from Transient Programs. To compensate for this, we have added a command to ZSDOS 2 to return the free space on disks (the most common reason for accessing these buffers) and tailored several utilities to adapt to banked and non-banked systems.

In addition to the primitives initiated by Digital Research, we added functions to directly access Words and Bytes in extended banks of memory, Directly accessing software routines contained in alternate memory banks, and properly managing the system when errors occur. These features make B/P Bios much more robust and resilient than other products. These features are implemented by methods transparent to the system utilities so that the same functions are available in both banked and non-banked versions.

2.5 Other Features.

B/P Bios contains a standardized identification method which may be used to determine the hardware on which the software is operating. This allows applications to "adapt" to the environment in a manner similar to that used in the rest of the Z-System community. It also minimizes system "crashes" by executing programs which assume certain hardware features which may be detrimental if executed on other systems. The effects of identification of physical system parameters is most readily noticed by virtue of a single suite of support programs performing low-level functions such as formatting and diagnostics which function across widely differing hardware platforms. Portability on this scale can rarely be seen in other computer systems.

The ZCPR 3.4 Environment with extensions is mandatory in a B/P Bios system. Beginning with the addition of System Segment address and size information for CPR, DOS and BIOS which were added in the ZCPR 3.4 Environment. B/P Bios also adds a Resident User Space which may be used to locate unique routines for custom applications in a manner similar to, but more consistent than NZ-COM. An Environment Version number of 90H identifies the Z3 Environment as being compliant with B/P definitions.

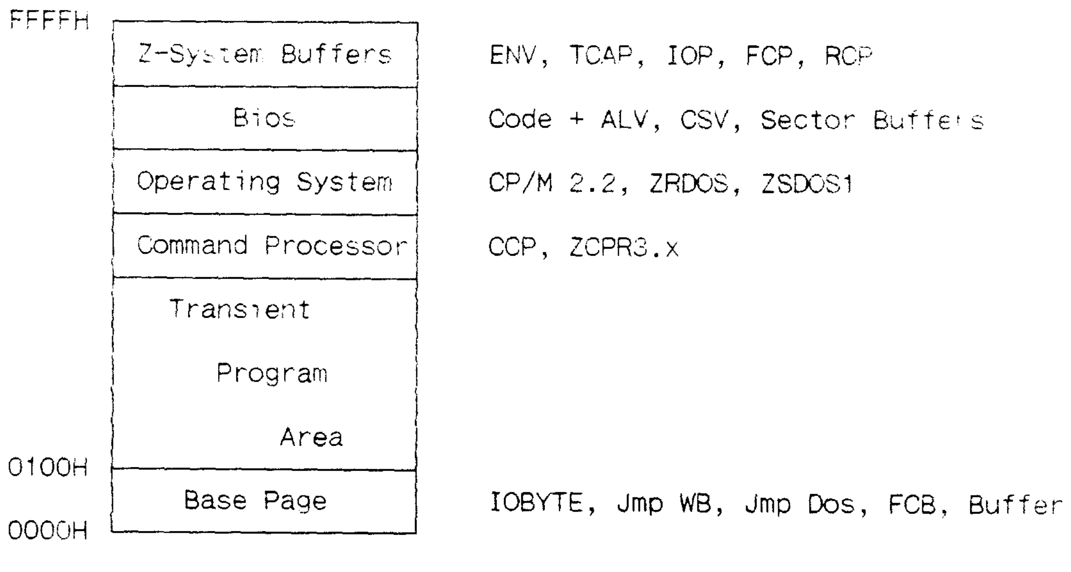
In Banked systems, application programs may also be placed in alternate memory banks using location and sizing information contained at standard positions within the Bios Header Structure. This feature permits significantly greater functionality without sacrificing precious Transient Program Area. While the scheme employed in the initial distribution is subject to minor adjustments as the banked ZSDOS2 becomes more firmly developed, experimentation and suggestions into this realm are encouraged.

3. Tailoring a B/P Bios.

To customize a B/P Bios for your use, or adapt it to a new hardware set, you will need an editor and an assembler capable of producing standard Microsoft Relocatable files. Systems using the Hitachi HD64180 or Zilog Z180 must be assembled with either ZMAC or SLR180 which recognize the extended mnemonic set, or with a Z80 assembler and MACRO file which permits assembly of the extended instructions. For Z80 and compatible processors, suitable assemblers include ZMAC and Z80ASM. For any assembler, failure to produce standard Microsoft Relocatable code will preclude the ability of our Standard utilities to properly install B/P Bios systems.

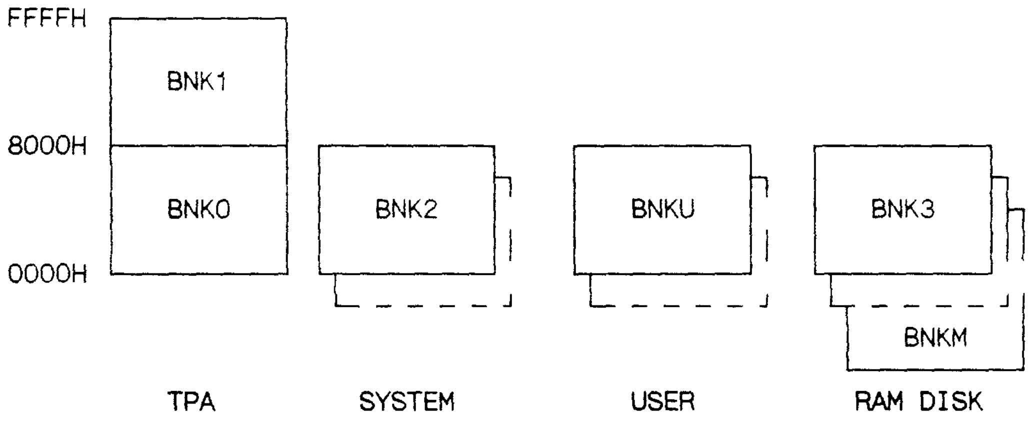
3.1 Theory of Operation.

In order to understand the need for, and principles behind B/P Bios, you must understand the way in which CP/M 2.2, as modified by the Z-System, uses the available memory address space of a Z80 microprocessor. For standard versions of CP/M and compatible systems, the only absolute memory addresses are contained in the *Base Page* which is the range of 0 to 100H. All addresses above this point are variable (within certain limits). User programs are normally run from the Transient Program Area (TPA) which is the remaining space after all Operating System components have been allocated. The following depicts the assigned areas pictorially along with some common elements assigned to each memory area:



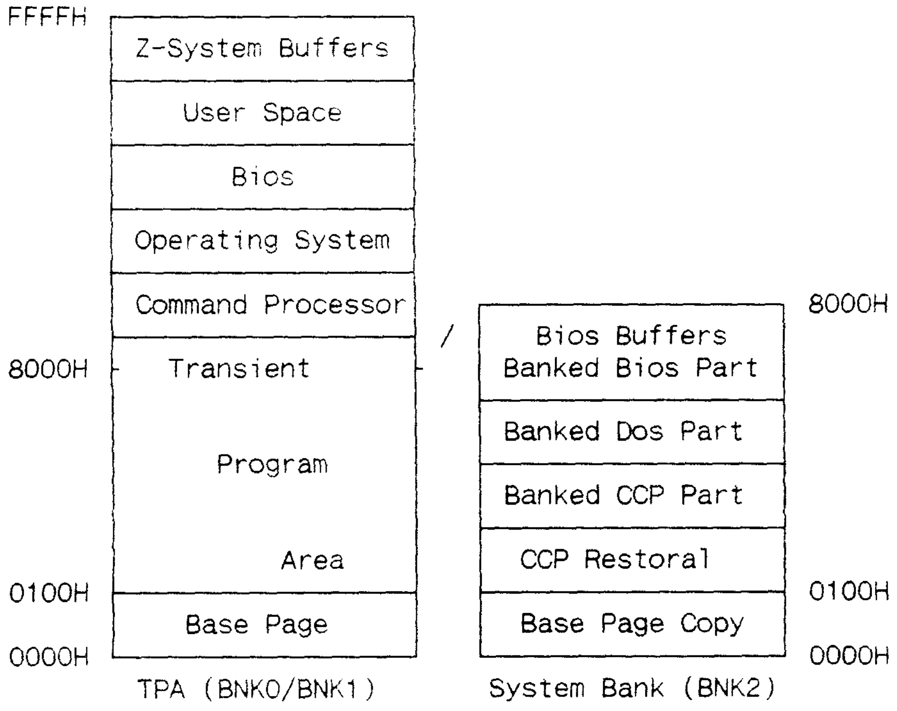
As more and more functionality was added to the Z-System Buffers, bigger drives were added using more ALV space, and additional functionality was added to Bios code in recent systems, the available TPA space has become increasingly scarce.

B/P Bios attacks this problem at the source in a manner which is easily adaptable to different hardware platforms. It uses additional memory for more than the traditional role of simple RAM Disks, it moves much of the added overhead to alternate memory banks. The generic scheme appears pictorially as:



As can be seen from the above diagram, multiple banks of memory may be assigned to different functional regions of memory, with each 32k bank (except for the one defined as BNK1) being switched in and out of the lower 32k of the processor's memory map. The bank defined as BNK1 is ***ALWAYS*** present and is referred to as the *Common Bank*. This bank holds the portions of the Operating System (Command Processor, Operating System, BIOS, and Z-System tables) which may be accessed from other areas, and which therefore must always be "visible" in the processor's memory. It also contains the code to control the Bank switching mechanisms within the B/P Bios.

To illustrate this functional division, the memory map of a basic B/P Bios system is divided as:



The B/P Bios banking concept defines a one byte Bank Number permitting up to 8 Megabytes to be directly controlled. Certain assumptions are made in the numbering scheme, the foremost of which is that BNK0 is the lowest physical RAM bank, BNK1 is the next incremental RAM bank, with others follow in incrementing sequential order. A couple of examples may serve to illustrate this process. The YASBEC is offered with a couple of options in the Memory Map. Units with the MEM-1, 2 or 3 decoder PALs assign the first 128k bytes of physical memory to the Boot ROM, so BNK0 is set to 4 (Banks 0-3 are the ROM). The MEM-4 PAL only uses the first 32k (Physical Bank 0) for the ROM which means that BNK0 is assigned to 1, BNK1 to 2 and so on up to the 1 Megabyte maximum where BNKM is 31.

The Ampro Little Board equipped with MDISK, on the other hand, completely removes the Boot ROM from the memory map leaving a maximum of 1 MB of contiguous RAM space. In this system, BNK0 is set to 0 and BNKM to 31 of a fully equipped 1 MB MDISK board.

The region beginning after BNK1 is referred to as the System Bank. It begins at the bank number assigned to BNK2 and ends at the bank number immediately before that assigned to the User Bank, BNKU if present, or BNK3 if no User Bank area is defined.

If present, one or more 32k banks of memory may be defined with the BNKU equate for unique user programs or storage areas. This area begins with the bank number set to the label and ends at the bank number immediately before the BNK3 label. BNK3 defines a high area of physical memory which is most often used for a RAM Disk providing fast temporary workspace in the form of an emulated disk drive.

B/P Bios contains protection mechanisms in the form of software checks to insure that critical portions of the memory map are enforced. In the case of Non-banked systems, a check is made to insure that the system size is not so great that the Bios may overwrite reserved Z-System areas in high memory (RCP, IOP, etc). If a possible overflow condition is detected, the message

++ mem ovfl ++

will be issued when the system is started. In Banked Bios systems, this message will be displayed if the top of the system portions in the SYStem Bank exceeds the 32k bank size. For most systems, this space still permits drives of several hundred megabytes to be accommodated.

Since the Common portions of the operating system components must remain visible to applications, a similar check is made to insure that the lowest address used by the Command Processor is equal to or greater than 8000H. This factor is checked both in both MOVxSYS and BPBUILD with either a warning issued in the case of the former, or validity checks on entry in the case of the latter.

3.2 B/P Bios Files.

This BIOS is divided into a number of files, some of which depend highly on the specific hardware used on the computer, and some of which are generic and need not be edited to assemble a working system. Much use is made of conditional assembly to tailor the resulting Bios file to the desired configuration. The Basic file, BPBIO-xx.Z80, specifies which files are used to assemble the Bios image under the direction of an included file, DEF-xx.LIB. It is this file which selects features and contains the Hardware-dependent mnemonic equates. By maintaining the maximum possible code in common modules which require no alterations, versions of B/P Bios are relatively easy to convert to different machines. The independent modules used in the B/P Bios system are:

BOOTRAM.Z80	- (only needed in BOOT ROM applications)
BOOTROM.Z80	- (only needed in BOOT ROM applications)
BYTEIO.Z80	- Character IO per IOBYTE using IIO-xx routines
DEBLOCK.Z80	- Disk Deblocking routines
DPB.LIB	- 3.5/5.25" Floppy Format Definitions (if AutoSelect)
DPB8.LIB	- 8"/Hi-Density Floppy Format Definitions (if AutoSelect)
DPB2.LIB	- Additional Floppy Definitions (optional if AutoSelect)
DPBRAM.LIB	- Fixed Floppy Format Definitions (if Not AutoSelect)
DPH.LIB	- Disk Parameter Header Table & Floppy definitions
FLOPPY.Z80	- Floppy Disk High-Level Control
SECTRAN.Z80	- Sector Translate routines
SELFLP1.Z80	- Floppy Select routine (if Not auto selecting)
SELFLP2.Z80	- Floppy Select routine (if auto selecting)
SELRWD.Z80	- Generic Read/Write routines
Z3BASE.LIB	- ZCPR 3.x file equate for Environment settings

Other files are hardware version dependent to varying extents. These modules requiring customization for different hardware systems are given names which end with a generic "-xx" designator to identify specific versions. Tailoring these modules ranges from simple prompt line customization to complete re-writes. Versions of B/P Bios generated to date are identified as:

"-18"	- MicroMint SB-180	(64180 CPU, 9266 FDC, 5380 SCSI)
"-YS"	- YASBEC	(Z180 CPU, 1772 FDC, DP8490 SCSI)
"-AM"	- Ampro Little Board	(Z80 CPU, 1770 FDC, 1MB MDISK)
"-CT"	- Compu/Time S-100 board set	(Z80 CPU, 1795 FDC, 1MB Memory)
"-TT"	- Teletek	(Z80 CPU, 765 FDC)

Files associated with specific hardware versions or require tailoring are:

BPBIO-xx.Z80	- Basic file, tailored for included file names
CBOOT-xx.Z80	- Cold Boot routines, Sign-on prompts
DEF-xx.LIB	- Equates for option settings, mode, speed, etc.
DPBHD-xx.LIB	- Hard Drive Partition Definitions (optional)
DPBM-xx.LIB	- Ram Drive Definition (optional)
DPHHD-xx.LIB	- Hard Drive DPH definitions (optional)
DPHM-xx.LIB	- Ram Drive DPH Definition (optional)
FDC-xx.Z80	- Floppy Disk Low-Level interface/driver routines
HARD-xx.Z80	- Hard Drive Low-Level interface/driver routines (optional)

IBMV-xx.Z80	- Banking Support Routines (if banked)
ICFG-xx.Z80	- Configuration file for speed, Physical Disks, etc
IIO-xx.Z80	- Character IO definitions and routines
RAMD-xx.Z80	- Ram Drive interface/driver routines (optional)
TIM-xx.Z80	- Counter/Timer routines and ZSDOS Clock Driver
WBOOT-xx.Z80	- Warm Boot and re-initialization routines

3.3 B/P Bios Options.

The most logical starting point in beginning a configuration is to edit the DEF-xx.LIB file to select your desired options. This file is the basic guide to choosing the options for your system, and some careful choices here will minimize the Bios size and maximize your functionality. Some of the more important options and a brief description of them are:

MOVCPM - Integrate into MOVCPM "type" loader? If the system is to be integrated into a MOVCPM system, the Environment descriptor contained in the CBOOT routine is always moved into position as part of the Cold Start process. If set to *NO*, a check will be made to see if an Environment Descriptor is already loaded, and the Bios copy will not be loaded if one is present.

NOTE: When assembling a Bios for Boot Track Installation (MOVCPM set to YES), many options are deleted to conserve space and the Bios Version Number is forced to 1.1.

BANKED - Is this a banked BIOS? If set to *YES*, the Bank control module, IBMV, is included in the assembly, and much of the code is relocated to the system bank. Note that a Banked system CANNOT be placed on the System Tracks, or integrated into a MOVCPM image.

IBMOVS - Are Direct Inter-Bank Moves possible? If set to *YES*, direct transfer of data between banks is possible such as with the Zilog Z180/Hitachi 64180. If *NO*, a 256-byte transfer buffer is included in high Common Memory and Inter-bank moves require transfer of bytes through this buffer.

ZSDOS2 - Assemble this for a Banked ZSDOS2 system? If *YES*, the ALV and CSV buffers will be placed in the System bank invisible to normal programs. This has the side effect that many CP/M programs which perform sizing of files (Directory Listers, DATSWEEP, MEX, etc) which do not know about this function will report erroneous sizes. The advantage is that no sacrifice in TPA is required for large Hard Disks. Set this to *NO* if you want strict CP/M 2.2 compatibility.

FASTWB - Restore the Command Processor from the System Bank RAM? If set to *YES*, Warm Boots will restore the Command Processor from a reserved area in the System RAM bank rather than from the boot tracks. For the maximum benefit of B/P Bios, always attempt to set this to *YES*. In systems without extended memory, it MUST be set to *NO*.

MHZ - Set to Processor Speed in closest even Megahertz (e.g. for a 9.216 MHz clock rate, set to 9). The value entered here is used in many systems to compute Timing values and/or serial data rate parameters.

CALCSK - Calculate Diskette Skew Table? If *NO*, a Skew table is used for each floppy format included in the image. Calculating Skew is generally more efficient from a size perspective, although slightly slower by factors which are so small as to be practically unmeasurable.

HAVIOP - Include IOP code into Jump table? If the IOPINIT routine satisfies your IOP initialization requirements, you may turn this off by setting to *NO* and save a little space. This typically will be turned off when generating a system for MOVCPM integration to conserve space.

INROM - Is the Alternate Bank in ROM? Set to *NO* for Normal Disk-based systems. Please contact the authors if you need additional information concerning ROM-based system components.

BIOERM - Print BIOS error messages? Set this to *YES* if you desire direct BIOS printing of Floppy Disk Error Messages. If you are building a BIOS for placement on Boot Tracks, however, you will probably not have room and must turn this Off. Set to *NO* to simply return the normal Success/Fail error flag with no Message printout.

FLOPY8 - Include 8"/Hi-Density Floppy Formats? Some systems (SB-180, Compu/Time) can handle both 5.25" and 8" disks. If your hardware supports the capability and you want use 8" disks as well as the normal 3.5 and 5.25" diskettes, setting this to *YES* will add formats contained in DPB8.LIB and control logic to the assembly. Future systems may take advantage of the "High-Density" 3.5 and 5.25" Floppy Disks which use higher data rates. Their definitions will be controlled by this flag as well.

NOTE: If AUTOSL is set to *NO*, this option will probably cause the BIOS to be larger than necessary since these additional formats may not be accessible.

MORDPB - Use more Floppy DPB's (in addition to normal 4-5.25" and optional 8")? If *YES*, the file DPB2.LIB is included. Many of the formats are Dummies and may be filled with any non-conflicting formats you desire.

NOTE: If AUTOSL if set to *NO*, this option will probably cause the BIOS to be larger than necessary since these additional formats may not be accessible.

MORDEV - Include Additional Character Device Drivers? Is set to *YES*, user-defined drivers are added to the Character IO table, and associated driver code is assembled. Systems featuring expansion board such as the SB-180 and YASBEC may now take advantage of additional serial and parallel interfaces within the basic Bios. Set to *NO* to limit code to the basic 4 drivers.

NOTE: When assembling a Bios for Boot Track Installation (MOVCPM set to YES), MORDEV is overridden to conserve space, and the Bios Version Number is forced to 1.1 in the distribution files.

BUFCON - Use type ahead buffer for the Console? If set to *YES*, code is added to create and manage a type-ahead buffer for the driver assembled as the console. This device will be controlled by either interrupts (in systems such as the YASBEC and SB-180) or background polling (in Ampro and Compu/Time). This means that characters typed while the computer is doing something else will not be lost, but will be held until requested.

BUFAUX - Use type ahead buffer on Auxiliary Port? As with BUFCON above, setting to *YES* will add code to create and manage a type ahead buffer for the auxiliary device. Since the AUX port typically is used for Modem connections, buffering the input will minimize the loss of characters from the remote end.

AUTOSL - Auto-select floppy formats? If set to *YES*, selection of Floppy disks will use an algorithm in SELFLP2.Z80 to identify the format of the disk from the DPB files included (DPB.LIB, optional DPB8.LIB, and optional DPB2.LIB) and log the disk if a match is found. There must be *NO* conflicting definitions included in the various files for this to function properly. See the notes in the various files to clarify the restrictions. If set to *NO*, the single file DPBRAM.LIB is included which may be tailored to contain only the fixed format or formats desired per disk drive. This results in the smallest code requirement, but least flexibility.

RAMDSK - Include code for a RAM-Disk? If set to *YES*, any memory above the System or User bank may be used for a RAM Drive (default is drive M:) by including the file RAMD-xx.Z80. Parameters to determine the size and configuration are also included in the files DPHM-xx.LIB and DPBM-xx.LIB. In systems without extended memory, or to conserve space such as when building a system for the boot tracks, this may be disabled by setting to *NO*.

HARDDSK - Include SCSI Hard Disk Driver? Set to *YES* if you wish to include the ability to access Hard Disk Drives. In a floppy-only system, a *NO* entry will minimize BIOS code.

HDINTS - (System Dependent) In some systems such as the YASBEC, Interrupt-driven Hard Disk Controllers using DMA transfer capabilities may be used. If you wish to use this type of driver specified in the file HARDI-xx.Z80 instead of the normal polled routines included in HARD-xx.Z80, set this option to *TRUE*. In most cases, this driver will require more Transient Program Area since the Interrupt Handling routine must be in Common Memory.

CLOCK - Include ZSDOS Clock Driver Code? If set to *YES*, the vector at BIOS+4EH will contain a ZSDOS-compatible clock driver with the physical code contained in the TIM-xx.Z80 module. If set to *NO*, calls to BIOS+4EH return an error code.

TICTOC - (System Dependent) Use pseudo heartbeat counter? This feature is used in systems such as the Ampro Little Board and Compu/Time SBC880 which do not have an Interrupt scheme to control a Real Time Clock. Instead, a series of traps are included in the code (Character IO Status polls, Floppy Disk Status polls) to check for overflow of a 1-Second Counter. It is less desirable than an Interrupt based system, but suffices when no other method is available. Set to *NO* if not needed.

QSIZE - Size in bytes of type ahead buffers controlled by BUFCON and BUFAUX.

REFRSH - Activate Dynamic Refresh features of Z180/HD64180 processors? In some computers using these processors such as the YASBEC, refresh is not needed and merely slows down processing. Set to *NO* if you do not need this feature. If your processor uses dynamic memory, or needs the signal for other purposes (e.g. The SB180 uses Refresh for Floppy Disk DMA), Set this to *YES*.

Z3 - Include ZCPR init code? Since a Z3 Environment is mandatory in a B/P Bios (which now "owns" the Environment), this option has little effect.

For assembly of a Banked version of B/P Bios, the identification of various banks of memory must be made so that the various system components "know" where things are located. Refer to Section 3.1 above for a description of these areas. The BNK0 value should be the first bank of RAM in the System unless other decoding is done. The following equates must be set:

BNK0	- First 32k TPA Bank (switched in/out)
BNK1	- Second 32k TPA Bank (Common Bank)
BNK2	- Beginning of System Bank (BIOS, DOS, CPR) area
BNKU	- Beginning of Bank sequence for User Applications
BNK3	- Beginning of Extra Banks (first bank to use for RAM Disk)
BNKM	- Maximum Bank Number assigned

3.4 Configuration Considerations.

When assembling a version of B/P Bios for integration into an IMG file, size of the resulting image is not much of a concern, so you need not worry about minor issues of size. For integration into a system for loading onto diskette boot tracks, however, the limitation is very real in order to insure that the CPR/DOS/BIOS and Boot Sector(s) can fit on the reserved system tracks. Typically, a limit of slightly under 4.5k exists for the Bios component. When the MOVCPM flag is set to YES for this type of assembly, warnings will be issued when the image exceeds 4352 bytes (the maximum for systems with 2 boot records), and 4480 bytes (the maximum for systems with a single boot record). Achieving these limits often requires disabling many of the features.

The first thing you should do before assembling the BIOS is to back up the entire disk, then copy only the necessary files onto a work disk for any editing. After setting the options as desired, edit the hardware definitions in ICFG-xx.Z80 to reflect the physical characteristics of your floppy and hard drives, as well as any other pertinent items. Then edit the logical characteristics for your Hard and Ram Drives (if any) in DPBHD-xx.LIB and DPBM-xx.LIB. If you do not desire any of the standard floppy formats or want to change them, edit DPB.LIB and/or DPB2.LIB (if using auto selection) or DPBRAM.LIB if you are using fixed floppy formats. Finally edit the DPH files to place the logical drives where desired in the range A..P.

Decide whether you want to generate a system using the Image file construct developed in support of B/P Bios (BPBUILD/LDSYS), or for integration on a floppy disk's boot tracks. If the latter, you probably will not be able to have all options turned on. For example, with the MicroMint SB-180, the following options must be turned Off: BANKED, ZSDOS2, BIOERM, FLOPYS, MORDPB, BUFAUX and usually either CLOCK or RAMDSK. As an aid to space reduction, conditional assembly based on the MOVCPM flag automatically inhibits all but double-sided Floppy formats from DPB.LIB. If configuring for Floppy Boot tracks (MOVCPM flag set to TRUE), a warning will be printed during assembly if the size exceeds that available for a One or Two-sector boot record. Using the BPBUILD/LDSYS method, you may vary nearly all system parameters, even making different systems for later dynamic loading.

If you are using a version of the B/P Bios already set for your type of computer, you are now ready to assemble, build a system and execute it. The only remaining task would be an optional tailoring of the sign on banner in the file CBOOT-xx.Z80 and reassembly to a .REL file.

For those converting a standard version of the B/P Bios to a new hardware system, we recommend that you begin with a Floppy-only system in Non-Banked mode then expand from there. The easiest way to test out new versions is to use the System Image (IMG file) mode, then advance to boot track installations if that is desired. Enhancements that can be added after testing previous versions may be to add Hard Drives, RAM Drive, and finally Banking.

4 Installing a B/P Bios.

The Distribution diskette(s) on which B/P Bios is furnished are configured for booting from the vanilla hardware for the version ordered. A 9600 bps serial terminal is standard, and will allow you to immediately bring up a minimal non-banked floppy disk system. Due to the variety of different system configurations and size restrictions in some versions, only the Floppy Disk Mass Storage capability can be assured on the initial boot disk. Where space remained on the boot tracks, limited Hard Drive support is also provided, and in some configurations, even RAM Drive support exists.

After booting from either an established system, or the boot tracks of the distribution disk, format one or more fresh diskettes and copy the distribution diskette(s) contents to the backup diskette(s). Copy the boot tracks from the master to the copies using BPSYSGEN (see 6.6). Remove the master diskette(s) for safekeeping and work only with the copies you just made.

Using the backup diskette with the B/P utilities on it, execute BPCNFG in the Boot Track configuration mode (see 6.2), adjusting all the options to your specific operating environment. When you have completed tailoring the system, it is ready for booting by placing the diskette in drive A: and resetting the system.

The sample STARTUP.COM file on the distribution disk will automatically execute a sequence of instructions when the system is booted. It contains various instructions which further tailor the system and load portions of the operating system which are too big to fit on the boot tracks. The default instruction sequence is:

```
LDDS                             <-- Load the DateStamper style File
                                       Stamp routine and clock
LDR SYS.RCP,SYS.FCP,SYS.NDR      <-- Load ZCPR 3 Environment segments
                                       for Resident Command Processor,
                                       Flow Control Pkg and Named Dirs
IOPINIT                          <-- Initialize the IO Processor Pkg
TD S                             <-- Prompt for Date and Time. Set Clk
                                       Alternatives are to use TDD
                                        (6.21) or SETCLOK (6.18)
IF ~EX MYTERM.Z3T                <-- If the file MYTERM.Z3T does Not
                                       exist...
TCSELECT MYTERM.Z3T              <-- ..select which terminal you have
                                       creating a MYTERM.Z3T file
FI                               <-- ...end of the IF
LDR MYTERM.Z3T                   <-- Load the Terminal Definition data
```

If you wish to alter any of these initial instructions to, for example, initialize the RAM drive using INITRAMD, add File Time Stamp capabilities to it with INITDIR or PUTDS and copy some files there with COPY, these may be added with ALIAS, VALIAS, SALIAS or other compatible files available from the ZSYSTEM or ZCPR33 areas on Z-Nodes.

After the initial system is up and running from the Default Boot Track system, you may expand the operation by generating systems for different purposes in order to gain the most advantage from your system. Many types of installation are possible, the simplest of which is a Non-Banked system using only 64k of the systems memory, all of which is in primary memory. Such a system uses a normal Command Processor such as the ZCPR3.x family, and a Non-Banked Operating System such as our ZSDOS Version 1. Non-Banked systems may be installed on a Disk's Boot Tracks, or created as an Image File for dynamic loading using the LDSYS Utility (see 6.15).

Banked systems *MUST* be created with the BPBUILD Utility (see 6.1) and loaded with LDSYS (see 6.15). The techniques to manage different memory banks to form a complete Operating Environment are rather intricate and are best handled by our utilities. Many Image files may be created and loaded as needed to tailor your system for optimum performance. The following sections describe these various types of installations in detail.

4.1 Boot Track Installation.

For most of the existing CP/M compatible computers to begin executing a Disk Operating System, a program must be placed on a specified area of a Floppy or Hard Disk Drive. Normally, the first two or three tracks on the disk are reserved for this purpose and are referred to as the "Boot Tracks". Since the space so defined is generally restricted, neither a complete B/P Bios nor a banked installation is possible. Instead, a scaled-down system roughly equivalent to those currently in use is used to start the computer and serve as the Operating System, with larger systems loaded later as needed.

If you are using a pre-configured version of B/P Bios for your hardware, you may simply continue to use the Boot Track system from the distribution disk(s) by copying the system as described in Section 4 above using BPSYSGEN (see 6.6). If you elect to alter or otherwise customize the Boot Track system, you must assemble the B/P Bios source setting certain of the equates in the DEF-xx.LIB file to insure a correct type of system. To assemble a Boot Track system, the most important equates are:

MOVCPM	Set to *YES*
BANKED	Set to *NO*
ZSDOS2	Set to *NO*

One element of Banked Systems is available in a Boot Track installation if additional memory is available, and your B/P Bios routines support such a feature. This feature reloads the Command Processor from Banked memory instead of from the Boot Tracks of a disk, and generally produces less code (taking less space on the Boot Tracks) and executes faster. It is set with:

FASTWB	Set to *YES* if desired, *NO* if Warm Boot from disk

Some of the features that generally need to be disabled to scale a smaller system are set as:

MORDPB	Set to *NO*
DPB8	Set to *NO*
MORDEV	Set to *NO*

When at least these equates and any others you desire to change (see section 4) have been made to the component files of the system, assemble your BPBIO-xx file to a MicroSoft standard REL file. This output file may be used to overlay the Bios portion of the MOVxSYS.COM system generation utility (see 6.16) furnished with your distribution disk, or an equivalent program provided with your computer. MOVxSYS or its equivalent (MOVCPM, MOVZSYS, etc) is a special program customized for your particular hardware containing all the Operating System components which will be placed on the Boot Tracks, along with a routine to alter the internal addresses to correspond to a specified memory size.

To Add the new Bios you just assembled, execute INSTAL12 (see procedures in 6.13) specifying your computer's MOVxSYS or equivalent program and follow the prompts to overlay the new Bios. Once INSTAL12 has saved a relocatable or absolute file, you are ready to create a boot disk containing the modified system.

If you used the command **INSTAL12** to install system segments on MOVxSYS or equivalent program, you must first create an Absolute System Model file. Since the functional portion of your new program is identical to the original MOVxSYS or equivalent, use the method explained in your original documentation to generate a new system. With MOVxSYS, the command is:

```
        MOVxSYS nn *              <-- replace MOVxSYS with your version
```

Where **nn** is the size of the system (typically **51** for a moderate boot system). The asterisk tells the program to retain the image in memory and not write it to a disk file. You may now use BPSYSGEN to write the new image to the system tracks of your boot diskette. Do this by executing BPSYSGEN with no arguments and issue a single Carriage Return when asked for the source of the Image.

If you used the command **INSTAL12 /A** to install replacement system segments over a System Image file, or used a utility which wrote the new image to a disk file, use BPSYSGEN to write the image file to the system tracks of your boot disk. The proper command is

```
        BPSYSGEN filename
```

where **filename** is the name of the disk file you just created by executing MOVxSYS or equivalent with output to a disk file, or with INSTAL12 on an existing image file.

If the system is written to a Hard Disk, and your system supports booting from a Hard Disk such as the YASBEC, you normally must alter the default Boot Sector from the default Floppy Disk Boot Sector contained in MOVxSYS or equivalent. This alteration is accomplished by HDBOOT (see 6.9) which must be customized to the specific Hardware System used.

After the above actions have been completed as appropriate, tailor the Boot Track system to reflect the desired starting configurations with BPCNFG (see 6.2). Such items as the desired Startup file name, Bank Numbers (critical if FASTWB is used), and drive types and assignments are routinely tailored at this point. When the you have finished this step, test your new system by resetting the system, or cycling the power and you should be up and running!

4.2 Non-Banked Image Installation.

A Non-Banked system may be installed as an Image File as opposed to the basic Boot Track installation covered in 4.1 above. To create an Image File, you must have REL or ZRL versions of a Command Processor (ZCPR3.x or equivalent recommended), an Operating (ZSDOS.ZRL recommended), and a REL version of B/P Bios for your system assembled with the **MOVCPM** equate in DEF-xx.LIB set to *NO*. Other equates in this file may be set as described above for the Boot Track system. Since Image Files are not as constrained in size as is installation for Boot Tracks, more features may generally be activated such as Error Messages, RAM Drive, additional Hard Drive partitions, and complete Floppy Format suites. The main precaution here is that large Hard Drives will rapidly cause significant loss of Transient Program Area since all Drive parameters must be in protected high memory above the Bios.

After the Bios has been assembled, an Image file must be produced. This is accomplished with the BPBUILD Utility (see 6.1). Set the File names in Menu 1 to reflect only Non-Banked files (or minimally banked Bios if FASTWB is set to *YES*), and let BPBUILD do the work. Since the standard Non-Banked System segments are normally set to the "standard" CP/M 2.2 sizes, you may answer the "autosize" query with a *Y* to obtain the maximum Transient Program Area in the resulting system. When BPBUILD completes its work, a file, normally with the default type of .IMG, will have been placed in the currently logged Drive/User area and you are ready to perform the next step in preparation of the Non-banked Image.

As with the Boot Track installation covered above, several system items must be tailored before the Image may be safely loaded and executed. This is done by calling BPCNFG with the Image file name as an argument, or specify Image configuration from the interactive menu (see 6.2). Set all items as you desire them in the operating system, particularly the Bank Numbers (if FASTWB is active), and the Disk Drive characteristics and assignments. When this has been satisfactorily completed, you are ready to load and execute the newly-created system.

Installing an Image File (default file type of .IMG) is extremely easy. Only the utility LDSYS.COM (see 6.15) is needed. If the file type has not been changed from the default .IMG, only the basic name of the Image File need be passed to LDSYS when executed as:

```
        LDSYS IMGFILE           <-- Where IMGFILE is your Image file name
```

The operating parameters of the currently-executing system are first examined for suitability of loading the Image File. If it is possible to proceed, the Image File is loaded, placed in the proper memory locations, and commanded to begin execution by calling the B/P Bios Cold Boot Vector. The Cold Boot (Bios Function 0) performs final installation, displays any desired opening prompt and transfers control to the Command Processor with any specified Startup file for use by a ZCPR3.x Command Processor Replacement.

Since a non-banked Image File will probably closely resemble that contained on the Boot Tracks, the same STARTUP file may generally be used to complete the initial tailoring sequence. If a different file is desired, the Image File may be altered to specify a different file using BPCNFG.

4.3 Banked Bios, Non-banked System Installation.

With the B/P Bios system, an Image system may be created and loaded which places portions of the Bios *Only* in the System bank, retaining a non-banked Operating System and therefore maximum compatibility with existing applications software. A few thousand bytes can normally be reclaimed for Transient Programs in this manner, although large and/or increasing numbers of logical drives will still reduce TPA space because of the need to store Allocation Vector information in Common Memory.

To prepare such a system, simply edit the needed Bios files if necessary with particular emphasis on the DEF-xx.LIB file where the following equates must be set as:

MOVCPM	Set to *NO*
BANKED	Set to *YES*
ZSDOS2	Set to *NO*

Since banked memory *MUST* be available for this type of installation, you will probably want the Fast Warm Boot feature available to maximize system performance. To activate this option, set the following equate as:

FASTWB	Set to *YES*

When the editing is complete, assemble the Bios to a MicroSoft .REL file with an appropriate assembler such as ZMAC and build an Image system with BPBUILD (see 6.1) changing the Bios file name in menu 1 to the name of the newly created Bios file. Next, configure the default conditions if necessary with BPCNFG (see 6.2) and you are ready to activate the new system in the same manner as all Image files by calling LDSYS with the Image file argument as:

```
LDSYS BBSYS              <-- where BBSYS is your Image File Name
```

As with the completely Non-Banked system described above in Section 4.2, no new requirements are established for a Startup file over that used for the initial Boot System, since both the Command Processor and Disk Operating System are unbanked, and no data areas needed by application programs are placed in the System Bank. As with all Image Files, additional features such as full Bios Error Messages, more extensive Floppy Disk Formats and RAM drive may generally be included in the System definition prior to assembly since the size constraints of Boot Track systems do not apply.

4.4 Fully Banked Image Installation.

To create a system taking maximum advantage of banked memory, a special banked Operating System and Command Processor are needed. These have been furnished in initial form with this package as ZSDOS20.ZRL and Z40.ZRL respectively. They use the Banking features of B/P Bios and locate the maximum practicable amount of executable code and data in the System Bank. Of significant importance to maximizing the Transient Program Area is that the Drive Allocation Bit maps are placed in the System Bank meaning that adding large hard drives, or multiple drives produce only minimal expansion to the resident portion of the Bios.

A Fully banked Bios is created by editing the B/P Bios files as needed to customize the system to your desires. Insure that the following DEF-xx.LIB equates are set as:

MOVCPM	Set to *NO*
BANKED	Set to *YES*
ZSDOS2	Set to *YES*

Assemble the resultant B/P Bios to a MicroSoft .REL file. Build an Image file with BPBUILD (see 6.1) and configure the produced Image file with BPCNFG (see 6.2). When you are confident that all default settings have been made, activate the file by entering:

```
LDSYS FBANKSYS          <-- where FBANKSYS is your Image file name
```

Several differences may exist in the Startup file used for Fully banked systems. Generally the changes amount to deleting items such as a File Stamp module for the Non-banked ZSDOS1 which is not necessary with the fully-banked ZSDOS 2 and Z40. Only the type of clock need be specified for ZSDOS2. Furthermore, since the Z40 Command Processor Replacement contains most commonly-used commands gathered from a number of Resident Command Processor (RCP) packages, there is normally no need to load an RCP. A simple Startup file found adequate during development of the fully-banked B/P system is:

```
ZSCFG2 CB                <-- Set ZSDOS 2 clock to Bios+4EH
LDR SYS.FCP,SYS.NDR      <-- Load ZCPR 3 Environment segments
                             for Flow Control and Named Dirs
IOPINIT                  <-- Initialize the IO Processor Pkg
TD S                     <-- Prompt for Date and Time, Set Clk
                             Alternatives are to use TDD
                              (6.21) or SETCLOK (6.18)
IF ~EX MYTERM.Z3T        <-- If the file MYTERM.Z3T does Not
                             exist...
TCSELECT MYTERM.Z3T      <-- ..select which terminal you have
```

```
                                     creating a MYTERM.Z3T file
FI                                   <-- ...end if the IF
LDR MYTERM.Z3T                       <-- Load the Terminal Definition data
```

Since the requirements for a fully-banked system differ significantly from a non-banked one, we recommend that you use a different name for the Startup file. For example, STARTUP.COM is the default name used with Boot Track systems for initial operation, and with Non-banked Image Files, while STARTB may be a suitable name for the script to be executed upon loading a fully-banked system. The name of the desired Startup file may be easily altered in either Boot Track or Image systems from Option 1 in BPCNFG (see 6.2).

An option available to start from a large Image File is to configure a Startup file for execution by the Boot Track system containing a single command. The command would simply invoke LDSYS with the desired Banked Image File as an argument such as:

```
          LDSYS BANKSYS          <-- Where BANKSYS.IMG is your Image file
```

In this case, none of the normal initialization sequences cited above would be executed by the Boot Track system, and only those contained in the Startup for BANKSYS.IMG would occur. Other options abound and are left to the community to invent new combinations and sequences.

4.5 In Case of Problems...

While We attempted to outline procedures for the majority of installations we considered feasible, there may be occasions where you inadvertently find yourself in a position where you seem to have lost the ability to get your system up and running.

PROBLEM: When loading an .IMG file with LDSYS, the screen displays the LDSYS banner, system addresses, and halts with the last screen displaying:
"...loading banked system".

-- **SOLUTION** Something is not set correctly in the Bios, since all lines after the last one displayed are printed from the newly-loaded Bios. One of the most common causes for this problem is incorrect bank number settings. Use the hidden selection in Menu 1 of BPCNFG (see 6.2) to verify that the correct bank numbers have been set for TPA and SYStem banks. Another common cause of this problem is incorrect settings for the Console port, or a setting in the IOBYTE which directs Console data to a device other than the one intended. Use Menu 2 BPCNFG to properly set the IOBYTE and the console parameters.

PROBLEM: You boot from or load a B/P Bios system from a Hard Drive, and immediately after starting, the system attempts to log onto Floppy Drive 0.

-- **SOLUTION:** The most common cause for this symptom is that the desired Hard Drive and Floppy Drive definitions were not swapped to define a Hard Drive Partition as the A: drive. Use BPCNFG (see 6.2), Menu 5 to exchange drives to the desired configuration. A similar situation may exist where a Hard Drive is activated immediately after booting when a Floppy drive is desired as the A: Drive.

PROBLEM: The computer seems to boot satisfactorily, but after a few programs or any program which executes a Warm Boot (or entering Control-C), the system goes into "Never-never Land" and must be reset.

-- **SOLUTION:** This symptom is most often caused by an inability to access and load the Command Processor. This is most probably caused by assembling B/P Bios with the FASTWB equate in DEF-xx.LIB set to *YES* when the system contains no extended memory, or incorrect settings of the Bank Numbers. To check Bank Number settings, use the hidden function in BPCNFG, Menu 1 (see 6.2).

PROBLEM: When doing a Cold Boot from a Hard Drive (from Power up or Reset), the system goes to a Floppy Drive *before* displaying the initial sign on messages, and remains logged on the Floppy.

-- **SOLUTION:** This is most often due to your forgetting to run the HDBOOT utility on the Hard Drive Boot system after applying it with BPSYSGEN. Normally, systems created with MOVxSYS contain a Floppy Disk Boot sector which will load the initial Operating System from a Floppy. HDBOOT (see 6.9) modifies this record on a specified Hard Drive Unit so that the Operating System is loaded from a Hard Drive. Run HDBOOT on the Desired Hard Drive, then use BPCNFG (see 6.2) to insure that the logical drives are positioned as desired (Menu 5).

PROBLEM: When Booting, the system console either doesn't display anything, or prints strange characters.

-- **SOLUTION:** This is most often due to incorrect settings for the current Console, most probably the Data rate, or CPU Clock Frequency. Boot from a good system, then use BPCNFG (see 6.2) to adjust the settings on the problem system. Pay particular attention to Menu 1 (CPU Clock Rate) and Menu 2 (IOBYTE and Serial Port Data Rates).

PROBLEM: When running a fully-banked system with ZSDOS 2, some programs seem to "hang" or "lock up" the system on exit.

-- **SOLUTION:** One of the most common sources of this symptom is with the application program where the author used code which assumes that the BDOS and Command Processor are of a certain size, or bear a fixed relationship to the addresses in page 0. You may experience this most often when using an IMG system built by answering *YES* to the Autosizing query in BPBUILD (see 6.1). To compensate for such ill-behaved programs, you may use a two-step build process as:

1) Use BPBUILD to create an IMG file answering *YES* to Autosizing on exit. This maximizes TPA placing the Resident Bios as high as possible in memory.

2) Execute BPBUILD again with an argument of the name you gave to the file

just created above. This loads the definition from the IMG file. Immediately exit with a Carriage Return, and answer *NO* to Autosizing, and *YES* to placing system segments at standard locations. This procedure keeps the Bios address constant, but will move the starting addresses of BDOS and Command Processor down, if possible, to simulate "standard" sizes used in CP/M 2.2.

5. Programming for B/P Bios.

For most existing purposes, programming for B/P Bios is no different than for standard CP/M 2.2 BIOSes. Even adapting CP/M 3 programs for a B/P Bios should present no great hurdle due to the close similarity retained with the corresponding extended functions. The power of a B/P Bios interface, however, is in using the combined features to produce portable software across a wide variety of hardware platforms by exercising all of the B/P Bios features in concert. This section describes the interfaces available to the programmer of a system using the B/P Bios, and the functions available to ease direct floppy and hard drive accesses for specialized programming in a consistent manner.

One of the architectural flaws which we considered in CP/M Plus was the odd way in which direct BIOS access was handled. We designed B/P Bios to be as compatible with CP/M 2.2 as possible, yet provide the expanded functionality needed in Banked applications. To that end, direct interface with BIOS calls follows CP/M 2.2 conventions as much as possible.

The following pages on programming assume some familiarity with the basic CP/M fundamentals, and with Z80/Z180 assembly language, since it is beyond the intent of this manual, and our literary writing skills, to present an assembly programming tutorial. Should you need additional assistance in this area, please refer to the annotated bibliography for reference material.

5.1 Bios Jump Table.

The BIOS Jump table consists of 40 Jumps to various functions within the BIOS and provides the basic functionality. It includes the complete CP/M 2.2 sequence, most of the CP/M 3 (aka CP/M Plus) entry points (although some differ in parameter ordering and/or register usage), and new entry points needed to handle banking in a consistent and logical manner.

Bios entry points consist of a Table of Absolute 3-byte jumps placed at the beginning of the executable Image. Parameters are passed to the Bios in registers as needed for the specific operation. To avoid future compatibility problems, some of the ground rules for Bios construction include: No alteration of Alternate or Index registers as a result of Bios calls, and all registers listed in the documentation as being Preserved/Unaffected MUST be returned to the calling program in their entry state. Bios entry points are:

Function 0 (xx00)	Cold Boot
Enter: None	Exit: None. Execution resumes at CPR Uses: All Registers

Execute Cold Start initialization on the first execution. The jump argument is later overwritten, and points to the IOP Device jump table. The reason for this is that code to perform the initialization is often placed in areas of memory which are later used to store system information as a memory conserva-tion measure. Attempts to re-execute the initialization code would then encounter data bytes instead of executable instructions, and the system would most assuredly "crash".

Among other functions performed during initial execution of the Cold Boot code are: Establishing an initial Z3 Environment if necessary, initializing any Z3 system segments such as an Extended Path, Flow Control Package, Named Direc-tory Buffer and such: setting system-specific values such as the locations of Allocation Vector buffers for RAM and Hard Drives: and executing the Device Initialization routine (see Function 21). The Cold Boot routine usually exits by chaining to the Warm Boot Function (Function 1) to set vectors on Page 0 of the TPA memory bank.

Function 1 (xx03)	Warm Boot
Enter: None	Exit: None. Execution returns to CPR Uses: All Registers

This function re-initializes the Operating System and returns to the Command Processor after reloading it from the default drive boot tracks, or banked memory if the Bios was assembled with the Fast Warm Boot option.

Unless altered by an ill-behaved Resident System Extension (RSX) or other operating transient program, the Warm Boot Vector at location 0 in memory points to this vector. Well-behaved programs will not alter this address but should, instead, alter the destination argument of the Jump vector in the Bios header. There is a singular exception to this in the case of NZCOM where the Warm Boot vector points to the NZBIOS, and Not the 'Real' Bios. In such a case, the address of the 'Real' Bios must be separately determined (See Func-tion 30).

Function 2 (xx06)	Console Input Status
Enter: None	Exit: A = 0FFH if Char Ready, NZ A = 0 if No Char Ready, Z Uses: AF

This function returns a flag indicating whether or not a character has been entered from the Console device selected by the IOBYTE on Page 0 of the TPA Bank. The return status is often used by Transient Programs to determine if the user has attempted to start or stop program execution.

Function 3 (xx09)	Console Input
Enter: None	Exit: A = Masked Input Character Uses: AF

This function waits for a character to be entered from the Console device selected by the IOBYTE on Page 0 of the TPA Bank, and returns it to the calling routine. According to strict CP/M 2.2 standards, the Most Significant bit of the input byte must be set to Zero, but this may be altered by the input mask for the Console Device.

Function 4 (xx0C)	Console Output
Enter: C = Character to send to Console	Exit: None. Uses: AF

This function sends a specified character to the Console Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask bits as specified in the Character Device Configuration for the device as an Output.

Function 5 (xx0F)	List Output
Enter: C = Character to send to List Dev (Printer)	Exit: None. Uses: AF

This function will send a specified character to the List Device (Printer) defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask it as specified in the Character Device Configuration for the Output device.

Function 6 (xx12)	Auxiliary Output
Enter: C = Character to send to Auxiliary Device	Exit: None. Uses: AF

This function will send a specified character to the Auxiliary Output Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask it as specified in the Character Device Configuration for the Output device.

Function 7 (xx15)	Auxiliary Input
Enter: None	Exit: A = Masked Input Character Uses: AF

This function will read a character from the Auxiliary Input Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for a character to be received, and will mask it as specified in the Character Device Configuration for the Input device.

Function 8 (xx18)	Home Drive
Enter: None	Exit: None. Heads on selected drive moved to Track 0. Uses: All Primary Registers

This function will position the head(s) on the selected drive to Track 0. In B/P Bios, This operation performs no useful action, and is simply a Return. Pending Write purges and head repositioning is handled by the individual device drivers (Specifically Select Drive functions).

Function 9 (xx1B)	Select Logical Drive
Enter: C = Desired Drive (A=0..P=15)	Exit: (Success) A <> 0, NZ HL = DPH Address (No Drive) A = 0, Zero (Z) HL = 0 Uses: All Primary Registers

This function selects a specified logical drive as the current drive to which disk operations refer. If the operation is successful, the Disk Parameter Header (DPH) address is returned for later determination of the unit parameters. If the operation fails for any reason (non-existant drive, unknown or bad media, etc), a Zero value pointer is returned to signify that the drive cannot be accessed through the Bios.

Function 10 (xx1E)	Select Track
Enter: BC = Desired Track Number	Exit: None. Track Number saved Uses: No Registers

This function stores a specified Logical Track number for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes.

NOTE: While a 16-bit value is specified for this function, only the lower byte (8-bits) is used in most drivers.

Function 11 (xx21)	Select Sector
Enter: BC = Desired Sector Num	Exit: None. Sector Number saved Uses: No Registers

This function stores a specified Logical Sector Number for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes.

NOTE: While a 16-bit value is specified for this function, only the lower byte (8-bits) is used in all Floppy Disk and most Hard and RAM Disk drivers.

Function 12 (xx24)	Set DMA Address for Transfer
Enter: BC = Buffer Starting Addr	Exit: None. DMA Address saved Uses: No Registers

This Function stores a specified address to be used as the Source/Destination for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes. In banked systems, the Bank selected for the transfer may be altered by Function 28.

Function 13 (xx27)	Disk Read
Enter: None	Exit: A = 0, Z if No Errors A = Non-Zero if Errors, NZ Uses: All Primary Registers

This function reads a Logical 128-byte sector from the Disk, Track and Sector set by Functions 9-11 to the address set with Function 12. On return, Register A=0 if the operation was successful, Non-Zero if Errors occurred.

Function 14 (xx2A)	Disk Write
Enter: C = 1 for immediate write C = 0 for buffered write	Exit: A = 0, Z if No Errors A = Non-Zero if Errors, NZ Uses: All Primary Registers

This function writes a logical 128-byte sector to the Disk, Track and Sector set by Functions 9-11 from the address set with Function 12. If Register C=1, an immediate write and flush of the Bios buffer is performed. If C=0, the write may be delayed due to the deblocking.

Function 15 (xx2D)	List Output Status
Enter: None	Exit: A = 0FFH, NZ if ready for Output Character A = 0, Z if Printer Busy Uses: AF

This function returns a flag indicating whether or not the printer is ready to accept a character. It uses the IOBYTE on Page 0 of the TPA Bank to determine which physical device to access.

Function 16 (xx30)	Perform Sector Translation
Enter: BC = Logical Sector Num DE = Addr of Trans Table	Exit: HL = Physical Sector Num Uses: All Primary Registers

This function translates the Logical Sector Number in register BC (Only C used at present) to a Physical Sector number using the Translation Table obtained from the DPH and addressed by DE.

This ends the strict CP/M 2.2-compliant portion of the Bios Jump Table. The next series of entry Jumps roughly follows those used in CP/M 3, but with corrections to what we perceived to be deficiencies and inconsistencies in the calling parameters and structures.

Function 17 (xx33)	Console Output Status
Enter: None	Exit: A = 0FFH, NZ if Console Ready for output char A = 0, Z if Console Busy Uses: AF

This function returns a flag indicating whether or not the Console Device selected by the IOBYTE on Page 0 of the TPA Bank is ready to accept another output character.

Function 18 (xx36)	Auxiliary Input Status
Enter: None	Exit: A = 0FFH, NZ if Aux Input has character waiting A = 0, Z if No char ready Uses: AF

This function returns a flag indicating whether or not the Auxiliary Input selected by the IOBYTE on Page 0 of the TPA Bank has a character waiting.

Function 19 (xx39)	Auxiliary Output Status
Enter: None	Exit: A = 0FFH, NZ if Aux Output Ready for output char A = 0, Z if Aux Out Busy Uses: AF

This function return a flag indicating whether or not the Auxiliary Output selected by the IOBYTE on Page 0 of the TPA Bank is ready to accept another character for output.

Function 20 (xx3C)	Return Pointer to Device Table
Enter: None	Exit: HL = Addr of Device Table Uses: HL

This function roughly corresponds to an analogous CP/M Plus function although precise bit definitions vary somewhat. The Character IO table consists of four devices: COM1, COM2, PIO, and NUL. Each has an input and output mask, data rate settings and protocol flags. Not all defined settings (e.g. ACK/NAK and XON/XOFF handshaking, etc) may be fully implemented in each version, but are available for later expansion and use.

Function 21 (xx3F)	Initialize Devices
Enter: None	Exit: None. Initialization done Uses: All Primary Registers

This function initializes Character IO settings and other functions which may be varied by a Configuration Utility. It is an extended version of the corresponding CP/M Plus function. Its primary use is to restore IO configurations, system parameters such as clock rate, wait states, etc, after alteration by programs which directly access hardware such as many modem programs and the configuration utility, BPCNFG (see 6.2).

Function 22 (xx42)	Return DPH Pointer
Enter: None	Exit: HL = Address of start of Table of DPH Pointers Uses: HL

This function returns a Pointer to a table of 16-bit pointers to Disk Parameter Headers for Drives A-P. A Null (0000H) entry means that no drive is defined at that logical position.

Function 23 (xx45)	<Reserved for Multiple Sector IO>
Enter: None	Exit: None. Uses: No Registers

This function is Reserved in the initial B/P Bios release and simply returns.

Function 24 (xx48)	Flush Deblocker
Enter: None	Exit: None. Pending Disk Writes Executed. Uses: All Primary Registers

This function writes any pending Data to disk from deblocking buffers as mentioned in Function 14 above. This function should be called in critical areas where tasks are being swapped, or media is being exchanged when it is possible that the Operating System will not detect the change.

Function 25 (xx4B)	Perform Possible Inter-Bank Move

Enter: HL = Start Source Address
DE = Start Dest Address
BC = Number Bytes to Move
	Exit: None. Data is moved
	Uses: All Primary Registers

This function moves the specified number of bytes between specified locations. For banked moves, the Source and Destination banks must have been previously specified with an XMOVE call (function 29). Note that the B/P implementation of this function reverses the use of the DE and HL register pairs from the CP/M 3 equivalent function.

Function 26 (xx4E)	Get/Set Date and Time

Enter: DE = Start of 6-byte Buff
C = 0 (to Get Date/Time)
C = 1 (to Set Date/Time)
	Exit: A = 1 of Successful
	A = 0 if Error or No Clock
	Uses: All Primary Registers

This function provides an interface to programs for a Real-Time Clock driver in the Bios. The function uses a 6-byte Date/Time string in ZSDOS format as opposed to Digital Research's format used in CP/M Plus for this function. Also, This function must conform to additional requirements of DateStamper(tm) in that on exit, register E must contain the entry contents of (DE+5) and HL must point to the entry (DE)+5. If the actual hardware implementing the clock supports 1/10 second increments, the current 1/10 second count may be returned in register D.

Function 27 (xx51)	Select Memory Bank
Enter: A = Desired Memory Bank	Exit: None. Bank is in Context in range 0..7FFFH Uses: AF

This function selects the Memory Bank specified in the A register and make it active in the address range 0-7FFFH. Since character IO may be used when a bank other than the TPA (which contains the IOBYTE) is activated with this function, the B/P Bios automatically obtains the IOBYTE from the TPA bank to insure that Character IO occurs with the desired devices.

Function 28 (xx54)	Select Memory Bank for DMA
Enter: A = Memory Bank for Disk DMA Transfers	Exit: None. Bank Number saved for later Disk IO Uses: No Registers

This function selects a memory Bank with which to perform Disk IO. Function 12 (Set DMA Transfer Address) operates in conjunction with this selection for subsequent Disk IO.

Function 29 (xx57)	Set Source & Dest Banks for Move
Enter: B = Destination Bank Num C = Source Bank Number	Exit: None. Bank Nums saved for MOVE operation Uses: No Registers

This function sets the Source and Destination Bank numbers for the next Move (Function 25). After a Move is performed, the Source and Destination Banks are automatically reset to TPA Bank values.

This marks the end of the CP/M Plus "Type" jumps and begins the unique additions to the B/P Bios table to support Banking, Direct IO and interfacing.

Function 30 (xx5A)	Return BIOS Addresses

Enter: None	Exit: A = Bios Version (Hex)
	BC = Addr of Bios Base
	DE = Addr of Bios Config
	HL = Addr of Device Table
	Uses: All Primary Registers

This function returns various pointers to internal BIOS data areas and the Bios Version Number as indicated above. The Bios Version may be used to determine currency of the system software, and will be used by various support utilities to minimize the possibility of data corruption and/or as an indicator of supported features.

The Base Address of the Bios Jump Table returned in register BC is often used to insure that the proper indexing is achieved into the B/P data structures in the event that a Bios "shell" has been added such as when running NZCOM. While the Warm Boot jump at memory location 0000H normally points to the Bios Base+3, it is not always reliable, whereas this function will always return a true value with B/P Bios.

Registers DE and HL return pointers which are of value to programs which alter or configure various Bios parameters. The pointer to the configuration area of the Bios should be used in utilities as opposed to indexing from the start of the Bios Jump Table since additions to the Jump Table or insertion of other data will affect the Configuration Area starting address. The pointer in HL is available for use in systems which may contain more than four character IO devices. This pointer enables exchanges of devices to place desired devices in the first four positions of the table making them available for selection via the IOBYTE. After any alterations are made to the devices, a call to the Device Configuration Bios Function 21 should be made to activate the features.

FLOPPY DISK SUBFUNCTIONS.

Function 31 permits low-level access to Floppy and Hard Disks (via SCSI interface) by specifying a Driver Number and desired Function. While some hardware types do not support all of the parameters specified, particularly for Floppy Drives, this architecture supports all types, although specific systems may ignore certain functions. In this manner, for example, a single Format program supports NEC765, SMC9266, WD1770/1772/179x and other controller types with widely differing interfaces. Floppy Disk functions are accessed by entering a 1 value into Register B (Floppy Driver Number) and the desired function number in Register C, then jumping to or calling BIOS Entry jump number 31.

Function 31 (xx5D) Floppy SubFunction 0	Set Floppy Read/Write Mode

Enter: A = 0 for Double Density
FF for Single Density
B = 1 (Floppy Driver)
C = 0 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine establishes the Density mode of operation of the Floppy Disk Controller for Read and Write accesses. It assumes that SubFunctions 1 (Set Size and Motor) and 3 (Set Sector) have been called first.

Function 31 (xx5D) Floppy SubFunction 1	Set Floppy Disk & Motor Parms

Enter: A = 0 for 300 rpm (normal)
FF for 360 rpm (8"/HD)
D = FF for Motor Control,
0 if Motor always on
E = Disk Drive Size
B = 1 (Floppy Driver)
C = 1 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine establishes some of the physical parameters for a Floppy Drive. The normal 5.25" and 3.5" disk drives holding 400 or 800 kb or less rotate at 300 rpm. Many of the newer drives can increase this speed to 360 rpm which is the rate used on older 8" floppy drives. This is the speed used on the "High Density" 1.2 MB (IBM formatted) 5.25" drives. The A register is used to indicate the fastest speed capable on the specified drive. Register D is used to indicate whether the Motor is always On, or will start and stop periodically. This is normally used by the Bios to delay for a period before writing if the motor is stopped to allow the diskette to come up to speed thereby minimizing chances of data corruption. Register E is used to indicate the physical media size as: 0=Hard Disk, 001B=8" Drive, 010B=5.25" Drive, and 011B=3.5". Nothing is returned from this command.

While all of these functions may not be supported on any specific computer type, the interface from using programs should always pass the necessary parameters for compatibility.

NOTE: This routine assumes that SubFunction 2 (Set Head and Drive) has been called first. Call this routine before calling Function 0 (Set Mode).

Function 31 (xx5D) Floppy SubFunction 2	Set Head and Drive

Enter: A = Drive # (Bits 0,1)
Head # (Bit 2)
B = 1 (Floppy Driver)
C = 2 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine is entered with register A containing the Floppy unit number coded in bits 0 and 1 (Unit 0 = 00, 1 = 01 .. 3 = 11), and the Head in Bit 2 (0 = Head 0, 1 = Head 1). Nothing is returned from this function. Call this Subfunction before most of the others to minimize problems in Floppy accesses.

Function 31 (xx5D) Floppy SubFunction 3	Set Floppy Disk Mode

Enter: A = Physical Sector Number
D = Physical Sector Size
E = Last Sctr # on Side
B = 1 (Floppy Driver)
C = 3 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine establishes information needed to properly access a specified sector unambiguously with a number of different controller types. On entry, Register A contains the desired physical sector number desired. D contains the sector size where 0 = 128 byte sectors, 1 = 256 .. 3 = 1024 byte sectors, and E contains the last sector number on a side. Normally register E is unused in Western Digital controllers, but is needed with 765 and 9266 units. Nothing is returned from this subfunction.

Function 31 (xx5D) Floppy SubFunction 4	Specify Drive Times

Enter: A = Step Rate in milliSec
D = Head Unload Time in mS
E = Head Load Time in mS
B = 1 (Floppy Driver)
C = 4 (Subfunction #)
	Exit: None.
	
	Uses: AF

This subfunction set various timing values used for the physical drive selected. On entry, the A register contains the drive step rate in milliseconds. Within the Bios, this rate is rounded up to the nearest controller rate if the specified rate is not an even match. Register D should contain the desired Head Unload time in milliseconds, and E to the desired Head Load time in mS.

NOTE: With Western Digital type controllers, only the Step Rate is universally variable. In these systems, rates signaled by the Bios settings are rounded up to the closest fixed step rate such as the 2, 3, 5, or 6 milliSecond rates in the WD1772 or 6, 10, 20, or 30 milliSecond rates used in the older WD1770 and WD1795. Nothing is returned from this function.

Function 31 (xx5D) Floppy SubFunction 5	Home Disk Drive Heads
Enter: B = 1 (Floppy Driver) C = 5 (Subfunction #)	Exit: A = 0, Zero Set (Z) if Ok A <> 0, NZ if Errors Uses: AF NOTE: Subfcns 1, 2, & 4 Needed

This subfunction moves the head(s) on the selected drive to track 0 (home). Only success/failure is indicated by the value in the A register. No other registers may be altered by this function (especially BC).

NOTE: This function requires that Subfunctions 1 (Set Disk and Motor Parameters), 2 (Set Head and Drive) and 4 (Specify Drive Times) be called first in order to establish the physical characteristics of the Drive.

Function 31 (xx5D) Floppy SubFunction 6	Seek Track

Enter: A = Desired Track Number
D = 0FFH to Verify,
0 for No Verification
E = 0 for No Double-Step
<>0 for Double-Step
B = 1 (Floppy Driver)
C = 6 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	<> 0, NZ if Error
	
	Uses: AF
	
	NOTE: Subfcns 2, 3, & 4 Needed

This subfunction moves the head(s) for the selected drive to a specified track on the media. If the Double-Step flag (Register E) is set to a Non-Zero value, then the controller will issue two step pulses for every track increment or decrement which is required. After the Seek, a Read ID function will be performed to verify that the desired track was found if the Verification Flag (Register D) is set to a Non-Zero Number, preferably 0FFH. Only the AF registers may be altered by this function.

NOTE: This function requires that Subfunctions 2 (Set Head and Drive), 3 (Set Floppy Disk Mode) and 4 (Specify Drive Times) be called first in order to establish the physical characteristics of the Drive.

Function 31 (xx5D) Floppy SubFunction 7	Read Floppy Disk Sector

Enter: HL = Dest Buffer Address
B = 1 (Floppy Driver)
C = 7 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	<> 0, NZ if Error
	
	Uses: AF, HL
	
	NOTE: Subfcns 0,1,2,4 & 6 Needed

This subfunction Reads a physical sector of data from the selected drive and places it in the buffer at the specified address. It is important that an appropriately sized buffer is provided for this task. The Value in the A register will indicate the success or failure of the function as indicated in the above chart. Only the AF and HL registers may be altered by this function.

NOTE: This function requires that Subfunctions 0 (Set Read/Write Mode), 1 (Set Disk & Motor Parms), 2 (Set Head & Drive), 4 (Specify Drive Times) and 6 (Seek Track) be called first in order to establish the physical and logical characteristics of the data transfer.

Function 31 (xx5D) Floppy SubFunction 8	Write Floppy Disk Sector

Enter: HL = Source Buffer Address
B = 1 (Floppy Driver)
C = 8 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	A <> 0, NZ if Error
	
	Uses: AF,HL
	
	NOTE: Subfcns 0,1,2,4 & 6 Needed

This subfunction writes data from the buffer beginning at the specified address to the track, sector and head selected by other subfunctions. The value in the A register along with the setting of the Zero Flag will indicate whether the operation succeeded or not. Only the AF and HL registers may be altered by this function.

NOTE: This function requires that Subfunctions 0 (Set Read/Write Mode), 1 (Set Disk & Motor Parms), 2 (Set Head & Drive), 4 (Specify Drive Times) and 6 (Seek Track) be called first in order to establish the physical and logical characteristics of the data transfer.

Function 31 (xx5D) Floppy SubFunction 9	Read Disk Sector ID
Enter: B = 1 (Floppy Driver) C = 9 (Subfunction #)	Exit: A = 0, Zero Set (Z) if Ok A <> 0, NZ if Error Uses: AF NOTE: Subfcns 0 & 2 Needed

This Subfunction reads the first correct ID information encountered on a track. There are no entry parameters for this function other than the Driver and Subfunction number. A flag is returned indicating whether or not errors occurred. An error indicates that no recognizable Sector ID could be read on the disk. In most cases, this is due to an incorrect Density setting in the Bios.

NOTE: This function requires that Subfunctions 2 (Set Head & Drive) and 3 (Set Floppy Disk Mode) are called first in order to establish the physical characteristics of the disk.

Function 31 (xx5D) Floppy SubFunction 10	Return Floppy Drive Status
Enter: B = 1 (Floppy Driver) C = 10 (Subfunction #)	Exit: A = Status Byte of last Opn BC = FDC Controller Type HL = Address of Status Byte Uses: AF, BC, HL NOTE: Subfcn 2 Needed

This function returns the status of the currently-selected drive. There are no entry parameters for this function other than the Floppy Driver and Function number. On exit, the raw unmasked status byte of the drive, or the last operation depending on the controller type, is returned along with a binary number representing the FDC controller type (e.g. 765, 9266, 1772, etc).

NOTE: This routine assumes that Subfunction 2 (Set Head & Drive) has been called before this routine to select the Physical Parameters.

Function 31 (xx5D) Floppy SubFunction 11	Format Floppy Disk Track

Enter: HL = Pointer to Data Block
D = # of Sectors/Track
E = # of Bytes in Gap 3
B = 1 (Floppy Driver)
C = 11 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	A <> 0, NZ if Error
	
	Uses: AF,BC,DE,HL
	NOTE: Use Subfcn 10 for Cont Type

This Sub function formats a complete track on one side of a Floppy Disk. It assumes that the Mode, Head/Drive, Track, and Sector have already been set. On entry, HL points to data required by the controller to format a track. This varies between controllers, so RETDST should be called to determine controller type before setting up data structures. On entry, D must also contain the number of Sectors per Track, and E must contain the number of bytes to use for Gap 3 in the floppy format. On exit, A=0 and the Zero flag is Set (Z) if the operation was satisfactorily completed, A <> 0 and the Zero flag cleared (NZ) if errors occurred. This routine may alter all primary registers (AF,BC,DE,HL).

NOTE: This routine assumes that Subfunction 10 (Return Floppy Drive Status) has been called first to determine the Controller type and insert the correct information in the Format Data Block.

HARD DISK SUBFUNCTIONS.

These functions are available to directly access Hard Drives connected by a SCSI type interface. They are accessed by loading the desired function number in the C register, loading a 2 (SCSI driver) into the B register and calling or jumping to Jump number 31 in the Bios entry jump table. Since this interface is not as standardized as Floppy functions in order to handle SASI as well as SCSI devices, the interface has only basic functions with the precise operations specified by the User in the Command Descriptor Block passed with Function 2. While this places a greater burden on User programs, it allows more flexibility to take advantage of changing features in the newer SCSI drives.

Function 31 (xx5D) Hard Disk SubFunction 0	Set Hard Disk Addresses

Enter: DE = Address of Data Area
B = 2 (Hard Disk Driver)
C = 0 (Subfunction #)
	Exit: A = # Bytes in Comnd Block
	
	Uses: AF

This Subfunction sets the User Data Area Address for Direct SCSI I/O, and returns the number of bytes available in the SCSI Command Descriptor Block. The Data Area must be AT LEAST 512 bytes long and is used to store data to be written, and to receive data read from the selected drive. This Data Area size is mandatory since 512 bytes are always returned from a direct access in order to handle the wide variety of controller types recognized in the B/P Bios drivers. The number of bytes available in the Command Descriptor Block within the physical driver is usually 10 in order to handle the extended SCSI commands, but may be scaled back to 6 in limited applications.

Function 31 (xx5D) Hard Disk SubFunction 1	Set Physical & Logical Drive

Enter: A = Device Byte (5.2.1)
B = 2 (Hard Disk Driver)
C = 1 (Subfunction #)
	Exit: A = Physical Device Bit
	
	Uses: AF

This Subfunction sets the Physical Device bit in the Bios for SCSI accesses and the Logical Unit Number in the SCSI Command Block (Byte 1, bits 7-5). The format of the Device Byte provided to this routine is defined in the Configuration Data, Section 5.2.1, CONFIG+61, and is available from the Extended Disk Parameter Header at DPH-1. On exiting this routine, a byte is returned with a "One" bit in the proper position (Bit 7 = Device 7...Bit 0 = Device 0) to select the desired unit via a SCSI command.

Function 31 (xx5D) Hard Disk SubFunction 2	Direct SCSI Driver

Enter: DE = Ptr to Comnd Desc Blk
A = 0 if No Write Data
A = FF if Data to Write
B = 2 (Hard Disk Driver)
C = 2 (Subfunction #)
	Exit: A = Bit1 Status, Flags Set
	H = Message Byte Value
	L = Status Byte Value
	
	Uses: AF,BC,DE,HL
	NOTE: Subfcns 0 & 1 Needed

This Subfunction performs the actions required by the command in the specified Command Descriptor Block. The flag provided in Register A signifies whether or not user data is to be written by this command. If set to a Non-Zero value, Data from the area specified with Function 0 will be positioned for SCSI Write operations. At the end of the routine, 512 bytes are always transferred from the Bios IO Buffer to the Users Space set by Subfunction 0. This may be inefficient, but was the only way we could accommodate the wide variety of different SASI/SCSI controllers within reasonable code constraints. The status returned at completion of this function is the Status byte masked with the Check Bit, Bit 1. The full Status Byte and Message Byte from SCSI operations are also provided for more definition of any errors.

NOTE: This routine assumes that the Command Descriptor Block has been properly configured for the type of Hard Disk Controller set in B/P Bios, and that the selected disk is properly described (if necessary) in the Bios Unit definitions. Errors in phasing result in program exit and Warm Boot. It assumes the user has called Functions 0 (Set Hard Disk Addresses) and 1 (Set Physical & Logical Drives) before using this Subfunction.

Function 32 (xx60)	Set Bank for Far Jump/Call
Enter: A = Desired Bank Number	Exit: None. Uses: No Registers

This Function sets the bank number for a later Function 33 Jump to a routine in an alternate Memory Bank.

Function 33 (xx63)	Jump to (HL) in Alternate Bank
Enter: HL = Address to execute in Bank set w/Fn 32	Exit: <Unknown> Called routine sets return status Uses: All Primary Regs (assumed)

This Function switches to the bank number previously specified with Function 32, then calls the routine addressed by HL. Upon completion, operation returns to the bank from which called, and the address on the top of the stack.

Function 34 (xx66)	Clear Stack Switcher
Enter: HL = Addr to resume exec in entry bank	Exit: None. Execution resumes at addr in HL in entry bank Uses: No Registers

This Function is used for error exits from banked routines to return to the entry bank.

Function 35 (xx69)	Load A,(HL) from Alternate Bank
Enter: HL = Addr of desired byte C = Desired Bank Number	Exit: A = Byte from C:HL Uses: AF

This Function gets a byte (8-bits) from the specified Bank and Address. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 36 (xx6C)	Load DE,(HL) from Alternate Bank
Enter: HL = Addr of desired word C = Desired Bank Number	Exit: DE = Word from C:HL Uses: AF,DE

This Function gets a Word (16-bits) from the specified Bank and Address. The bank is temporarily switched in context for the access (if required). then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 37 (xx6F)	Load (HL),A to Alternate Bank
Enter: HL = Addr of Dest Byte C = Desired Bank Number A = Byte to save at C:HL	Exit: None. Byte stored at C:HL Uses: AF

This Function saves a Byte (8-bits) to the specified Address and Bank. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 38 (xx72)	Load (HL),DE to Alternate Bank

Enter: DE = Word to store at C:HL
HL = Addr of Dest Word
C = Desired Bank Number
	Exit: None. Word stored at C:HL
	
	Uses: AF

This Function saves a Word (16-bits) to the specified Address and Bank. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 39 (xx75)	Return Current Bank in Context
Enter: None	Exit: A = Bank currently active in Addr 0..7FFFH Uses: AF

This Function returns the Memory Bank currently in Context in the address range of 0..7FFFH. It may be used in the "where am I" role in application programs to track memory accesses.

5.2 Bios Data Structures.

5.2.1 Configuration Area.

Much of the ability to tailor B/P Bioses to your specific operating needs is due to the standardized location of many discrete elements of data, and a facility to easily locate and change them, regardless of the particular hardware platform in operation. Bios Function 30, Return Bios Addresses, reports the base address of the Configuration Area in the DE register pair. In this section, we will review each of the specified elements, their functions, and which parts of the data must be rigidly controlled to insure that the supplied utilities continue to function, as well as guarantee the portability of other programs.

CONFIG-6 - Bios ID. Character String, 6 bytes.

This character string MUST begin with the three characters "**B/P**" in Uppercase Ascii, followed by three Version-specific identifying characters. As of March 1993, the following identifiers have been assigned to systems:

"B/P-YS"	YASBEC
"B/P-AM"	Ampro Little Board 100
"B/P-18"	MicroMint SB-180
"B/P-CT"	Compu/Time S100 Board Set
"B/P-TT"	Teletek
"B/P-XL"	Intelligent Computer Designs XL-M180

CONFIG+0 - IOBYTE. Byte.

This byte contains the initial definition of the byte placed at offset 3 on the Base Page (0003H) during a Cold Boot and determines which of the four defined character IO devices will be used as the Console, Auxiliary and Printer devices. The default setting may be altered by BPCNFG to reflect changed device configurations, or by reassembly of the Bios.
The bit definitions in this byte are:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | | └─┴──── Console Device
    | | | | └─┴──────── Auxiliary Input Device
    | | └─┴──────────── Auxiliary Output Device
    └─┴──────────────── Printer Device
```

CONFIG+1 - System Drive. Byte.

This byte contains the drive which will be accessed after a Cold Boot and is assumed to contain the Operating System files. It is a binary value where A = 0, B = 1...P = 15.

CONFIG+2 - Bios Option Flags. **Byte.**
This byte consists of individually mapped bits which display which options are active in the assembled Bios. The bits listed as <reserved> should not be defined without prior coordination with the system developers to preclude conflicts with planned enhancements. The byte is currently defined as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | | | |
    | | | | | | | └──── 0 = Unbanked Bios     1 = Banked Bios
    | | | | | | └────── 0 = Bank in RAM       1 = Bank in ROM
    | | | | | └──────── 0 = DPBs Fixed        1 = DPBs Assignable
    | | | | └────────── 0 = ALV/CSV in TPA    1 = ALV/CSV in Bank (ZSDOS2)
    └─┴─┴─┴──────────── <reserved>
```

The next five bytes define the memory map of a banked system in 32k slices. For a complete description of Bank allocations, please refer to Section 4. In non-banked systems, all except the RAM Drive Bank should all be set to 0. If no memory is available for re-assignment as a RAM drive, this byte as well should be set to 0.

CONFIG+3 - User Bank. **Byte.**
This Byte reflects the Bank number reserved for User Applications.

CONFIG+4 - TPA Bank. **Byte.**
This Byte reflects the Bank number reserved for the Transient Program Area in the address range of 0..7FFFH. The next sequential bank number is normally the Common Bank which always remains in Context in the addressing range of 8000..FFFFH and contains the Operating System, Bios and Z-System tables.

CONFIG+5 - SYStem Bank. **Byte.**
This byte reflects the Bank number containing any executable code and data specified for the System Bank.

CONFIG+6 - RAM Drive Bank. **Byte.**
This byte reflects the starting Bank number available for use as a RAM Drive. It is assumed that all RAM from this Bank through the Maximum Bank Number is contiguous and available as a RAM Drive.

CONFIG+7 - Maximum Bank Number. **Byte.**
This byte reflects the number of the last available bank of RAM in the system. In many systems, it may be set to different numbers depending on the number of RAM chips installed in the system.

CONFIG+8 - Common Page Base. **Byte.**
This byte reflects the Base Page of the Common area in systems which do not fully comply with the 32k Memory Banking architecture of B/P Bios, but can be made somewhat compliant. This Byte must be AT LEAST 80H, but may be higher if needed.

CONFIG+9 - DPB Size. **Byte.**
This byte contains the length of Disk Parameter Block allocations within the Bios. Since more information is needed than the 15 bytes defined by Digital

Research in CP/M 2.2, an extended format is used. All re-assignments of Disk Parameter data should use this byte to determine the size of records.

CONFIG+10 - Number of DPBs in Common RAM. Byte.
CONFIG+11 - Number of DPBs in System Bank.. Byte.
These two bytes indicate the complete complement of Floppy Disk formats available within the Bios. In most cases, one of these two bytes will reflect a zero value with all Disk Parameter Blocks resident either in the Common area or in the System Bank. The provisions are available, however with these two bytes to split the definitions for custom versions without voiding the support tools provided.

CONFIG+12 - Pointer to first Common DPB. Word.
CONFIG+14 - Pointer to first Banked DPB. Word.
These two words point to the first DPB in a sequential list within the respective memory banks for Disk Parameter Blocks defined in the preceding bytes. In most cases one of these two words will be a Null pointer (0000H) corresponding to no data as described in the count bytes above.

CONFIG+16 - Initial Startup Command. String.
This string contains the first command which will be initiated on a Cold Boot. It is loaded into the Multiple Command Buffer defined in the Environment Descriptor (See 5.2.4) and calls a file of the specified name with a type of "COM". The string may have up to eight characters and must be Null-terminated (end with a Binary 0). The string is defined as:

Byte	- Number of Characters (0..8)
String	- 8 bytes for Ascii characters (usually Uppercase)
Byte	- Terminating Null (binary 0)

CONFIG+26 - Pointer to Environment Descriptor. Word.
This Word points to the first byte of an extended Z34 Environment which MUST begin on a Page boundary (xx00H). See Section 5.3.1 for a complete description of the Environment Descriptor and B/P Bios unique features.

CONFIG+28 - Banked User Flag/Bank Number. Byte.
This Byte may be used as a flag to indicate whether or not a User Bank is defined. Bank 0 cannot be used as a User bank by decree of the system authors. Therefore, if this byte contains a binary 0, no User Bank is available.

CONFIG+29 - Pointer to Start of Banked User Area. Word.
This word contains the address of the first available byte in the Banked User Area, if one exists. Routines loaded into the User Bank should contain a standard RSX header structure to link sequential programs and provide a primitive memory management function.

CONFIG+31 - CPU Clock Rate in Megahertz. Byte.
This byte must contain the processor speed rounded to the nearest Megahertz. It may be used by software timing loops in application and utility programs to adapt to the clock speed of the host computer and provide an approximate time.

This byte is reflected in the Environment Descriptor (see 5.2.4) as well for programs which are Z-System "aware".

CONFIG+32 - Additional Wait State Requirements. Byte.
This byte is "nibble-mapped" to reflect the number of wait states needed for memory and IO accesses when these functions can be set via software. In the Z80/Z180, IO port accesses have one wait state inserted within the processor. This byte does not account for this fact, and reflects wait states IN ADDITION TO any which are built into the hardware. For older processors such as the Z80, these bytes normally have no effect since additional wait states must be added with hardware.

CONFIG+33 - Timer Reload Value. Word.
In many systems, Interrupts or Timer values are set by software-configurable countdown timers. the 16-bit value at this location is reserved for setting the timer value and may be "fine tuned" to allow the system to maintain correct time in the presence of clock frequencies which may deviate from precise frequencies needed for accurate clocks.

CONFIG+35 - Floppy Disk Physical Parameters. Table.
This table consists of four 5-byte entries which contain information on up to four physical drives. Each entry is defined as:

Byte 0 - Provides base for XDPH byte. Bit mapped as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | └─┴─┴── Disk Size 000=Fixed Disk, 001=8", 010=5.25", 011=3.5"
    | | | | └──────── 0 = Single-Sided,       1 = Double-Sided
    | | | └──────────  <reserved>
    | | └──────────── 0 = Motor Always On     1 = Motor Control Needed
    | └────────────── 0 = 300 RPM Max Speed   1 = 360 RPM (8" & HD)
    └────────────────  <reserved>
```

Byte 1 - Step Rate in milliseconds.
Byte 2 - Head Load Time in milliseconds.
Byte 3 - Head Unload Time in milliseconds.
Byte 4 - Number of Tracks (Cylinders) on drive.

Those bits in Byte 0 which are listed as unused must be set to 0 since this byte provides the initial value stored in the XDPH when assignable drives are used. For controllers which do not need the available information (e.g. Western Digital controllers do not need Byte 3), these values may be set to any arbitrary value, but MUST remain present in the structure to prevent changing subsequent addresses.

CONFIG+55 - Motor On Time in 1/10th Seconds. Byte.
This time may be used in some types of Floppy Disk controllers to keep the drive motors spinning for a specified time after the last access to avoid delays in bringing the spindle up to speed. Some controllers, notably the Western Digital 17xx and 19xx series to not support this feature. In this case, the byte may be set to any arbitrary value, but MUST remain present.

CONFIG+56 - Motor Spinup Time in 1/10th Seconds. Byte.

This time is the delay which will be imposed by the Bios before attempting to access a Floppy Disk drive when it senses that the motor is in a stopped condition. Providing such a delay will minimize the probability of data corruption by writing to disk which is rotating at the incorrect speed.

CONFIG+57 - Maximum Number of Retries. Byte.

This byte specifies the number of attempts which will be made on a Floppy Disk access before returning an error code. In some cases, such as diagnostic programs, it may be desirable to set this value to 1 to identify soft errors, or ones which fail on the first attempt, but succeed on a subsequent try. We recommend a value of 3 or 4 based on our experience. Larger values may result in inordinately long delays when errors are detected.

CONFIG+58 - Pointer to Interrupt Vector Table. Word.

This Word contains the address of the base of an Interrupt Vector Table which, when used, contains pointers to service routines. The precise definition of the table is not standardized and may vary considerably between systems. This pointer serves only to provide an easy and standardized method of locating the table for re-definition of services or system features.

CONFIG+60 - SCSI Controller Type. Byte.

To accommodate the widest variety of different controllers including the older SASI models, this byte is defined as containing a byte code to the specific model being used. In most cases, this byte has little if any effect within the Bios, but may have significant effects on Hard Disk Diagnostic programs, or User-developed utilities. Any additions to this table should be coordinated with the authors to insure that the standard support utilities continue to function. Current definitions are:

0 - Owl
1 - Adaptec ACB-4000A
2 - Xebec 1410A/Shugart 1610-3 (SASI)
3 - Seagate SCSI
4 - Shugart 1610-4 (Minimal SCSI subset)
5 - SCSI-2 (Newer Conner, Quantum and Maxtor drives)

CONFIG+61 - Hard Drive Physical Parameters. **Table.**
This table consists of three 9-byte entries defining up to three physical Hard Drives. While the SCSI definition allows for more units, three was considered adequate for most systems. If additional drives are needed, please contact the authors for methods of including them without invalidating any of the standard utilities or interfaces. Each of the three entries is defined as:

Byte - Physical and Logical Address as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | └─┴─┴── Physical Device (000-110B, 111B reserved for Host)
    | | | | └──────── <reserved>
    | | | └────────── 0 = Drive NOT Present, 1 = Drive Present
    └─┴─┴──────────── Logical Unit Number (000-111B) for controllers
                        capable of handling multiple drives
```

Word - Number of Physical Cylinders on Drive
Byte - Number of Usable Physical Heads on Drive
Word - Cylinder Number to begin Reduced Write Current
Word - Cylinder Number to begin Write Precompensation
Byte - Step Rate. This byte may either be an absolute rate in mS or a code based on controller-specific definitions

For many of the newer controllers, the last three items may not have any meaning in which case they can be set to any arbitrary value. Also in newer drives, the physical characteristics such as the number of cylinders and heads may be hidden within the drive electronics with re-mapped values provided to the controller via various SCSI commands. As with the last three entries, in this case, they may be set to any arbitrary value.

CONFIG+88 (Reserved Bytes). Five Bytes are reserved for future expansion.

CONFIG+93 - Character Device Definitions. **Table.**
This table consists of four or more 16-byte (8-byte in B/P versions prior to 1.1) entries and must be terminated by a Null (binary Zero) byte. Each entry defines the name and characteristics of a character device in the system. The first four of these are directly available for selection by the IOBYTE as the Console, Auxiliary IO and Printer. Other entries may be defined and exchanged with the first four to make them accessible to the system. The entries are defined as:

String - Four Ascii character Name as: COM1, PIO1, NULL, etc.

Byte - Data Rate capabilities as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | └─┴─┴─┴── Current Data Rate Setting
    └─┴─┴─┴────────── Maximum Rate Available (Bits-per-Second) as:
```

0000 = None	0001 = 134.5	0010 = 50	0011 = 75
0100 = 150	0101 = 300	0110 = 600	0111 = 1200
1000 = 2400	1001 = 4800	1010 = 9600	1011 = 19200
1100 = 38400	1101 = 76800	1110 = 115200	1111 = Fixed

Byte - Configuration Byte defined as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | | | └── 0 = 2 Stop Bits,       1 = 1 Stop Bit
    | | | | | | └──── 0 = No Parity,         1 = Parity Enabled
    | | | | | └────── 0 = Odd Parity,        1 = Even Parity
    | | | | └──────── 0 = 8-bit Data,        1 = 7-bit Data
    | | | └────────── 0 = No XON/XOFF,       1 = XON/XOFF Control Enabled
    | | └──────────── 0 = No CTS/RTS,        1 = CTS/RTS Control Enabled
    | └────────────── 0 = Device NOT Input,  1 = Device can be read
    └──────────────── 0 = Device NOT Output, 1 = Can Write Device
```

Byte - Input Data Mask. Bit-mapped byte used to logically AND with bytes read from Device Input.

Byte - Output Data Mask. Bit-mapped byte used to logically AND with bytes before being output to device.
Word - Pointer to Character Output routine.
Word - Pointer to Output Status routine.
Word - Pointer to Character Input routine.
Word - Pointer to Input Status routine.

NOTE: The last four pointers are not at these locations in B/P Bios versions prior to 1.1, but were accessed by a pointer returned by Bios Function 30.

5.2.2 Disk Parameter Header.

The Disk Parameter Header (DPH) is a logical data structure required for each disk drive in a CP/M compatible Disk Operating System. It consists of a series of eight pointers which contain addresses of other items needed by the DOS as well as some scratchpad space. The Address of the DPH associated with a given drive is returned by the Bios after a successful selection with Bios Function 9. If Errors occur during selection, or the drive does not exist, a Null Pointer (0000H) is returned.

For B/P Bios, it was necessary to add an additional four bytes to each DPH which contain additional information on physical and logical parameters as well as flag information. These additional bytes are referred to as the Extended DPH, or XDPH. While similar in concept to the extension added to CP/M 3, the implementation is different. The XDPH prepends the DPH and may be accessed by decrementing the returned address. As a convention, DPHs in B/P Bios source code have reserved certain label sequences for specific types of units with DPH00-DPH49 used for Floppy Drives, DPH50-DPH89 for Hard Drive Partitions and DPH90-DPH99 for RAM Drives.

An entire DPH/XDPH block is required for each logical drive in a B/P Bios system. While some pointers, such as the pointer to the Directory Buffer, may be common across a number of drives, for most systems, the other items will point to unique areas.

The DPH/XDPH elements as indexed from the DPH addresses accessible to application programs are:

DPH-4 - Format Lock Flag. **Byte.**
A Zero value indicates that the format of the disk is not fixed, but may be changed. If the Bios was assembled with the Auto-select option, the Bios will scan a number of different formats in order to identify the disk. If a 0FFH value is placed in this byte, it indicates that the format is fixed and cannot be changed. This is normally the case for RAM and Hard disk drives, as well as for alien floppy formats which have been selected in the emulation mode. If the Auto-select option was not chosen during assembly of the Bios, all Floppy Disk drives will also have a 0FFH byte in this position showing that the formats cannot be changed.

DPH-3 - Disk Drive Type. **Byte.**
This byte is bit mapped and contains flags indicating many parameters of the drive. For Floppy Drives, this byte contains a copy of the first byte in the Physical Drive Table (See 5.2.1, CONFIG+35) with the two reserved bytes set during the drive selection process. The byte is then defined as:

```
        Drive Type Byte
  Bit 7 6 5 4 3 2 1 0
      | | | | | |_|_|____ Disk Size  000=Fixed Disk, 001=8", 010=5.25", 011=3.5"
      | | | | |__________ 0 = Single Sided              1 = Double Sided
      | | | |____________ 0 = Single Step Drive         1 = Double Step Drive
      | | |______________ 0 = Motor Always On           1 = Drive Motor Control Needed
      | |________________ 0 = Max Speed 5.25" (300 rpm) 1 = 8" & HD Max Speed (360 rpm)
      |__________________ 0 = Double Density            1 = Single Density
```

For Hard Disk Partitions and the RAM Drive, this byte is not used and is set to all Zeros indicating a Fixed Drive type.

DPH-2 - Driver ID Number. **Byte.**
Three Driver Types are used in the basic B/P Bios configuration. A Zero value indicates a Non-existent driver, with other values used to direct disk accesses to the respective code appropriate to the device. Basic defined driver types exist for Floppy Disk (1), Hard Disk via the SCSI interface (2), and RAM Disk (3). If you wish to extend this table to include tailored drivers, please consult with the authors to preclude possible conflicts with planned extensions.

DPH-1 - Physical Drive/Unit Number. **Byte.**
This byte contains the Physical Drive or Unit Number hosting the logical drive. For Floppy Drives, this will usually be in the range of 0 to 3 for four drives. Hard drives may have several DPHs sharing the same physical drive number, while this field is ignored in the single RAM drive supported in the distribution B/P Bios version.

NOTE: The Physical Drive Number byte for Hard Drives is comprised of two fields to ease handling of SCSI devices. Up to seven devices (device 111B is reserved for the Host Computer) each having up to 8 Logical Units may be defined. The Byte is configured as:

```
Physical Drive Number
Bit 7 6 5 4 3 2 1 0
    | | | | | |_|_|____ Physical Device (000-110B, 111B reserved for Host)
    | | | | |_________ <reserved>
    | | | |___________ 0 = Unit Not Available, 1 = Unit Active
    |_|_|_____________ Logical Unit Number (000-111B)
```

DPH+0/1 - Skew Table Pointer. Word.
This word contains a pointer to the Skew table indicator. It rarely is used for Hard and RAM drives, but is required in Floppy Disk drives. If the Bios was assembled using the Calculated Skew option, the address is of a Byte whose absolute value indicates the numerical value of skew (normally in the range of 1 to 6) used for disk accesses. This term is often replaced with Interleave, and is synonymous for this purpose. If the value of the byte is negative, it means that the sectors are recorded in a skewed form on the disk and that Reads and Writes should be sequential. If the value is positive, then an algorithm is called to compute a physical sector number based on the desired logical sector and the skew factor. For systems assembled without Calculated skew, this word points to a table of up to 26 bytes which must be indexed with the desired Physical Sector number (0..Maximum Sector Number) to obtain the Corresponding Disk Sector number.

DPH+2 - Dos Scratch Words. 3 Words.
These three words are available for the Dos to use as it requires. No fixed values are assigned, nor are meanings for the data stored there of any value.

DPH+8/9 - Directory Buffer Pointer. Word.
This word points to a 128-byte Data area that is used for Directory searches. It is usually a common area to all DPH's in a system and is frequently updated by the Dos in normal use.

DPH+10/11 - DPB Pointer. Word.
This word points to another data structure which details many of the logical parameters of the selected drive or partition. Its structure is detailed in Section 5.2.3 below. Drives of the same type and logical configuration may share DPB definitions, so it is not uncommon to find the DPB pointers in different DPH structures pointing to the same area.

DPH+12/13 - Disk Checksum Buffer. Word.
This word points to a scratch RAM buffer area for removable-media drives used to detect disk changes. Normally this feature is used only for Floppy Disk Drives, and is disabled by containing a Zero word (0000H) for Hard and RAM drives. For Floppy Drives, a RAM area with one byte for every four directory entries (128-byte sector) is needed (See 5.2.3, DPH+11/12). This scratch area cannot be shared among drives.

It should be noted that in a fully Banked B/P Bios system with ZSDOS2, the Checksum Buffer is placed in the System Bank and not directly accessible by applications programs.

DPH+14/15 - Allocation Vector (ALV) Buffer. Word.
This word points to a bit-mapped buffer containing one bit for each allocation block on the subject drive (See 5.2.3, DPB+5/6). A "1" bit in this buffer means that the corresponding block of data on the device is already allocated to a file, while a "0" means that the block is free. This buffer is unique to each logical drive and cannot be shared among drives.

It should be noted that in a fully Banked B/P Bios system with ZSDOS2, the ALV Buffer is placed in the System Bank and not directly accessible by applications programs. Since access to the ALV buffer is frequently needed to compute free space on drives, ZSDOS2 contains an added function to return disk free space. Using this call allows applications access to the information without directly accessing the data structure.

5.2.3 Disk Parameter Block.

The Disk Parameter Block (DPB) is a data structure defined by Digital Research for CP/M which defines the logical configuration of storage on mass storage. It has been expanded in B/P Bios to include additional information to provide enhanced flexibility and capability. The expansion is referred to as the Extended DPB or XDPB, and prepends the actual DPB structure. The address of the DPB may be obtained from the DPH pointer returned by the Bios or Dos after a disk selection (See 5.2.2 above). All DPBs reside in the Common Memory area and are available to applications programs whether in a Banked or Unbanked system. For the sake of a convention, the DPBs are labeled in the same manner as DPHs with DPB00-DPB49 used for Floppy Drives, DPB50-DPB89 for Hard Drive Partitions, and DPB90-99 for RAM Drives.

The layout of the Disk Parameter Block as indexed from the available DPB pointer is:

DPB-16 - Ascii ID String. 10 Bytes.
This string serves as an identification which may be printed by applications programs such as our BPFORMAT. This string may be a mixed alphanumeric Ascii set of up to ten characters, but the last valid character must have the Most Significant Bit (Bit 7) Set to a "1".

DPB-6 - Format Type Byte 0. Byte.
This byte contains some of the information about the format of the drive, and the logical sequencing of information on the physical medium. The bits in the byte have the following significance:

```
Bit  7 6 5 4 3 2 1 0
     | | | | | |_|_|____ Disk Size:  000 = Fixed Disk, 001 = 8", 010 = 5.25", 011 = 3.5"
     | | |_|_|__________ Track Type
     | |                    000 = Single Side        001 = Reserved
     | |                    010 = Sel by Sec, Cont   011 = Sel by Sec, Sec # Same
     | |                    100 = S0 All, S1 All     101 = S0 All,S1 All Reverse
     | |                    110 = Sel by Trk LSB     111 = Reserved
     | |________________ 0 = Track 0 Side 0 is Double Density, 1 = Single Density
     |__________________ 0 = Data Tracks are Double Density, 1 = Single Density
```

For Hard Drives and RAM Drives, this byte contains all Zero bits to signify Fixed Media and format.

DPB-5 - Format Type Byte 1. Byte.
This byte contains additional information about the format of information. The bits have the following meanings:

```
Bit  7 6 5 4 3 2 1 0
     | | | | | |_|_|____ Sector Size:  000 = 128, 001 = 256, 010 = 512, 011 = 1024
     | | |_|_|__________ Allocation Size:  000=1K, 001=2K, 010=4K, 011=8K, 100=16K
     | |                    (NOTE: This should match the definition in DPH)
     | |________________ <Reserved>
     |__________________ 0 = Normal Speed (300 rpm)
                         1 = 8" & HD Floppy (360 rpm) or Hard Drive
```

For Hard Drives. The distribution version of B/P Bios and the support utilities assume that the Sector size is always 512 bytes. The remaining bits should be set as indicated.

DPB-4 - Skew Factor. Byte.
This byte is a signed binary value indicating the skew factor to be used during Format, Read and Write. It is normally used only with Floppy Drives and usually set to -1 (0FFH) for Hard and RAM drives to indicate that Reads and Writes should be done with No skew. If the option to calculate skew is in effect during Bios assembly, the Skew pointer in the DPH (DPH+0/1) points to this byte. If a skew table is used, this byte has no effect and should be set to 80H.

DPB-3 - Starting Sector Number. Byte.
This byte contains the number of the first Physical Sector on each track. Since most Disk Operating Systems use a Zero-based sequential scheme to reference sectors, this value provides the initial offset to correct logical to physical sector numbers.

DPB-2 - Physical Sectors per Track. Byte.
This byte contains the number of Physical (as opposed to logical) Sectors on each track. For example, CP/M computes sectors based on 128-byte allocations which are used on single-density 8" Floppy Disks. One of the popular five-inch formats uses five 1k physical sectors which equates to 40 logical CP/M sectors. This byte contains 5 in this instance for the number of 1k Physical Sectors.

DPB-1 - Physical Tracks per Side. Byte.
This byte contains the number of Physical Tracks per Side, also called the Number of Cylinders. It reflects the Disk, as opposed to the Drive capabilities and is used to establish the requirements for double-stepping of Floppy Drives. In the case of a 40-track disk placed in an 80-track drive, this byte would contain 40, while the Drive parameter in the Configuration Section contains 80 as the number of tracks on the drive. This byte has no meaning for Hard Drive partitions or RAM drives and should be set to Zero, although any arbitrary value is acceptable.

DPB+0/1 - Logical Sectors per Track. **Word.**
This value is the number of Logical 128-byte sectors on each data track of the disk. It is equivalent to the number of Physical Sectors times the Physical Sector Size MOD 128.

DPB+2 - Block Shift Factor. **Byte.**
DPB+3 - Block Mask. **Byte.**
DPB+4 - Extent Mask. **Byte.**
These three bytes contain values used by the Operating System to compute Tracks and Sectors for accessing logical drives. Their values are detailed in various references on CP/M and ZSDOS programming and should not be varied without knowledge of their effects.

DPB+5/6 - Disk Size (Capacity). **Word.**
This Word contains the number of the last allocation block on the drive. It is the same as the capacity in allocation blocks - 1. For example, if 4k allocation blocks are being used and a 10 Megabyte drive is being defined, this word would contain 10,000,000/4000 - 1 or 2499.

DPB+7/8 - Maximum Directory Entry. **Word.**
This Word contains the number of the last Directory Entry and is the same as the Number of Entries - 1. For example, if 1024 directories are desired, this word would be set to 1024 - 1 = 1023.

DPB+9/10 - Allocations 0 and 1. **2 Bytes.**
These two Bytes hold the initial allocations stored in the first two bytes of the ALV Buffer (See 5.2.2, DPH+14/15) during initial drive selection. Their primary use is to indicate that the Directory Sectors are already allocated and unavailable for data storage. They are bit-mapped values and are used in Hi-byte, Lo-byte form as opposed to the normally used Lo-byte, Hi-byte storage used in Z80 type CPUs for Word storage. The bits are allocated from the MSB of the first byte thru the LSB, then MSB thru LSB of the second byte based on one bit per allocation block or fraction thereof used by the Directory. The bits may be calculated by first computing the number of entries per allocation block, then dividing the desired number of entries by this number. Any remainder requires an additional allocation bit.

For example, if 4k allocation blocks are used, each block is capable of 4096/32 bytes per entry = 128 Directory Entries. If 512 entries are desired, then 512/128 = 4 allocation blocks are needed which dictates that Allocation byte 0 would be 11110000B (0F0H) and Allocation Byte 1 would be 00000000B.

DPB+11/12 - Check Size. **Word.**
This Word is only used in removable media (normally only Floppy Drives) and indicates the number of sectors on which to compute checksums to detect changed disks. It should be set to 0000H for Fixed and RAM Disks to avoid the time penalty of relogging after each warm boot.

DPB+13/14 - Track Offset. **Word.**
This Word indicates the number of Logical Tracks to skip before the present DPB is effective. It is normally used to reserve boot tracks (usually 1 to

3), or to partition larger drives into smaller logical units by skipping tracks used for other drive definitions.

5.3.1 Environment Descriptor.

The Environment Descriptor, referred to as simply the ENV, is the heart of what is now known as *The Z-System*. The most recent additions to the system by Joe Wright and Jay Sage replaced some relatively meaningless elements in the ENV with system dependent information such as the location of the Operating System components. Consequently, the ENV is not just a feature of the ZCPR 3.4 Command Processor Replacement, but is an Operating System Resource which allows other programs such as ZCPR 3.4 to access its information.

The B/P Bios requires an ENV to be present, and uses several items of information contained in it. The banked ZSDOS2 and Z40 Command Processor Replacement use even more ENV features. A few remaining bytes have been re-defined for support to B/P Bios-based systems. To denote the definition of B/P Bios data elements, a new Type, 90H, has been reserved. Using this "Type" byte, user programs can access and take advantage of the new definitions and features.

A template for the Environment Descriptor used in B/P Bios which takes its values from the Z3BASE.LIB file included in the distribution disk is:

```
************************** C A U T I O N ******************************
* If transitioning from an older operating environment which          *
* uses user-loaded Environment files such as SYS.ENV, you must        *
* either delete such load instructions or modify the ENV files        *
* to reflect the EXACT loaded system definition.                      *
* Failure to observe this fact will probably cause many               *
* undesired side effects!                                             *
***********************************************************************
```

```
; Environment Descriptor for ZCPR34

ENV:    JP      0               ; Leading jump (address is CBIOS when NZCOM)
        DEFB    'Z3ENV'         ; Environment ID
        DEFB    90H             ; Env type (>=90H means B/P Extended ENV w/User Area)
        DEFW    EXPATH          ; External path (PATH)
        DEFB    EXPATHS         ;
        DEFW    RCP             ; Resident command package (RCP)
        DEFB    RCPS            ;
        DEFW    IOP             ; Input/output package (IOP)
        DEFB    IOPS            ;
        DEFW    FCP             ; Flow command package (FCP)
        DEFB    FCPS            ;
        DEFW    Z3NDIR          ; Named directories (NDR)
        DEFB    Z3NDIRS         ;
        DEFW    Z3CL            ; Command line (CL)
        DEFB    Z3CLS           ;
        DEFW    Z3ENV           ; Environment (ENV) - Actual Starting Address
        DEFB    Z3ENVS          ;
        DEFW    SHSTK           ; Shell stack (SH)
        DEFB    SHSTKS          ;
        DEFB    SHSIZE          ;
        DEFW    Z3MSG           ; Message buffer (MSG)
        DEFW    EXTFCB          ; External fcb (FCB)
        DEFW    EXTSTK          ; External stack (STK)
        DEFB    0               ; Quiet flag (1=quiet, 0=not quiet)
        DEFW    Z3WHL           ; Wheel byte (WHL)
        DEFB    9               ; Processor speed (MHZ)
        DEFB    'P'-'@'         ; Max disk letter
        DEFB    31              ; Max user number
        DEFB    1               ; 1 = Ok to accept DU:, 0 = Not Ok
        DEFB    0               ; CRT selection                     ()
        DEFB    0               ; Printer selection                 ()
        DEFB    80              ; CRT 0: width
        DEFB    24              ; # of lines
        DEFB    22              ; # of text lines

; In Extended ENV, CRT 1 is replaced by System Info

;;      DEFB    132             ; . CRT 1: Width
;;      DEFB    24              ;         # of lines
;;      DEFB    22              ;         # of text lines

;               PONMLKJIHGFEDCBA
                 IF   RAMDSK
        DEFW    0001000001111111B ; Ram is Drive M:
                 ELSE
        DEFW    0000000001111111B ; Valid drives vector
                 ENDIF
        DEFB    0               ;                           (Reserved)
        DEFB    80              ; PRT 0: width
        DEFB    66              ; # of lines
        DEFB    58              ; # of text lines
        DEFB    1               ; FF flag (1=can Form Feed)

;========= Usurped Prt1 storage for Resident User Space Vectors =========
;;      DEFB    96              ; PRT 1: width
;;      DEFB    66              ; # of lines
;;      DEFB    58              ; # of text lines
;;      DEFB    1               ; FF flag (1=can Form Feed)

        DEFB    USPCS           ; Remaining Free User Space (recs)
USRSP:  DEFW    USPC            ; Res. User Space base Address (xx00H/xx80H)
        DEFB    USPCS           ; Size of Res. User Space in 128-byte recs

;=======================================================================
; In Extended ENV, Printers 2 and 3 are gone, replaced by System Info

;;      DEFB    132             ; . PRT 2: Width
;;      DEFB    66              ;         # of lines
;;      DEFB    58              ;         # of text lines
;;      DEFB    1                 ;         FF flag (1=can form feed)
;;      DEFB    132               ; . PRT 3: Width
;;      DEFB    88                ;         # of lines
;;      DEFB    82                ;         # of text lines
;;      DEFB    1                 ;         FF flag (1=can form feed)
        DEFW    CPR               ; CCP base address
        DEFB    [DOS-CPR]/128     ; Size of CCP in 128 byte records
        DEFW    DOS               ; Bdos base address (xx00H or xx80H)
        DEFB    [BIOSJT-DOS]/128  ; Bdos buffer size in 128 byte records
        DEFW    BIOSJT            ; Bios base address (NZBIO if NZCOM running)
        DEFB    'SH      '        ; Shell variable filename
        DEFB    'VAR'             ; Shell variable filetype
        DEFB    '        '        ; File 1
        DEFB    '   '             ;
        DEFB    '        '        ; File 2
        DEFB    '   '             ;
        DEFB    '        '        ; File 3
        DEFB    '   '             ;
        DEFB    '        '        ; File 4
        DEFB    '   '             ;
        DEFB    0                 ; Public drive area (ZRDOS +)
        DEFB    0                 ; Public user area (ZRDOS +)
                                  ; Env 128 bytes long
```

5.3.2 Terminal Capabilities.

In addition to the Basic Environment Descriptor described above, a dummy Terminal Capability record structure (TCAP, also known as TERMCAP) is attached to reserve space for, and define default capabilities of the computer terminal. The default TERMCAP is fully compliant with the VLIB routines used in the Z-System Community after VLIB4D. The skeleton record structure is:

```
;*************************************************************************
; This TermCap Data for the New Z-System complies with VLIB4D specs and more
; fully describes the terminal and its capabilities.  Edit the fields with
; values for your terminal characteristics, or use it as a template for an
; outboard definition loaded from the Startup file.

TCAP:   DEFB    '               '  ; Terminal Name (13 bytes, space terminated)

B13:    DEFB    GOELD-ENV2         ; Offset to GOELD in graphics section
B14:    DEFB    10000000B          ; Bit 7 = Extended TCAP, remainder undefined

; B15 b0   Standout                0 = Half-Intensity,  1 = Reverse Video
; B15 b1   Power Up Delay          0 = None,            1 = 10-second delay
; B15 b2   No Wrap                 0 = Line Wrap,       1 = No Wrap if char written
;                                                         to last character in line
; B15 b3   No Scroll               0 = Scroll,          1 = No Scroll if char written to
;                                                         last char in last line of diplay
; B15 b4   ANSI                    0 = ASCII,           1 = ANSI
```

```
B15:    DEFB    00000000B          ; Reverse Vid, Wrap, Scroll, ASCII
                          ; Additional single character cursor motion bytes
        DEFB    'E'-'@'            ; Cursor Up
        DEFB    'X'-'@'            ; Cursor Down
        DEFB    'D'-'@'            ; Cursor Right
        DEFB    'S'-'@'            ; Cursor Left

        DEFB    0                  ; CL Delay for Screen Clear
        DEFB    0                  ; CM Delay for Cursor Motion
        DEFB    0                  ; CE Delay for Clear to End-of-Line
                          ; Strings start here
        DEFB    0                  ; (CL) Home Cursor and Clear Screen
        DEFB    0                  ; (CM) Cursor Motion
        DEFB    0                  ; (CE) Clear to End-of-Line
        DEFB    0                  ; (SO) Reverse On
        DEFB    0                  ; (SE) Reverse Off
        DEFB    0                  ; (TO) Terminal Init
        DEFB    0                  ; (TE) Terminal De-init
                          ; Extensions to Standard Z3TCAP
        DEFB    0                  ; (LD) Delete Line
        DEFB    0                  ; (LI) Insert Line
        DEFB    0                  ; (CD) Clear from Cursor to End-of-Scr
                          ; Attributes setting parameters
        DEFB    0                  ; Set Attributes
        DEFB    0                  , Attributes String
                          ; Read items from screen
        DEFB    0                  ; Report Cursor Pos'n (ESC Y Pn Pn)
        DEFB    0                  ; Read Line Under Cursor

GOELD:  DEFB    0                  ; On/Off Delay
                          ; Graphics strings offset from Delay value.
        DEFB    0                  ; Graphics On
        DEFB    0                  ; Graphics Off
        DEFB    0                  ; Cursor Off
        DEFB    0                  ; Cursor On
                          ; Graphics Characters
        DEFB    '*'                ; Upper-Left corner                [*]
        DEFB    '*'                ; Upper-right corner               [*]
        DEFB    '*'                ; Lower-Left corner                [*]
        DEFB    '*'                ; Lower-right corner               [*]
        DEFB    '-'                ; Horizontal Line                  [-]
        DEFB    '¦'                ; Vertical Line                    [¦]
        DEFB    '*'                ; Full Block (hashed block)        [*]
        DEFB    '#'                ; Hashed Block (big X)             [#]
        DEFB    '+'                ; Upper Intersect (Upside down "T")  [+]
        DEFB    '+'                ; Lower Intersect ("T")            [+]
        DEFB    '+'                ; Mid Intersect (Crossing Lines)   [+]
        DEFB    '+'                ; Right Intersect ("T" rotated left)  [+]
        DEFB    '+'                ; Left Intersect ("T" rotated right)  [+]
        DEFB    0
        DEFB    0
ENVEND:
```

6. B/P Bios Utilities.

We have developed and adapted many support routines to assist you in using and tailoring B/P Bios installations. Many of these follow generally established functions. and their use may be readily perceived. Others are adaptations of routines which we developed to support our earlier ZSDOS operating system. or are adapted from routines available in the Public Domain. The remainder were written specifically to support this product.

Each support routine follows generally-acknowledged practices of Z-System Utilities. including built-in Help. and re-execution with the **GO** command. Help is accessed by typing the name of the desired routine followed by a double-slash (i.e. **LDSYS //**). Additional information is provided in the following sections of this manual.

Some of the features of B/P Bios are supported with Version-dependent programs. These vary among specific systems and should not be intermingled if you support a number of different computer types. Contact us if you need to create such a specific tailored version of one of these programs. In the following descriptions, the routines which are restricted to a specific system or which require tailoring are so annotated.

6.1 BPBUILD - System Image Building Utility.

The purpose of BPBUILD is to create a loadable System Image. Input files needed by this utility include the output of assembling the BPBIO-xx file (in MicroSoft REL format), a REL or ZRL image of an Operating System (ZSDOS for unbanked or ZSDOS2.ZRL for fully banked systems are recommended), and a REL or ZRL image of a Command Processor (ZCPR33.REL for unbanked, Z40.ZRL for fully banked systems are recommended).

BPBUILD is capable of incorporating many types of operating system components into the executable image. Among systems tested are customized BIOSes, ZRDOS, CP/M 2.2, ZCPR2, ZCPR3x and others. One restriction in any segment occurs in the use of Named COMMONs for locating various portions of the system within the processor's memory map. Most assemblers and linkers are limited in the number of different named COMMON bases which can be supported. The linker incorporated in BPBUILD can handle only the following named COMMONs: _ENV_, _MCL_, _MSG_, _FCB_, _SSTK_, _XSTK_, _BIOS_, BANK2, B2RAM, and RESVD.

6.1.1 Using BPBUILD.

BPBUILD operates with layered Menus and may be invoked to build a replacement for an existing Image file, or with defaults to construct a totally new Image file. All files needed to construct the desired image: Bios, Dos, and CPR must reside in the current Drive and User Area, or be accessible via the PUBlic attribute if operating under ZSDOS. The syntax for BPBUILD is:

```
BPBUILD           - Generate system w/defaults
BPBUILD fn[.ft]   - Make/Modify a specific system
                      (Default type is .IMG)
BPBUILD //        - Display this Message
```

It should be noted that if ZSDOS2 is the selected Dos, Bit Allocation buffers for Hard Drives are located in the System Bank.

6.1.2 Menu Screen Details.

Upon starting BPBUILD, you will be presented with the main menu screen. From this screen, you may select one of the three main categories to tailor the output image. At any point in the configuration, pressing Control-C or ESCape will exit the menu and return to the Command Processor prompt.

```
Main


        1  File Names

        2  BIOS Configuration

        3  Environment

               Selection :
```

6.1.2.1 Screen 1 - File Names.

Selecting option One from the main menu will present a screen listing the current file names for the three input files needed to build an image, and the name to be applied to the output file. If BPBUILD was executed with the name of an existing Image file, the file names will be those of the files used to build the specified Image file, otherwise it will be default names furnished by BPBUILD. A sample screen for a non-banked YASBEC system might be:

```
 Files (1.1)


        1  Command Processor File : ZCPR33   .REL

        2  Operating System File  : ZSDOS    .ZRL

        3  B/P Bios Source File   : B/P-YS   .REL

        4  B/P Executable Image   : BPSYS    .IMG

               Selection :
```

Selecting any one of the four options will allow you to change the name of the file to use for the desired segment. Default File types exist for each entry with both the Command Processor and Operating System files defaulting to .ZRL if no type is entered. The Bios file defaults to .REL, while the Image output file defaults to .IMG.

6.1.2.2 Screen 2 - BIOS Configuration.

Selecting BIOS Configuration from the Main Menu (Selection 2) presents the basic screen from which the sizes and locations of either existing system segments (if building from an existing IMG file) or default values from within BPBUILD. A sample screen might appear as:

```
 Environment (2.1)

 COMMON (Bank 0) MEMORY               BANK 2 MEMORY
 ----------------------           -----------------------
 A   Common BIOS  - D400H          E   Banked BIOS  - 0000H
       Size       -   83                 Size       -    0
 B   Common BDOS  - C600H          F   Banked BDOS  - 0000H
       Size       -   28                 Size       -    0
 C   Command Proc - BE00H          G   Command Proc - 0000H
       Size       -   16                 Size       -    0
 D   User Space   - E900H          H   User Space   - 0000H
       Size       -    6                 Size       -    0


               Selection :
```

While the ability to dictate locations and sizes for various system sizes is provided at this point, we urge you not to alter the values for Bank 2 Memory unless you are *VERY* familiar with the potential effects and willing to risk potentially disastrous consequences. The primary reason for including this screen was to allow setting Common Memory base locations and to dictate the size of the Resident User Space. Other than specifying the Common User Space size (the starting location defaults to the address in Z3BASE.LIB), the remaining values were included primarily for the few specialized users who require custom system locations.

6.1.2.3 Screen 3 - Environment Configuration.

Since the Environment Descriptor is an integral part of the Operating System due to the specification of low-level parameters such as memory allocations, this screen is provided to configure the memory map in a suitable fashion for the system being built. For example, in a Fully Banked system with ZSDOS2, there is normally no need for a Resident Command Processor. Using this screen then, selection D may be used to indicate that no space is used for the RCP by setting its location and size to zero. With this space freed, the IO Package (Selection C) may be raised to F400H, keeping its size at 12 records. Since these are the two lowest segments in the memory map, BPBUILD will use this as the lowest value used in the Environment, and move the Operating System segments (including the Resident User Space) up in memory for an increase of 2k bytes in Transient Program Area.

It is important to note that the Environment Descriptor defined in this screen is stored in a special area within the .IMG file produced and placed in memory by LDSYS when activated. Any alteration after loading, for example loading another ENV file as part of the STARTUP script may cause the system to operate incorrectly.

```
 Environment (3.1)

  A - Environment    - FE00H       F - Named Dirs       -  FC00H
      Size (# recs)-    2              # of Entries -     14
  B - Flow Ctrl Pkg - FA00H       G - External Path -  FDF4H
      Size (# recs)-    4              # of Entries -      5
  C - I/O Package    - EC00H       H - Shell Stack     -  FD00H
      Size (# recs)-   12              # of Entries -      4
  D - Res Cmd Proc   - F200H           Entry Size      -     32
      Size (# recs)-   16         I - Msg Buffer      -  FD80H
  E - Command Line   - FF00H       J - Ext. FCB        -  FDD0H
      Size (bytes) -  208         K - Ext. Stack      -  FFD0H

                  Selection :
```

When all configuration or inspection activity is complete at any menu, entering a return with no selection will return to the previous menu screen (Main menu from lower screens), and will start the build activity if a single return is entered from the main menu. All specified files are first read to determine their sizes, and internal memory address calculations are performed. You will be asked if you desire BPBUILD to use optimal addresses for the maximum amount of Transient Program Area (AutoSize), or use the values which were specified in menu 2.1. If you enter N at this point, the build will progress under the assumption that you want to use the values from Menu 2.1. A second prompt will ask you if you want to build a system of "standard" segment sizes. This refers to the CP/M 2.2 standard sizes of 16 records (2k) for the Command Processor and 28 records (3.5k) for the Basic Disk Operating System (BDOS). If you answer Yes, AND the segments are equal to or less than these sizes, the system will be built to reflect these system segment sizes. Since many ill-behaved, but very popular, programs assume the old CP/M segment sizes, this option should generally be used. If the autosize query is selected, BPBUILD automatically executes to completion and returns to the Command Processor prompt.

To obtain the maximum Transient Program Area with "standard" segment sizes, the easiest method is to execute BPBUILD exiting with the Autosize query answered in the affirmative (Yes), then execute BPBUILD again on the produced image answering the first query with N for No Autosizing, and the second query with a Y to adjust the lower segment sizes for "standard" segment locations. While this is a somewhat cumbersome procedure, it results in a much smaller and faster running utility than otherwise possible, and was a design tradeoff in the development process.

6.2 BPCNFG - Configuration Utility

The flexibility of the B/P Bios architecture permits customizing to your system and desired methods of operation. This utility consolidates some of the more common and important tailoring features in a single utility. BPCNFG, the *B/P Bios Configuration Utility*, provides an easy, menu-driven means of tailoring Hard and Floppy Drive Boot Sector images, Relocatable Image (IMG) files, and certain elements in an executing system. Using BPCNFG reduces the need to assemble a new Bios image for simple changes, and increases the speed with which changes can be made.

6.2.1 Using BPCNFG.

BPCNFG syntax follows the standard conventions summarized in Section 1.2, and responds to the standard help option sequence of two slash characters. The syntax under which BPCNFG is invoked is dictated by the type of system you wish to configure. The BPCNFG Syntax is:

```
BPCNFG //                      <-- Print Built-in Help Summary
BPCNFG                         <-- Run interactive, screen mode
BPCNFG *                       <-- Configure Executing System
BPCNFG d[:]                    <-- Configure Drive "d"
BPCNFG [du:]filename[.typ]     <-- Configure Image File
```

The First form of the syntax is self-explanatory and simply prints a short help file listing the purpose of the utility and the syntax of the program use. The Interactive mode of operation, execution of BPCNFG with no arguments, will first ask you whether you wish to configure a Memory, Disk or Image version of a B/P Bios system, and set internal parameters to that mode, as well as loading the requisite data if needed. If no file type is specified for an Image file, a type of .IMG is assumed.

BPCNFG may configure options in the currently operating system with some limitations. The most significant limitation is that Hard Drive partitions may not be altered since space for the required bit buffers (ALV and CSV) is allocated at system load (boot) time and cannot be reset when a system is already installed. With this exception, all other parameters may be varied and will be in effect until another system is loaded by Cold Booting the computer, or loading an Image file with LDSYS.

To abort the BPCNFG without changing current parameters, press either ESCape or Control-C from the main menu display. Option setting will occur if a Carriage Return only is entered from the Main Menu Screen.

6.2.2 Menu Screen Details.

BPCNFG is a Screen-oriented support routine using the terminal attributes defined in the currently-installed Termcap. Five main screens are currently defined at the main menu, with several selections resulting in sub-menu screens. The first is the main menu from which the general category is selected for alteration. The top screen line reflects the level and mode as:

```
Main Menu  -  Configuring Image File  [A10:T1.IMG       ]
              Bios Version 2.0

  1   System Options

  2   Character IO Options

  3   Floppy Subsystem Options

  4   Hard Disk Subsystem Options

  5   Logical Drive Layouts


       Enter Selection :
```

Each of the five selections results in a sub-menu screen which depicts the specific elements which may be changed and the current values/settings. Each of the selection screens is detailed below.

6.2.2.1 Screen 1 - System Option Parameters.

The System Options may be set from Menu Screen 1 which may appear as:

```
Menu 1 - System Options


  1   System Drive     = A:

  2   Startup Command = "START01 "

  3   Reload Constant = 23040 (5A00H)

  4   Processor Speed = 9 MHz

  5   Memory Waits = 0,  IO Waits = 0


       Enter Selection :
```

The Logical System drive selected by the first entry is used by the Bios Warm boot to load the operating system on a Cold boot, and on Error Exits. The Startup command is normally the name of a COM file which performs additional system initialization when first started. Such a file is normally created by entering a sequence of instructions (file names and arguments) into a script using ALIAS, SALIAS, VALIAS or other similar ZCPR3 tool.

The remainder of the entries concern the Physical hardware within the computer and generally do not need to be changed for other than initial installation. The Reload Constant at Selection 3 may be "fine tuned" by slightly varying the value. The normal effect of this is to adjust the speed of the Real Time clock within the computer to allow it to keep proper time. The Processor Speed (Selection 4) should be the CPU Clock speed rounded to the nearest MegaHertz. For example, the rate shown in the sample screen is, in fact, 9.216 MHz. When the Clock Speed is changed, an option is offered to automatically scale the Reload Constant (Selection 3) to scale it by a proper factor based on the amount that the Clock Frequency was changed. This is not always needed, but is generally desirable in HD64180/Z-180 systems using an interrupt clock timer.

The number of Memory and IO Waits at Selection 5 are in addition to any Wait states inserted automatically in the hardware. For example, the Zilog Z180 inserts one wait state into all IO Port addresses. Since BPCNFG is a general purpose utility, it cannot know this. The number set with this selection is therefore *in addition to* any hardware injected wait states.

A sixth menu also exists, but since inadvertent alteration of its values can result in bizarre and potentially destructive consequences, its existence is hidden and prompted if selected. The interaction to select option 6 is:

```
Enter Selection : 6
-- This is DANGEROUS...Proceed? (Y/[N]) :
```

If an explicit Y is entered, the hidden menu is displayed appearing as follows for a YASBEC with the MEM4 addressing PAL which allows the full 1 Megabyte range to be used:

```
Menu 1.1 - System Bank Numbers


  1   TPA Bank #        = 1

  2   System Bank #     = 3

  3   User Bank #       = 4

  4   RAM Drive Bank #  = 5

  5   Maximum Bank #    = 31


      Enter Selection :
```

The bank numbers should reflect the *Physical* 32k bank numbers in the system progressing from the lowest, or first available, to the highest. The Bank numbers are Zero-based, and may range from 0 to 255 (0FFH). The User Bank and RAM Drive Banks may be set to Zero if No Banked User area or RAM Drive respectively is desired. It should be noted that the actual control for these two features is in other screens, and setting the Bank Numbers to Zero while the options are active can have adverse consequences.

6.2.2.2 Screen 2 - Character IO Parameters.

Screen Menu 2 allows configuration of the Character IO subsystem, to include assignment of logical devices to physical devices. A sample screen for the YASBEC might appear as:

```
Menu 2 - Character IO Options


  1    IOBYTE Assignment:
               Console   = COM2
               Auxiliary = COM1
               Printer   = PIO

  2    COM1 - 9600 bps, 8 Data, 1 Stop, No Parity, [In(8)/Out(8)]

  3    COM2 - 19.2 kbps, 8 Data, 1 Stop, No Parity, [In(7)/Out(8)]

  4    PIO1 - [Out(7)]

  5    NULL - [In(8)/Out(8)]

  6    COM3 - 38.4 kbps, 8 Data, 1 Stop, No Parity, [In(8)/Out(8)]
  7    Swap Devices


        Enter Selection :
```

Selection 1 determines which of the active devices is assigned to the address-able functions dictated by the IOBYTE. This location in memory is used by the operating system for Character IO normally consisting of the Console, Printer (List Device in CP/M terminology) and an Auxiliary IO (Reader and Punch in CP/M). The exact Devices assigned to these Logical functions depends on the bit-mapped characteristics of the IOBYTE, and positioning within the table portrayed in this Menu.

Early versions of B/P Bios (Versions 1.0 and earlier) distributed as "Starter Systems" and test versions used a fixed number of devices and a different data structure for device access. B/P Bios Production systems with Version numbers of 1.1 through 1.9 will contain only four device drivers, but use the newer data structures. Beginning with Version 2.0, however, many Character Devices may be accommodated, with the first four being directly accessible by the IOBYTE, and the ability to exchange devices to place any four in the first positions for access by applications programs. The above screen depicts a typical display under a Version 2.0 Bios. Smaller systems such as placed on the Boot Tracks will use only four fixed devices with a typical Version Number of 1.1 and will not include selections 6 or 7.

Selection 1 offers the opportunity to select which of the first four physical devices is assigned to the three logical IOBYTE defined functions. A secondary menu will prompt you for the exact functions and devices. An example of the interaction with selection 1 is:

```
        Set [C]onsole, [A]uxiliary, or [P]rinter : C
          Set Com[1], Com[2], [P]io, or [N]ull ?
```

Selections 2, 3 and 6 in this sample screen allow configuration of the serial ports included in this B/P Bios. If you are making hardware changes in the Bios, please follow the specifications in the source code to insure that this utility will function as described. For most system installations covered to date, one or more of these options will have no effect, but the capability still exists to configure the option. This selection results in a series of options being presented for verification or alteration. Current settings are listed as default values which will remain selected if no entry is made before the ending Carriage Return (Enter key). A Sample interaction on a YASBEC is:

```
        Configuring COM1 [In(8)/Out(8)]:
            Baud Rate  (Max=38.4 kbps) =   Selections are:
                1-134.5 bps     2-50 bps         3-75 bps        4-150 bps
                5-300 bps       6-600 bps        7-1200 bps      8-2400 bps
                9-4800 bps      10-9600 bps      11-19.2 kbps    12-38.4 kbps
            Select      [10]    :
            Data = 8-bits.      Change? (Y/[N]) :
            Stop Bits = 1-bits. Change? (Y/[N]) :
            Parity = None       Change? (Y/[N]) :
            XON/XOFF Flow = No  Change? (Y/[N]) :       * Defaults selected
            RTS/CTS Flow = No   Change? (Y/[N]) :
            Input is 8-bits.    Change? (Y/[N]) :
            Output is 8-bits.   Change? (Y/[N]) :
```

Selection four tailors the default parallel port, normally a Centronics printer interface. As with previous selections, not all options may be active, but will appear in the configuration sequence to retain the generality of BPCNFG. Current settings (in braces) will be retained if a Carriage Return is entered.

```
        Configuring PIO1 [Out(7)]:
            XON/XOFF Flow = No  Change? (Y/[N]) :
            RTS/CTS Flow = No   Change? (Y/[N]) :
            Output is 7-bits.   Change? (Y/[N]) :
```

The final selection (only appearing in Bios Versions 2.0 and later) allows the exchange of devices. In the sample Menu 2, Selections 2-5 are active devices, with Selection 6 existing in the Bios, but not accessible. Using the last selection, COM3 could be exchanged with one of the first four making it an active device upon exiting BPCNFG which calls the Device Configuration function. The interaction may appear as:

```
        Exchange Device : 3    With Device : 6
```

After this choice, active devices will be COM1, COM3, PIO1 and NULL.

6.2.2.3 Screen 3 - Floppy Disk Parameters.

Extensive tailoring of the Floppy Disk subsystem is possible at this Screen. While many popular systems do not support all options (mainly due to the Controller logic), all options and respective settings will appear to be set at this point. Also, while such items as the Step rate will accept increments of a set size (1 mS for Step Rate), many systems have discrete increments that do not match allowable values. For example, the SMS 9266 used in MicroMint's SB-180 will only accept 2 mS increments with 5 1/4" Floppy Drives, the WD 1770 used in the Ampro Little Board only steps at 6, 12, 20 and 30 mS, while the 1772 used in the YASBEC permits 2, 3, 5, and 6 mS Steps. The rates set with BPCNFG are suitably rounded up to the most appropriate rate within the Bios code during drive selection to allow tailoring of drives in a generalized manner. Options tailorable for Floppy Disk drives as seen on a YASBEC are:

```
Menu 3 - Floppy Disk Options


  1   Floppy Drive Characteristics:
       Drv0 = 3.5" DS, 80 Trks/Side
               Step Rate = 3 mS, Head Load = 4 mS, Unload = 240 mS
       Drv1 = 5.25" DS, 40 Trks/Side
               Step Rate = 4 mS, Head Load = 24 mS, Unload = 240 mS
       Drv2 = 5.25" DS, 80 Trks/Side
               Step Rate = 3 mS, Head Load = 24 mS, Unload = 240 mS
       Drv3 = 5.25" DS, 80 Trks/Side
               Step Rate = 6 mS, Head Load = 24 mS, Unload = 240 mS

  2   Motor ON Time (Tenths-of-Seconds) : 100

  3   Motor Spinup (Tenths-of-Seconds)  : 5

  4   Times to Try Disk Operations : 4


       Enter Selection :
```

Selecting one of the four items on this menu will prompt you for the information needed, allowing you to retain current settings as a default. An example of the additional prompts resulting from selection one on the above screen is:
cp

```
Configure which unit [0..3] : 2
    Size  8"(1),  5.25"(2),  3.5"(3)?        [2]     :
    Single or Double-Sided Drive ?           (S/[D]) :
    Motor On/Off Control Needed ?            ([Y]/N) :
    Motor Speed Standard or Hi-Density       ([S]/H) :
    Tracks-per-Side (35,40,80)               [80]    :
    Step Rate in Milli-Seconds               [3]     :
    Head Load Time in Milli-Seconds          [24]    :
    Head Unload Time in Milli-Seconds        [240]   :
```

6.2.2.4 Screen 4 - Hard Disk Parameters.

The B/P Bios Hard Disk Subsystem is centered around the Small Computer Systems Interface (SCSI) standard. Backward compatibility with the earlier Shugart Associates System Interface (SASI) is also provided to allow older controllers to be used. During the course of B/P Bios development, several controller types were incorporated, with unique features accommodated in a transparent way within the utilities. As a compromise between flexibility and program size, a limit of three physical hard drive units was placed on the system. This limit does not impact the 16 possible logical drives which the computer may handle. A sample Menu for a single Conner CP-3100 SCSI Drive of 100 Megabytes is:

```
Menu 4 - Hard Disk Options

  1   Hard Drive Controller = Seagate SCSI

  2   First Drive  :  Physical Unit 0,  Logical Unit 0
        No. of Cylinders = 776,        No. of Heads   = 8

  3   Second Drive : - inactive -

  4   Third Drive  : - inactive -


        Enter Selection :
```

The first available selection allows you to select the type of controller installed or used in the system. The controller definition applies across all three physical drives, so some care must be applied in mixing different controller types on the same system. For example, a Shugart 1610-3 controller is incompatible with a Seagate SCSI or SCSI-2 drive attached directly to the SASI bus since initialization information is required of drives on the 1610-3 which must not be sent to the SCSI due to the differing commands. The controller types defined in the initial release of B/P Bios appear in Selection 1 as:

```
Select Controller Type as:
        (1) Owl
        (2) Adaptec ACB-4000a
        (3) Xebec 1410a/Shugart 1610-3
        (4) Seagate SCSI
        (5) Shugart 1610-4/Minimal SCSI
        (6) SCSI-2

Enter Selection :
```

Selections two, three and four from Menu 4 allow you to specify the *physical* parameters of one of the three possible drives. As with other options, some of the parameters do not apply, such as Reduced Write and Precompensation with SCSI drives, but appear here to allow a general configuration tool. Current settings appear in square braces and are selected if only a Carriage Return is entered. A sample entry configuring the Conner CP-3100 is:

```
Activate Drive ([Y]/N) ?
Physical Unit (0..7)           [0]     :
Logical Unit Number (0..7)     [0]     :
Number of Physical Cylinders   [776]   :
Number of Heads                [8]     :
Reduced Write Starting Cylinder [0]    :
Write Precomp. Start Cylinder  [0]     :
```

6.2.2.5 Screen 5 - Partition Parameters.

Menu 5 permits arranging the physical drive complement into Logical Drives, and dividing physical units into multiple logical Partitions.

```
Menu 5 - Logical Drive Layout

A: = Unit 0, 64 Sctrs/Trk, 4k/Blk, 7984k (998 Trks), 1024 Dirs
B: = Unit 0, 64 Sctrs/Trk, 4k/Blk, 20000k (2500 Trks), 1024 Dirs
C: = Unit 0, 64 Sctrs/Trk, 4k/Blk, 20000k (2500 Trks), 1024 Dirs
D: = Unit 0, 64 Sctrs/Trk, 4k/Blk, 54432k (6804 Trks), 2048 Dirs
E: = Floppy 0
F: = Floppy 1
G: = Floppy 2
H: =    -- No Drive --
I: =    -- No Drive --
J: =    -- No Drive --
K: =    -- No Drive --
L: =    -- No Drive --
M: = RAM
N: =    -- No Drive --
O: =    -- No Drive --
P: =    -- No Drive --

     1 Swap Drives, 2 Configure Partition 3 Show Drive Allocations


        Enter Selection :
```

Selection 1 allows for swapping two specified logical drives. For example, the above screen shows a system which will boot from a hard drive. If the system were being configured for a Floppy-based system you might swap drive A with drive E with Selection 1 as:

```
                Enter Selection : 1
   Swap drive [A..P] : A  with drive [A..P] : E
```

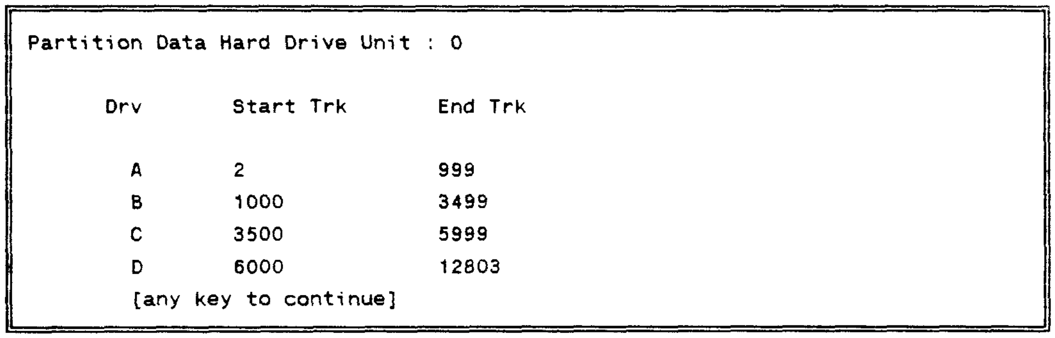
Selection 2 permits defining logical partitions on a Hard drive, and of the RAM drive, if active. It queries you for the needed information from which to set internal Bios values. If converting from an existing system, SHOWHD (See 6.19) will display the values to enter for a specified Partition. An example of the interaction is:

```
        Configure which Drive [A..P] : D
        Allocation Size (1, 2, 4, 8, 16, 32k)   [4]     :
                Number of Dir Entries   [2048]  :
                Starting Track Number   [6000]  :
                # Tracks in Partition   [6804]  :
                Physical Unit Number    [0]     :
```

Selection 3 from Menu 5 may be used in conjunction with the allocation to view the existing allocations for a given Hard Drive Unit. Selecting 3 prompts you for the Desired Hard drive as:

```
  Display Allocations for which Hard Drive [0..2] :
```

and will show the current Partitions and allocations. As an example, the four partitions on the Conner CP-3100 depicted in the Menu 5 screen are reported here on a separate screen as:



```
Partition Data Hard Drive Unit : 0

     Drv       Start Trk        End Trk

       A       2                999
       B       1000             3499
       C       3500             5999
       D       6000             12803
       [any key to continue]
```

6.3 BPDBUG - B/P Bios Debug Utility

This utility provides a low-level tool patterned after Digital Research's DDT, but extended to provide a more useful user interface, the ability to handle Z80 and Z180 mnemonics in disassembly, and memory banking using B/P Bios interfaces. While the description is primarily oriented to screen output, the Operating system permits also sending output to the defined Printer by toggling Control-P which may be enabled and disabled within BPDBUG.

6.3.1 Using BPDBUG.

The syntax for BPDBUG is simple with only three variants as:

```
BPDBUG //                 <-- Print a short help message
BPDBUG                    <-- Execute BPDBUG
BPDBUG [fn[.ft]]          <-- Execute BPDBUG, Loading named file
```

When executed, BPDBUG relocates the majority of the code to high memory immediately below the BDOS, overwriting the Command Processor. If a file load is specified as in the third method of invocation shown above, and the file type is .HEX, then a file in Intel HEX format is assumed and it is converted to binary form at the specified address.

In addition to the short help message available with the double-slash option, a built-in command summary is available at all times from the main BPDBUG prompt by entering a Question Mark. The summary appears as:

```
B/P Bios DBug  V 0.3
-?

Available commands:
  B{ank}  Bank#
  D{ump}  [From  [To]]
  E{nter}  Addr
  F{ill}  Start  End  Byte
  G{o}  Addr
  H{ex sum/diff}  Word1  Word2
  I{nput}  Port#
  L{ist}  [From  [Thru]]
  M{ove}  Start  End  Dest
  N{ame}  FN[.FT]  {for Read/Write}
  O{utput}  Port#  Byte
  R{ead file}  [Offset]
  T{race}   {Trace mode On}
  U{ntrace} {Trace mode Off}
  W{rite file}  Number_of_256-byte_blocks
  X {set breakpoint}  Addr
  Z{ero breakpoints}
  ? {Show this msg}
```

Square braces in the summary indicate optional parameters, while text strings and abbreviated names indicate parameters and types.

6.3.2 BPDBUG Commands.

6.3.2.1 Select Memory Bank. [B]

To select a memory bank in the range of 0..255, simply enter the Command "B" followed by the bank number in Hexadecimal. Optional spaces may be placed between the command letter and the Bank number. The syntax for this command is:

```
B[ ]nn
```

The Bank number selected by this command will be made the current bank for Display (D), Enter (E) and List (L) commands. If the specified bank number exceeds the largest bank number physically existing in the system, the bank number will be set to the last bank defined in the B/P Bios Header.

6.3.2.2 Dump (Display) Memory in Hex and Ascii. [D]

This command displays memory contents in both hexadecimal and ascii form. Where the current byte in the display is a control character (less than 20H), a period character is printed in the ascii column.

Defaults for this command are 100H and the TPA bank as the starting address, and 256 as the number of bytes to display if no End Address is specified. For subsequent uses of the Dump command, the Starting address will be one more than the ending address of the last Dump. The Bank Number will remain that last specified, or the TPA bank if never changed in a session. The syntax is:

```
D[ ][Start_Addr] [End_Addr]
```

A sample of the output appearing on the screen is:

```
-d100 12f
01:0100   ED73 FE03 31FE 0321 5D00 7E23 FE2F 2006   .s..1..!].~#./ .
01:0110   BE11 AD01 2825 2A01 002E 5A7E FEC3 2018   ....(%*...Z`.. .
01:0120   CDAC 0121 FAFF 197E FE42 200C 237E FE2F   ...!...~.B .#`./
```

6.3.2.3 Enter Values in Memory. [E]

This command permits entering values into memory. A period terminates entry.

```
E[ ][Start_Addr]
```

```
-e100
:41 43 44
:.
```

6.3.2.4 Fill Memory with Constant Value. [F]

Entire memory areas may be set to a single constant value with this command. All three arguments (Start, End and Value) must be specified, and the command will not be executed if fewer arguments are given. The syntax is:

```
F[ ]Start End Value
```

As an example, the following command sets the sixteen bytes from 100H through 10FH in the currently selected bank to binary Zero;

```
-f100 10f 0
```

6.3.2.5 Go (Execute) Program in Memory. [G]

Execution may be started at any arbitrary address with the "Go" command. If no target address is specified, 100H is assumed since it is the normal starting address of programs loaded into the Transient Program Area. The syntax of the command is:

```
G[ ][Address]
```

6.3.2.6 Hex Sum and Difference. [H]

Simple Hexadecimal addition and subtraction is performed with the "Hex" command. When the command is executed with two addresses, both their sum (modulo 65536) and difference (also modulo 65536) are displayed in that order. The syntax of this command is:

```
H[ ]Value1 Value2
```

For example, if an offset of 45H from a base of 0ED3FH was desired, the resulting g positive and negative address could be determined as:

```
-hed3f 45                 <-- Entered
ED84 ECFA                 <-- ..returned Sum and Difference
```

6.3.2.7 Display Value from Input Port. [I]

This command will read the desired Input Port in the range of 0 to 0FFFFH and display the resulting byte. Since 16-bit address calculations are used, this command will properly read the built-in ports of the HD64180 and Z180. The syntax of the Input command is:

```
I[ ]Port_num
```

6.3.2.8 List (Disassemble) Memory Contents. [L]

Disassembly of executable instructions in memory is accomplished with this command. As with the "Dump" command, the starting address defaults to 100H when first loaded, and is assumed to be the instruction following the last one disassembled for subsequent uses of this command if no address is explicitly entered. If no Ending address is specified, 23-26 bytes will be contained in the listing depending on the length of the last instruction. The syntax is:

```
L[ ][Start] [End]
```

A sample of an entry and the resulting output with an arbitrary program is:

```
-1120 127                                    <-- Entered
0120  CDAC01        CALL    01AC             <-- ..displayed
0123  21FAFF        LD      HL,FFFA
0126  19            ADD     HL,DE
0127  7E            LD      A,(HL)
```

Note that the actual bytes included in the disassembled instructions are also listed in contrast to other similar programs to provide additional information for you. Additionally, an extra blank line is displayed after all unconditional jumps and returns to serve as a visual representation as an absolute change in control flow. The mnemonics are standard Zilog Z180 codes.

6.3.2.9 Move Memory Contents. [M]

This command permits blocks of data to be moved within memory. Memory address bounds checking is performed to insure that overlapping addresses are handled correctly so that minor shifts in blocks of data may be accomplished. The syntax for this command is:

```
M[ ]Start End Destination
```

6.3.2.10 Set File Name for Read/Write. [N]

This command is used to set the file name and optional type prior to a read or write operation. The name remains active until changed or BPDBUG is exited. The syntax is:

```
N[ ]FileName[.FileTyp]
```

6.3.2.11 Send Value to Output Port. [O]

This command forms the complement of the Input command covered above. It sends a specified byte to the addressed Output port. The syntax is:

```
O[ ]Port_num Value
```

As with all arguments, if more digits than the number needed are specified, only the last two (for a Byte) or four (for an address) are used in the expression.

6.3.2.12 Read a File into Memory. [R]

This command reads the file specified by the Name command into memory at the default address of 100H (if no offset is specified), or at a starting address of Offset+100H. The Offset value must be specified in Hexadecimal. The syntax of the Read Command is:

```
R[ ][Offset]
```

When the file is loaded, you will be informed of the current setting of the default address for the base of current memory (PC value) and the byte after the last one loaded by the Read Command (Next). The display might appear as:

```
Next  PC
0880 0100
```

6.3.2.13 Activate Trace Mode. [T]

To assist in debugging programs, a Trace function is included which is activated with this command. Upon encountering a breakpoint (See X Command below) the program enters the Trace mode in which each instruction is trapped and the state of the processor displayed along with a Disassembled listing of the instruction. Entering a single letter "T" activates the Trace Mode.

A fragment of a program run with Trace On is:

```
S0Z0H0P0N1C0  A=00 BC=0000 DE=0000 HL=0000 SP=0100
    IX=A4AE  IY=FFFE          0100  C30B01        JP      010B

S0Z0H0P0N1C0  A=00 BC=0000 DE=0000 HL=0000 SP=0100
    IX=A4AE  IY=FFFE          010B  2A0500        LD      HL,(0005)
S0Z0H0P0N1C0  A=00 BC=0000 DE=0000 HL=52C3 SP=0100
    IX=A4AE  IY=FFFE          010E  CDBD07        CALL    07BD
S0Z0H0P0N1C0  A=00 BC=0000 DE=0000 HL=52C3 SP=00FE
    IX=A4AE  IY=FFFE          07BD  7C            LD      A,H
S0Z0H0P0N1C0  A=52 BC=0000 DE=0000 HL=52C3 SP=00FE
    IX=A4AE  IY=FFFE          07BE  B5            OR      L
S1Z0H0P0N0C0  A=D3 BC=0000 DE=0000 HL=52C3 SP=00FE
    IX=A4AE  IY=FFFE          07BF  C8            RET     Z
```

6.3.2.14 De-Activate Trace Mode. [U]

This command turns the Trace Mode Off so that subsequent execution occurs at full speed with no trapping. Entering a single letter "U" deactivates the Trace Mode.

6.3.2.15 Write File to Storage. [W]

This command is the complement to the Read command covered above. It assumes that the data to be written starts at 100H and writes the specified number of 256-byte blocks to the file last specified with a "Name" command. The syntax of this command is:

```
W[ ]#Blocks
```

6.3.2.16 Set Breakpoint. [X]

This command is used to tag locations within the program to be executed under BPDBUG which, when executed, will temporarily stop executing and either return to the BPDBUG prompt or print information for the Trace output. Up to two breakpoints may be active at any point in time. The syntax is:

```
X[ ]Address
```

6.3.2.17 Clear Breakpoints. [Z]

This command clears all breakpoints set with the X command cited above. Entering the single letter "Z" clears all breakpoints.

6.3.2.18 Display On-Line Help. [?]

Entering a single Question Mark ("?") as a command displays the Build-In help display containing a summary of the commands available from within BPDBUG.

6.4 BPFORMAT - Floppy Disk Format Utility

BPFORMAT is the general-purpose format routine for Floppy Disk Drives in the B/P Bios system. It automatically adapts to the specific hardware used in your computer to present a single interface across a wide range of platforms, and incorporates the ability to format disks in formats not implemented in your computer. This capability allows you to format disks for exchange with other users in their native disk format using the same library of alien disk formats used by the EMULATE program (see 6.8).

This program is B/P Bios-specific and will not function under other Bios systems. Its operation is the same under banked or unbanked systems, and with the many types of physical Disk Controller integrated circuits available. In the initial version, the following Controller types are supported:

765	1692	1770	1771	1772	1790	1791
1792	1793	1795	2790	8473	9266	

6.4.1 Using BPFORMAT.

6.4.1.1 Built-in Formats.

The simplest way of formatting diskettes is to use one of the formats included in the currently-running B/P Bios. BPFORMAT may be invoked by simply entering the program name, or by following it with a drive letter and colon as:

```
BPFORMAT D:
```

Optionally, a Named directory may replace the drive letter and will format the drive associated with the named directory. For example, if a directory named *WORK:* is defined to be Drive C:, User 10, the command

```
BPFORMAT WORK:
```

would format Drive C:. When invoked in either of the above manners, BPFORMAT will list the built-in formats available for this drive from those included in the file DPB.LIB (and optionally DPB2.LIB, see sections 4.1 and 4.2). Only those formats which exactly match the drive characteristics will be presented, so only 80-track formats will be offered for an 80-track drive and so forth. For example, the offerings for a 40-track 5.25" disk drive may result in:

```
Available formats are:

    A - Ampro DSDD     B - Ampro SSDD

Select format (^C to exit) :
```

As precautions against inadvertently formatting diskettes, confirmation prompts are included as the program progresses, and the opportunity exists to escape from the format program to Command processor. An example appears at this point where a Control-C aborts the format operation.

6.4.1.2 Library Formats.

If formatting of a diskette is desired in a format not supported by the built-in selections featured in the executing Bios, the library of formats used by EMULATE (see 6.8) may be used. Reasons for using the library may range from the need to format in a mode used on another type of computer to a choice made internally unavailable by sizing constraints, as when tailoring a system for Boot Track installation. Whatever the reason, this flexibility is offered as an inherent feature of B/P Bios and is specified by specifying the L Option when invoking BPFORMAT. If specifying the desired drive on the command line as in either example above, simply add the option character at the end (with optional slash) as:

```
BPFORMAT D: L
```

If you wish to be prompted for the drive letter as part of the program flow, the slash becomes mandatory to inform BPFORMAT that you are specifying the Library option instead of Drive L:. The invocation thereby becomes:

```
BPFORMAT /L
```

When executed with the Library option, you will be presented with a menu of formats which may be used with the physical drive as defined in the Bios header. A sample display appears as:

```
Available formats are:

  A - Actrx SSDD    B - Ampro SSDD    C - VT180 SSDD    D - H-100/4 1D
  E - H89/40  1S    F - H89/40  1D    G - H89/40  1X    H - Kaypro 2
  I - Osborne 1S    J - Osborne 1D    K - Ampro DSDD    L - H-100 DSDD
  M - H89/40  2D    N - H89/40  2X    O - QC-10 DSDD    P - Kaypro 4
  Q - MD-3  DSDD    R - PMC-101       S - Sanyo 1000    T - TV 802/803
  U - XBIOS-3 2D    V - XL-M180 T2

Select format (^C to exit) :
```

Entering one of the letters corresponding to a format will set all parameters and proceed with the format operation. Entering a Control-C at this point will return you to the Command Processor at this point avoiding any inadvertent disk formatting.

The Assembly source to the Format library is provided in the B/P Bios package as an aid in accepting formats not included in the default distribution package, or to experiment with new formats.

6.4.2 Configuration.

Two options exist for custom tailoring BPFORMAT to operate in a method you find most comfortable. The first is a Quiet option which will minimize extraneous output to the Console. It is set by a Boolean flag consisting of a Byte at an offset of 11 (0BH) bytes from the beginning of the program. A Zero byte in this location signifies Verbose operation where all defined prompts and status information is displayed. A Non-Zero value (normally 0FFH) indicates that Console output should be minimized with only essential output displayed.

The second option is for selection of the File Name and Type to be used for the Library of formats used with the L option. The default value of this entry is ALIEN.DAT (in formatted FCB form) which is also used with the EMULATE program. This field begins at an offset of 12 (0CH) bytes from the beginning of the program.

6.4.3 BPFORMAT Error Messages.

Must be wheel to FORMAT!!!

As a safety feature, only users with Wheel privileges may format diskettes. This error message identifies an attempt without the proper authorization.

***** ERROR ! Not B/P Bios, or Old Version !**

In most cases, this error will be seen if an attempt is made to format a disk under a Bios other than B/P Bios. If some of the mandatory data structures have been altered, or if an attempt is made to run the release version of B/P Bios under one of the early test versions of B/P Bios, this message will also be displayed.

***** ERROR ! The selected format is not supported by FORMAT!**

This error will be displayed if a format from a library of formats is incompatible with the specified drive, such as if a 5.25" format is selected for an 8" drive.

***** ERROR ! The detected FDC is not supported by FORMAT!**

The Bios reported a Floppy Disk Controller (FDC) that is not in the list of Controllers supported by BPFORMAT. To view the list of controllers supported, view the internal Help by using the double-slash option.

***** ERROR ! Disk is Write Protected!**

An attempt was made with the Write Protect Tab ON (for 5.25"), OFF (for 8") or in the Protect position (3.5") for the specified disk. Set the disk to Read/Write by altering the physical setting and try to format the disk again.

***** ERROR ! Disk won't recalibrate**

The drive heads could not be restored to Track 0 position. This error is often due to a failure in the drive mechanism, but can also be caused by deformed diskettes or loose drive cable.

Format Error : xx

An error was detected during the format process. The "xx" will be the Hexadecimal byte returned by the Bios portion of the format routine, and bits set to a "1" value should represent an error code decipherable from the FDC Data or programming sheet.

+++ Can't Open : fn.ft

BPFORMAT could not open the format library file. To access a format library, it must either be in the default library, or accessible from it either via the PUBlic bit or along the ZSDOS path.

No formats available for this drive!

This error will be reported if no formats (internal or from a format library, depending on how it was invoked) are supported on the specified drive. For example, if all internal formats are for 5.25/3.5" drives and the target drive is specified as an 8" drive, then no formats will be available if BPFORMAT is invoked using the internal format method of operation.

6.5 BPSWAP - Logical Disk Swap Utility

This utility allows you to exchange the drive letters defining two logical drives or partitions within the system. It performs any operations necessary to properly adjust the Operating System to account for drive redefinition, and relogs both drives using Dos Function 37 to force rebuilding of the Allocation Bit Map.

BPSWAP is a B/P Bios utility and will not execute under any other system. It may be operated in an interactive mode, fully "expert" mode with arguments passed on the command line, or a combination where the first drive letter is passed on the command line and the second entered in response to a query. If running in the interactive mode, entering a Control-C instead of a drive letter will interrupt the program and return to the Command Processor. BPSWAP is re-executable under ZCPR with the "GO" command.

6.5.1 Using BPSWAP.

6.5.1.1 Interactive Operation.

To execute BPSWAP in the interactive query/response mode, simply invoke the program by entering its name as:

```
BPSWAP
```

The program will insure that the system is running a B/P Bios, gather internal data from the operating environment, and display the prompt:

```
First Drive to Swap [A..P] :
```

At this point, a drive letter (upper or lowercase) should be entered within the specified range of "A" through "P". All invalid characters, except for Control-C which aborts the program, will result in repeated prompts for the first drive letter. When a valid drive letter is detected, the second is likewise requested with the prompt:

```
Second Drive to Swap [A..P] :
```

BPSWAP responds to entries at this point in an identical manner to the first, repeatedly prompting for a valid letter, or the abort character. When a valid letter is received, each logical drive is reassigned to the physical definitions of the other.

6.5.1.2 Command Line Operation.

BPSWAP can accept and parse drive letters passed to it on the Command Line in order to include drive exchanges in Startup scripts or other alias commands. To invoke the program in this manner, enter the program name with two drive letters in the range of "A" through "P" with a delimiter between each field. Each of the drive letters may be followed by an optional colon. Delimiting characters are TAB, space, and Comma. A summary of the complete syntax is:

```
BPSWAP <Drv1>[:] <tab| |,> <Drv2>[:]
```

To illustrate, the following are valid commands executing BPSWAP:

```
BPSWAP A: E:          <-- Exchange E drive with A
BPSWAP D,H            <-- Exchange D drive with H
```

If an invalid character is detected for either or both of the drive letters when called in the Command Line mode, operation automatically reverts to the Interactive mode and the respective prompt(s) will be given for valid drive letter(s). This feature permits a hybrid mode of operation to be specified wherein the first drive letter is passed on the Command Line, and the second entered in response to the second drive prompt.

6.5.2 BPSWAP Error Messages.

The only error message which may be printed by BPSWAP is in response to internal routines which validate the presence of a B/P Bios. Any attempt to run this utility on other Bioses results in the error:

```
+++ Not B/P Bios ... aborting +++
```

after which point the program aborts and control returns to the Command Processor. No effect on drive allocations will occur if this error is displayed.

6.6 BPSYSGEN - System Generation Utility

BPSYSGEN is our generic version of the classic SYSGEN program used to place an executable system image onto the boot sectors of a Floppy or Hard Disk. It uses information provided by the Bios in the form of DPB/XDPB data (see 5.2.3) which defines the physical and logical drive characteristics to write system information from the system tracks of one drive to another, or from an image produced by MOVxSYS (see 6.16) to the boot tracks of a drive.

6.6.1 Using BPSYSGEN.

6.6.1.1 Interactive Operation.

The basic Interactive mode is initiated by simply entering the program name at the Command Line prompt as:

```
BPSYSGEN
```

You will first be prompted for the source drive from where to obtain a bootable system image, then for a destination drive to save the image. To provide a visual clue that the program is executing, a series of periods is printed on the screen with each period representing a physical sector of data. At the conclusion of the operation, the program exits to the Command Processor prompt.

A binary file produced by MOVxSYS (see 6.16) may be placed on the system tracks of a hard or floppy disk by specifying the file name as a command line argument as:

```
BPSYSGEN B:ZSDOS64.BIN
```

When activated in this manner, you will be prompted for the destination drive letter after BPSYSGEN loads the image file and validates it as a valid system image. Alternatively, you may replace the file name with a drive letter followed by a colon to automatically load the image from a specific drive, and be prompted for the destination drive.

6.6.1.2 Command Line Operation.

A single operation may be completely specified from the command line arguments thereby avoiding drive prompts. When invoked in this manner, the first argument specifies the source for the system (drive designator or file) with the second argument being the drive specification on which to place the bootable system image. The syntax for the Command Line method of operation is:

```
BPSYSGEN {d: ¦ fn[.ft]} d:          <-- d is drive in [A..P]
```

6.6.1 BPSYSGEN Error Messages.

*** Read Error

An unrecoverable error was encountered reading either the boot tracks of a drive, or a specified bootable file.

*** Bad Source!

The source drive does not exist or could not be selected.

*** Write Error

An unrecoverable error was encountered writing the boot tracks of the specified destination drive.

*** Bad Destination!

The destination drives does not exist or could not be selected.

*** No System!

There are no valid System Tracks on the Source or Destination Drive, or an anomalous condition (more than 3 reserved tracks) was detected.

*** Can't Open Source File!

The specified boot image file could not be located, or an error occurred during the attempted File Open.

6.7 COPY - Generic File Copy Utility

COPY.COM is a file copy program derived from the ZCPR3 MCOPY tool written by Richard Conn. It blends the many modifications by Bruce Morgen, Howard Goldstein and others in MCOPY48 with further enhancements in the spirit of the ZSDOS environment. File date stamping is supported for the full range of stamping capabilities provided by ZSDOS. A user-definable "Exclusion list" is now supported to prevent copying of specific files or file groups, and two options to ease file backups with the Archive bit have been added. COPY is also more user-friendly than MCOPY, and provides increased error checking and user feedback.

COPY only operates in the Command Line Driven or *Expert* mode. As with the other utilities provided with ZSDOS, COPY displays a short Help message when invoked with a double-slash argument as explained in Section 1.2. The Help message also includes a list of available options along with the effect of each when included as command line arguments.

While COPY is ready to run without special installation procedures, you may wish to change the default parameters to customize it to your operating style. In this manner, you can minimize the number of keystrokes required to perform routine operations by avoiding passing many options on the command line. To set default conditions, insure that COPY.COM, COPY.CFG and ZCNFG.COM and available to the system, and execute ZCNFG as described in Section 4.8 of the ZSDOS 1.0 Manual.

6.7.1 Using COPY.

The basic syntax for COPY follows the original CP/M format by listing the destination drive/user, an equal sign, then the source drive/user and file name. An alternate syntax added by Bruce Morgen in MCOPY48 permits specifying transfers in the "Source-Destination" form popularized in MS-DOS. In this alternate form, you first enter the source drive/user and filename, a space, and then the destination drive/user and optional filename. Using the normal symbology, the syntax is summarized as:

```
        COPY dir:[fn.ft]=[dir:]fn.ft,... [/]options
    or
        COPY [dir:]fn.ft dir:,... [/]options
```

If no destination filename is specified, a number of unique files may be copied to a specified directory by catenating source files separated with commas. Where a destination file name is specified, both source and destination file names and types must be free of wildcard characters. This popular "Rename" feature in a copy was a much requested addition to the ZSDOS copy utility. Options to tailor the actions of COPY may be appended after the source file list.

Yet another method of transferring files was retained from the original MCOPY roots. If no destination drive/user is recognized in the command line arguments, all referenced files will be copied to a default drive/user location which is contained in the header portion of COPY. The default location is Drive B, User 0 in the distribution program, but may be changed as described below. If options are desired with this syntax, the slash option delimiter is *Mandatory*. The syntax for this method is summarized as:

```
          COPY [dir:]fn.ft,... /options
```

Various configuration options detailed later allow you to customize COPY to suit your operating style. For example, status displays of each operation may be suppressed for a "Quiet" mode, verification that copied files match the original (or at least produce the same error check code) may be enabled or disabled, etc. If a method of Date and Time Stamping is active under ZSDOS, the original Stamp information will be transferred to the destination file. The following examples in the "Verbose" method of operation will serve to illustrate by copying a file from the current Drive and User area to the same drive, User 10.

```
         COPY ZXD.COM 10:

      COPY  Version 1.71 (for ZSDOS)
      Copying C2:ZXD     .COM to C10:
       -> ZXD     .COM..Ok (Dated)  Verify..Ok
       0 Errors
```

In this case, No file of the same name existed in the destination area, but some form of File Stamping was active, so the source Stamp information was successfully transferred to the destination. Performing the same activity with the other syntax now produces:

```
          COPY 10:=ZXD.COM

       COPY  Version 1.71 (for ZSDOS)
       Copying C2:ZXD     .COM to C10:
        -> ZXD     .COM  Replace Same (Y/N)? Y..Ok (Dated)  Verify..Ok
        0 Errors
```

Since COPY now detected a destination file of the same name, and File Stamping as well as duplicate checking (another option flag) were in effect, COPY compared the Last Modified dates for both source and destination files. Finding a match, the prompt "Replace Same" was issued, and received a (Y)es response to copy the file anyway. Other responses, depending on the results of the date comparison are "Replace Older", which means that an older file exists on the destination, and "Replace Newer" which means that you are trying to replace a newer file on the destination with an older version.

A similar error check is made if a duplicate file is found to determine if the file was found with the PUBlic Attribute bit. If a Public file is detected on the destination drive, a warning to the effect is printed. Answering Yes to replacement at this point will result in a Read-Only error unless ZSDOS has been set to permit writes to Public Files (see 2.8.3 of the ZSDOS 1.0 Manual).

As stated earlier, COPY has no Interactive mode of operation per se, but the

Inspect option provides a means to select files for transfer in a somewhat interactive manner. In this mode, all files selected by the file specification in the command line are displayed, one at a time, and you may enter "Y" to copy the file, "N" to Not copy the file, or "S" to forget the rest of the selected files. An example copying all files from the current Drive and User to User 10 is:

```
    COPY *.* 10: /I

  COPY  Version 1.71 (for ZSDOS)
  Copying C2:????????.??? to C10:
   Inspect -- Yes, No (def), Skip Rest
  BU16    .COM - (Y/N/S)? Y
  BU16    .MZC - (Y/N/S)? N
  COPY    .COM - (Y/N/S)? Y
  COPY    .Z80 - (Y/N/S)? S
```

If operating in the Verbose mode, status on each file will be printed as the copies progress.

6.7.2 COPY Options.

Several option characters are available to customize COPY operations. Most of these options may be set as default conditions using Al Hawley's ZCNFG Configuration Utility. Alternatively, you may enter any of them on the command line to alter the functions of a single operation. The command line option characters are as follows:

```
A - Archive
E - Test for File Existence
I - Inspect Files
M - Multiple Copy
N - No replacement if File exists
O - Test Existence of R/O Files on Destination
Q - Quiet
S - exclude System Files
V - Verify
X - Archive Only if File exists
```

From the brief syntax summaries listed above, you will note that the standard option delimiter, a slash, is optional if both source and destination specifications are listed on the command line. If only one specification is listed, is when copying to the default drive, the delimiter is *Mandatory*. Each option is described in the following paragraphs.

6.7.2.1 Archive Option.

When this option is active either by specifying in the command line or as a default, only files which do *Not* have their Archive Attribute set will be selected. After the selected files are copied, the Archive Attribute on the Source file will be Set to indicate that the file has been "Archived". When used in conjunction with the default drive and user settings, the **A** option provides a simple method of archiving files in a single user area. The default for this option is *Off*, for No control of selection by the Archive Attribute. Adding the **A** option to the command line reverses the configured setting.

It should be noted that this option is incompatible with the "M" (Multiple Copy) option. The first copy operation will set the Archive bits on selected files, and they will not appear in subsequent copies.

6.7.2.2 File Existence Option.

This option controls the test for an already-existing file on the destination drive by the same name. Adding the **E** option to the command line argument reverses the configured setting. The default in the ZSDOS distribution version is *On*, or Check for Existing files. This option does *not* affect the check for PUBlic files on the destination drive, which is always active.

6.7.2.3 Inspect Files Option.

As illustrated previously, the I option provides a means of selectively copying files, without entering the name of each file. The distribution default for this option is *Off*, or do Not inspect the selected file list. Specifying this option on the command line argument list reverses the configured setting.

6.7.2.4 Multiple Copy Option.

This option may be used to copy a file, or group of files to the same drive several times, as when making several copies of the same file group on different disks. A prompt is given before each copy operation begins, and you may abort at the prompt, or change disks before beginning the copy. The distribution default for this option is *Off*, for No Multiple copying. Adding the **M** option to the command line argument list reverses the configured setting for this option.

6.7.2.5 No Replacement Option.

When added as a command line argument, the **N** option will not allow replacement of a file which already exists on the destination Drive/User. This option cannot be configured, and always assumes the same initial state when COPY is called. The default initial state for this option is *Off* to permit replacement of existing files.

6.7.2.6 Read-Only File Test.

This option, when added as an argument, reverses the configured setting of a flag which checks for the existence of file(s) satisfying the specified name and type with the Read-Only attribute set. If this flag is active and a Read-

Only file is located satisfying the criteria. the file will not be automatically overwritten. The E (File Existence) flag will still dictate how other files are handled.

6.7.2.7 Quiet Option.

When used on a system with ZCPR3, this option causes a reversal in operation of the ZCPR3 Quiet flag. If the ZCPR3 Quiet flag is active. COPY with the Q option operates in a Verbose mode. If you do not use ZCPR3, or the ZCPR3 Environment defines the Quiet flag as inactive, this option will disable unnecessary console messages for a Quiet mode of operation. There is no default condition for this option, and it is only effective for a single call of COPY.

6.7.2.8 System Files Option.

This option controls whether or not files with the SYStem Attribute set will be located by COPY. The distribution default is *Off* to include SYStem files in COPY file lists and permit copying of such files. The default may be configured as described below. and the default may be reversed by adding an S in the command line option list.

6.7.2.9 Verify Option.

To add a measure of confidence that no errors occurred in a COPY operation. the Verify option may be activated. When active. the destination file is read in order to compute a Cyclic Redundancy Check (CRC) word. This word is then compared to a value calculated when reading the source file. If the two values match. you can be *reasonably sure that the destination file is a true* copy of the source file. The distribution default for this option is *True* to verify each file copied. This option may be changed by configuration. or reversed by adding a V to the command line option list.

6.7.2.10 Archive if Only if File Exists Option.

Occasionally, you may wish to update frequently archived files to the same destinations in a simpler manner than naming each file, or by using the Inspect option. The X option was created for just this purpose. When this option is added. COPY first searches the source directory for files which have not been archived. then checks the destination directory for each file. If a match is found. the file is copied. and the source file deleted. unless it is marked as *Read-Only*. There is *No* configurable setting for this option which is always assumed to be OFF when beginning COPY.

6.8 EMULATE - Alien Disk Emulation Utility

EMULATE locks any or all Floppy Disk Drive(s) to specified formats, native or alien, from a Database of formats. It may also be used to display current settings and restore drives to auto-selection if the Bios was assembled with the AutoSelect option (see 4.2). The Floppy Disk format information is contained in a file named ALIEN.DAT whose use is shared with BPFORMAT (see 6.4). This sharing of a common database of formats allows formatting, as well as reading and writing of a large number of the hundreds of formats used by CP/M vendors over the years.

6.8.1 Using EMULATE

This utility is only usable with B/P Bioses which have been assembled with the Auto-Select option (AUTOSEL) active. This is the normal mode for release versions of B/P Bios, although some versions placed on the boot tracks of floppy disks may have a scaled-down complement of built-in formats to reduce the system image size (see 4.3). EMULATE can be executed either in an interactive query/response mode or in a command line "expert" mode with arguments passed on the command line. The EMULATE syntax is:

```
EMULATE //              <-- Print Built-in Help Summary
EMULATE [/]X            <-- List Current Floppy Format Settings
EMULATE [/]U            <-- Return All Floppies to Autoselect
EMULATE                 <-- Execute in interactive Query/Response mode
EMULATE d[:]            <-- Select format of Drive d: interactively
EMULATE d[:] [nn]       <-- Set Drive d: format to entry nn (expert)
```

To keep the numbering of formats in the Database file constant, thereby allowing the expert mode of configuration, all formats in the ALIEN.DAT file are loaded without validation against the actual drive parameters. Once a format is selected, the required drive characteristics (disk size, number of sides, speed and number of tracks) are compared to the physical drive parameters contained in the B/P Bios header structure (see 5.2.1, CONFIG+35). If the selected format can be accommodated by the physical drive, then the format information is loaded into the Extended DPH/DPB fields for the specified drive and the format locked to prevent re-assignment on warm boots.

The following formats are currently included in the ALIEN.DAT file, the source code for which is included in the distribution version of B/P Bios as ALIEN.LIB:

1	Actrx SSDD	2	Ampro SSDD	3	VT180 SSDD	4	H-100/4 1D
5	H89/40 1S	6	H89/40 1D	7	H89/40 1X	8	Kaypro 2
9	Osborne 1S	10	Osborne 1D	11	Ampro DSDD	12	H-100 DSDD
13	H89/40 2D	14	H89/40 2X	15	QC-10 DSDD	16	Kaypro 4
17	MD-3 DSDD	18	PMC-101	19	Sanyo 1000	20	TV 802/803
21	XBIOS-3 2D	22	XL-M180 T2	23	Ampro SSQD	24	DEC Rainbo
25	Eagle-IIE	26	H89/80 1D	27	H89/80 1X	28	Ampro DSQD
29	Amstrad WP	30	H89/80 2D	31	H89/80 2X	32	XBIOS-4 2Q
33	CCS SSDD	34	IBM 3740	35	Bower 8"1D	36	TTek SSDD
37	Bower 8"2D	38	CCS DSDD	39	TTek DSDD2	40	TTek DSDD1

Current drive format allocations may be examined at any time with the X option which will list the 10-character name the format assigned to each floppy drive in the system. or state that it is Autoselecting. The U option removes all fixed formats, returning them to Autoselecting.

6.8.2 EMULATE Error Messages

+++ Can't Open Database File +++

EMULATE could not locate the ALIEN.DAT file in the currently logged directory. Solutions include setting the PUBlic attribute of ALIEN.DAT and insuring that the Dos Path includes the drive containing the file.

+++ Format Not Supported on this drive!

Self-explanatory. Common causes of this error are selecting an 80-track format on a 40-track drive, or an 8" format on a 5.25" drive. Check the ALIEN.DAT source code to determine any needed data on the exact drive requirements for each format.

6.9 HDBOOT - Hard Drive Boot Utility (tailored)

HDBOOT is a specialized routine which is only available for those computers which feature the ability to boot from Hard Drives from a cold start such as the YASBEC and Ampro Little Board computers in the initial version. HDBOOT is a customized utility which is tailored for specific versions and will not execute on B/P Versions which it does not recognize. It modifies the boot record of a Floppy Disk System image placed on a drive by BPSYSGEN (see 6.6) to allow the system to be started from the Hard Drive at power-on or from a system Reset.

6.9.2 Using HDBOOT

HDBOOT is extremely simple to use, and accesses the B/P Bios Data structures of the target system for any system-specific data required, such as initialization parameters for the Shugart/Xebec controller types. When invoked, the existing system is checked to insure that it is a valid B/P Bios version. If valid, you will be asked to specify which of the three possible physical SCSI units to access, and from there on the operation is automatic. A sample screen for a successful execution of this utility is:

```
B/P HDBOOT Utility  V1.0  31 Aug 92
    Copyright 1992 by H.F.Bower/C.W.Cotrill

 Configure whicn unit for Booting [0..2] : 0

Target Controller is : Seagate SCSI
 ...Reading Boot Record...
 ...Writing Boot Record... Ok..

A0:BASE>_
```

It should be noted that a system must have been placed on the target unit with BPSYSGEN (see 6.6) before executing this utility, or an error message will be issued and the operation aborted.

6.9.3 HDBOOT Error Messages

*** No System!

The specified target Unit does not contain a valid Boot System. Place a valid Boot Track system on the unit with BPSYSGEN and execute HDBOOT again.

*** Invalid Unit Number ***

The Unit number specified on the command line is invalid. Either it is not "0", "1" or "2", or the unit is not active.

*** Invalid Boot Record ***

The Boot Record existing on the specified Unit is not valid for this type of Computer. Normal causes are no system currently exists on the specified unit or the system in place is not a valid one for this system. Both of these may be corrected by placing a system on the first physical partition of the unit with BPSYSGEN (see 6.6)

+++ Image is Not B/P Bios, or Wrong Version +++

The image read from the Boot Tracks of the specified system was not a valid version of B/P Bios. The two most common causes of this are: not placing a Boot System on the System Tracks with BPSYSGEN, or altering the fixed data structures of the Bios source code in a way which violates the standard layout resulting in a system which cannot be recognized.

+++ Unit Not Active! Run BPCNFG to Set Drives.

The specified Hard Drive Unit (0, 1 or 2) was not tagged as an active unit. This can be changed by first executing BPCNFG (see 6.2) on the executing memory system. then re-invoking HDBOOT.

+++ Not B/P Bios ... aborting +++

An attempt was made to execute this utility on a system which was not running under B/P Bios. Boot the system with a B/P Bios-equipped system and try again.

*** Read Error

An unrecoverable error occurred while trying to read the target system's Boot Record. This is most often due to media errors on the first cylinder of the target unit and cannot be rectified. Another cause may be an incorrect definition of the physical characteristics of the controller and/or drive.

*** Write Error

An unrecoverable error occurred while trying to write the modified Boot Record to the Hard Drive unit. If a second attempt at execution is unsuccessful. it probably indicates either an incorrect physical definition of the Hard Drive unit. or unrecoverable media errors on the first cylinder of the drive.

6.10 HDIAG - Hard Disk Format/Diagnostic Utility

HDIAG is a generic B/P Utility program to Run Diagnostics, Format, Verify and examine Hard Drive parameters using any of the defined controller types in a B/P Bios system where such capabilities are defined. The ability to select the controller type in the beginning of the program is allowed to enable you to check and initialize drives using controller types other than that defined in the executing Bios for added flexibility. The following controller types are handled in the initial B/P Bios release:

Adaptec ACB-4000A
Shugart 1610-3 / Xebec 1410A
Seagate SCSI
Shugart 1610-4 (Minimal SCSI)
SCSI-2 subset for Conner and others

6.10.1 Using HDIAG

This utility tool only operates in an interactive mode, so it is simply invoked with its name and no arguments (other than the standard double-slash Help request). When activated, it reads the controller type from the B/P Bios header structure and asks you if this is the controller type you wish to use. If you wish to use a different controller type, such as diagnosing a Seagate SCSI drive from a system which has an Adaptec controller for normal use, you may alter the controller definition for the remainder of the HDIAG session. The interaction through to the main loop prompt may appear as:

```
B/P Bios Hard Disk Utility  V0.6,  25 May 92

Controller = Adaptec  Ok ([Y]/N) ? : N
 [0] Owl, [1] Adaptec, [2] 1410/1610-3, [3] SCSI1, [4] 1610-4, [5] SCSI2 : 5

Functions:        F  - Format
                  V  - Verify
                  D  - Run Diagnostics
                  P  - Show Disk Parameters

Select (^C or ESC to Quit) :
```

6.10.1.1 Show Disk Parameters. [P]

If you are running HDIAG on a Hard Drive Unit which is already defined in the Bios and was previously formatted, or one of the self-identifying SCSI drives, then you may view the current drive parameters with the P command. The display varies with the controller type and the amount and type of information that is available. Some of the data may be from Bios definitions, and other data from either the controller (e.g. Adaptec) or the drive electronics (SCSI or SCSI-2). Samples of the forms of information are:

```
    Unit : 0     CONNER  CP3100-100mb-3.5          <-- SCSI1 setting

      Total Blocks = 204864   (12804 Eq. Tracks)
      Sctrs/Track  =  33
      Sector Size  =  512
      Interleave   =  1
      # Cylinders  =  776
      Num of Heads =  8

    Unit : 0                       <-- Adaptec ACB-4000. Syquest SQ-312 10 MB

      Total Blocks = 22140   (1383 Eq. Tracks)
      Sctrs/Track  =  18
      Sector Size  =  512
      # Cylinders  =  615
      Num of Heads =  2
      Reduced Wrt. =  615
      Precomp. Cyl =  615
      Step Rate    =  12 uS Buffered
      Media type   =  Removable
      Landing Zone =  615
```

6.10.1.2 Hard Disk Diagnostics [D]

Some drives feature built-in diagnostics routines which test the unit's electronics and media. Other systems simply execute the power-up sequence which generally includes a sequence of self-tests. Normally, only the re-initialize function can be relied on, and is included in the standard suite of HDIAG functions with this command. Sample output resulting from this function is:

```
      Select (^C or ESC to Quit) : D
      Unit Number [0..2] (^C or ESC to Abort) : 0
    Re-Initializing Unit : 0 ..Ok
     ..Waiting for Ready..
```

Pauses may occur in the execution of the sequence, most noticeably after the status prompt stating "Re-Initializing Unit" before the "..Ok" appears. Depending on the exact system, this time is often when the actual controller electronics are being checked. and may involve moving the drive head which can be a time consuming task. There is also a pause very often after the prompt "..Waiting for Ready..". particularly if the heads were moved and must be re-positioned over the outer cylinder of the drive. When the drive returns a ready status. then the main selection menu is again displayed and HDIAG is ready for another command.

6.10.1.3 Verify Drive Media [V]

This function permits evaluating the condition of a formatted drive to identify defects to a varying extent. By using a mix of defeating the Error Correcting code where possible. and enabling or disabling the Individual Sector checks. a relatively extensive. albeit often time consuming. non-destructive status of the drive unit may be obtained.

```
   Select (^C or ESC to Quit) : V
   Unit Number [0..2] (^C or ESC to Abort) : 0
   Verify Individual Sectors (Y/[N]) : N
Verifying Unit : 0      CONNER  CP3100-100mb-3.5

Block 891
   ...aborted...
```

6.10.1.4 Format Drive [F]

Setting all of the data storage areas on the disk to a constant value, and renewing the control information on the drive is the purpose of this function. It is destructive, and ***any data on the drive will be lost.*** For this reason, several checks are included in the program to insure that you do not inadvertently activate this command.

While most of the information needed to format a drive is available from either the built-in data which can be read from the drive or controller, or from the Bios data areas, some items are still required from the user. You will therefore be asked to provide any data necessary to format the drive. In the older SASI systems (1610-3, Xebec, etc) this can amount to a considerable number of entries. Fortunately, formatting of drives is not often required, and the method of formatting used in HDIAG is flexible enough to allow a wide range of devices to be connected.

6.10.2 HDIAG Error Messages

Several error messages will be presented for specific problems encountered in the operation of HDIAG. Many of the messages will be specific to certain operations, and others will change the specific information depending on the capabilities of the controller type selected. The most general of these concerns the SCSI/SASI Sense command. The Newer SCSI systems use "Extended Sense" which can return more information than the basic Sense values. When Extended Sense is detected, the "Key" value is displayed in many error messages rather than the basic "Sense" byte. Consult the programming manual for the specific controller or drive for the specific meanings of these bytes. Such a message will usually be displayed as:

```
                 Error!  (Comnd = xx)  Sense: xx
      or         Error!  (Comnd = xx)  Key = xx
```

Also, to provide additional information during operation of many of the functions, the raw status byte read from the controller when an error occurs is also displayed as part of an error message as:

```
                 (status = xxH)
```

The interpretation of the hexadecimal byte presented may be gathered from the programming manual for your specific drive or controller type.

+++ Not B/P Bios ... aborting +++

An attempt was made to execute HDIAG under a Bios other than B/P, or modifications made to the Bios altered the locations or values needed to correctly identify the system.

****** SCSI Block Length Error !**

This fatal error message will be displayed if the Command Descriptor Block returned from the Bios is too small to allow the extended commands needed for the requested operation. It usually results from alterations to the Hard Driver module which change necessary values.

****** Controller Not Readable !**

HDIAG could not read parameters from the drive or controller. This will only appear in the **R** (Read Drive Parameters) function, when the controller type is "Owl".

****** No Diagnostics for : <contyp>**

The Controller selected cannot perform Diagnostics in a way that HDIAG can access or perform the needed functions. This will only appear in the **D** (Perform Diagnostics) function.

****** Verify Not Available !**

The specific drive/controller selected has not been defined adequately to allow verification of the drive. This will only appear in the **V** (Verify) function when the controller type is "Owl".

+++ 1610-3 Initialization Error...Sense = xxH

An error was detected sending the initialization string to a Shugart 1610-3 or Xebec 1410A controller. Insure that this is the correct type of controller setting for your hardware configuration. It will only appear in the **V** (Verify) function.

6.11 INIRAMD - RAM Disk Initialization Utility

INIRAMD is a B/P Bios utility that initializes the Directory of a RAM Drive and optionally initializes it for DateStamper (tm), P2DOS, or both types of file stamps. It contains protective features to preclude inadvertent initialization of an already formatted RAM Disk, and may be command line driven for execution from within STARTUP scripts.

6.11.1 Using INIRAMD

This utility is designed to be operated with Command Line arguments, but features built-in defaults which can be configured by either overlaying bytes in the beginning of the program with new default settings, or by configuring with Al Hawley's ZCNFG tool. To execute with the default settings, simply enter:

```
INIRAMD
```

The complete syntax for INIRAMD is:

```
INIRAMD [d:][/][Q][D][P]
```

6.11.2 Command Line Mode

By entering arguments on the Command Line when INIRAMD is invoked, several internal default values can be set to the specific settings desired at the time. The first argument expected when parsing the Command Line tail is a Drive Letter. This is optional and will override the default drive M: built into the program. To specify a drive other then the default, enter:

```
INIRAMD d:
```

Extraneous prompt and status messages can be withheld during execution of INIRAMD by either setting the default "Quiet" flag embedded in the program, setting the "Quiet" flag in the Environment, or passing a Q as an argument when invoking the program. Initializing the RAM Drive in *Quiet* mode using the default drive is then accomplished by entering:

```
INIRAMD Q
```

You may also specify which types of Date/Time Stamps to add to the RAM drive during preparation with the P (for P2DOS Stamps) and/or D arguments. If not operating in the Quiet mode, INIRAMD notifies you which type(s) of Stamping methods have been added to the RAM Drive after the directory area is initialized to a blank value. Initializing drive M: for both types of Stamps then is initiated by entering:

```
INIRAMD M: PD
```

INIRAMD Error Messages

+++ Not B/P Bios ... aborting +++

An attempt was made to execute HDIAG under a Bios other than B/P, or modifications made to the Bios altered the locations or values needed to correctly identify the system.

+++ Already Formatted...Proceed anyway (Y/[N]):

This warning and prompt will be issued if INIRAMD detected the dummy file name used as a tag that the RAM Drive is already formatted. This is most often seen in systems that contain battery backed-up RAM and INIRAMD is invoked either directly or in a Startup alias script. To minimize the appearance of this message if you desire to have the RAM Disk initialized in the Startup script, include the following:

```
IF ~EX M:-RAM.000          <-- Assume RAM Disk is M:
INIRAMD M:                 <-- Assuming RAM Disk is M:
FI
```

+++ Drive d: does NOT Exist

Either the default or explicit drive specified in activating INIRAMD was has not been defined to the system. One possibility is that the RAM Drive was swapped for an undefined drive letter. Check the drive assignments with BPCNFG. Option 5 if you wish to check the Drive assignments.

+++ Drive d: is Not a RAM Drive!

Either the default or explicit drive specified in activating INIRAMD was a valid drive, but was not a RAM Drive. As with the previous error, one possibility is that the RAM Drive was swapped for another drive which was of another type. Use BPCNFG, Option 5 to check the Drive assignments.

6.12 INITDIR - P2Dos Stamp Initialization Utility

INITDIR prepares disks for P2DOS-type file stamping. It does this by replacing every fourth entry in the disk's directory tracks with a time and date entry which is prefixed with a special character (hexadecimal 21). Existing directory entries in the fourth position are then shifted to the first entry in the next logical sector and the initialized directory sectors are written back to the disk.

```
******************** W A R N I N G ********************
* INITDIR should not be run on disks containing valid Date-  *
* Stamper file stamps since it rearranges directory data.    *
* To install both DateStamper and P2DOS stamping on one      *
* disk. start with a blank disk, or one with no datestamps   *
* of either type and run both PUTDS and INITDIR on the disk  *
* before using it. Doing otherwise will invalidate any       *
* existing stamp data.                                       *
**************************************************************
```

6.12.1 INITDIR Interactive Mode.

Initdir can be run in either an interactive mode by simply entering its name at the Command Prompt. or in "Expert Mode" by specifying a drive letter as an argument on the command line. In the interactive mode. you will be asked for a drive letter which will specify the drive to initialize with P2DOS stamps.

If the DateStamper !!!TIME&.DAT file is detected on the disk. INITDIR issues a warning and asks if you want to proceed or not (see 6.12.2 below). To avoid the possibility of loss of Time and Date Stamp information from DateStamper. this routine should only be run on freshly-formatted or blank drives. The syntax of INITDIR is summarizes as:

```
INITDIR //        <-- Print summary help message
INITDIR           <-- Execute in Interactive Mode
INITDIR d[:]      <-- Initialize Drive D:
```

6.12.2 INITDIR Error Messages.

Directory already initialized
The selected disk is already prepared for P2DOS stamps.

Illegal drive name
The character entered was not in the range of "A" thru "P".

Not enough directory space on disk
The directory on the selected disk is more than three-fourths full. Not enough space is available to support P2DOS file stamps.

Directory read error
An error was encountered in reading the disk directory.

Directory write error

An error occurred while writing the initialized directory. It will probably result in loss of file data.

--> DateStamper !!!TIME&.DAT File Found <--
Proceed anyway (Y/[N]) :

The special DateStamper !!!TIME&.DAT file exists on the disk. If other files are also on the disk, most of the DateStamper time and date information will be lost. On freshly-formatted or empty disks, no DateStamper file stamp data exists, so it is safe to answer with a Y and initialize the disk.

6.13 INSTAL12 - Boot Track Support Utility

INSTAL12 - Install CPR, ZSDOS, B/P Bios in a MOVxSYS "type" image
from standard size (2k CCP, 3.5k DOS, 4.375k Bios) files

INSTAL12 is the latest modification to the ZSDOS INSTALOS utility distributed with ZSDOS 1.0 which automatically overlays your computer's *System Image file*, such as MOVCPM.COM (CP/M) or MOVZSYS.COM (ZRDOS) program, or *Absolute System Model file* (e.g., CPM64.COM) with ZSDOS or ZDDOS to produce a new file containing ZSDOS/ZDDOS instead of your original Basic Disk Operating System. INSTAL12 also allows you to set the defaults of various ZSDOS parameters during the installation process (these parameters may also be changed later with the ZSCONFIG program).

INSTAL12 is designed to make the installation process as easy as possible. With INSTAL12 you may load files from all drives and user areas from A0: to P31:. Error detection is extensive, and Section 6.13.3 of this manual fully explains all INSTAL12 error messages. Finally, you may safely abort INSTAL12 at nearly all points by pressing Control-C.

Before using INSTAL12, ensure that any necessary files from your B/P Bios and/or ZSDOS Distribution Disk are present:

- o MicroSoft .REL formatted assembly of B/P Bios (MOVCPM = YES)
- o ZSDOS.ZRL (if replacing Operating System)
- o ZCPR33.REL, ZCPR34.ZRL or other Comnd Proc (if replacing CPR)
- o INSTAL12.COM

The following file from your B/P Bios Distribution Disk, CP/M or ZRDOS System Disk must also be accessible:

- o MOVxSYS.COM (B/P Bios), MOVCPM.COM (CP/M), MOVZSYS.COM (ZRDOS), or System Image file for systems such as the *Oneac ON!*

6.13.1 Using INSTAL12.

To run INSTAL12, most users should simply enter

INSTAL12

at the Command Prompt. This tells INSTAL12 that you are installing a system segment over a System Image relocation file, such as MOVxSYS.COM, MOVCPM.COM or MOVZSYS.COM. If you need to install a segment over an Absolute System Model file such as a CPM59.COM, ZSYSTEM.MDL, or Oneac ON! file, you should enter

INSTAL12 /A

to run INSTAL12 in *Absolute* mode. INSTAL12 now displays its opening banner and requests the name of a file as:

```
System Image file to patch (Default=MOVCPM.COM) :
```

in Relocatable mode, or

```
Absolute System Model (Default=SYSTEM.MDL) :
```

in Absolute mode.

You need not enter all of the information; INSTAL12 will fill in any missing items with the default disk, user, or filename. If you simply hit RETURN, INSTAL12 searches the current directory for the default System Image or Absolute System Model file (MOVxSYS.COM, MOVCPM.COM or SYSTEM.MDL). Here are some sample responses:

```
System Image file to patch (Default=MOVCPM.COM) : B3:
        (Selects MOVCPM.COM on drive "B" in user area 3)

System Image file to patch (Default=MOVCPM.COM) : 10:MOVYSYS
        (Selects MOVYSYS.COM on the current drive, user 10)

System Image file to patch (Default=MOVCPM.COM) : C:MOV18SYS.OLD
   (Selects MOV18SYS.OLD on drive "C", current user area)
```

Once INSTAL12 finds the requested file, it validates your operating system image. If the CCP, BDOS or BIOS portions of the System Image or Absolute System Model file are invalid, INSTAL12 prints an error message and quits at this point. This may occur if an Absolute System Image was loaded but INSTAL0S was invoked without the /A suffix. If both methods of calling INSTAL12 fail, first ensure that your system image or generation program is operating properly. If you are sure that you have a working MOVxSYS, MOVCPM, MOVZSYS, or Absolute Model file that INSTAL12 cannot validate, you will need to contact your distributor who will initiate actions to correct your problem.

If all values in your operating system file match expected parameters, a summary of those values is displayed. If you specified a System Image file (e.g., MOVxSYS.COM), the display should be similar to:

```
Addresses in system image (as seen under DDT) :
        CCP : 0980H        Map @ 3610H
        BDOS: 1180H        Map @ 3710H
        BIOS: 1F80H        Map @ 38D0H
```

The addresses shown will probably differ from these, but if both columns display values other than 0000H, INSTAL12 will correctly overlay the three system segment portions of the image with specified files.

If you specified an Absolute System Model, the display will be similar to:

```
          Addresses in system image (as seen under DDT) :
               CCP : BC00H
               BDOS: C400H
               BIOS: D200H
```

As above. the addresses will probably differ from those in the example, which are for a 54K system.

If no error message appears. INSTAL12 has properly validated your file. Next, a menu of choices appears:

```
               1 - Replace CCP
               2 - Replace DOS
               3 - Replace BIOS
               4 - Save and Exit
          Enter Selection (^C Quits) : _
```

To install a new B/P Bios image from your assembly. select option 3. You will be asked to enter the name of the file as:

```
          Name of BIOS file (Default=CBIOS.REL) : _
```

The default file type is REL and the file *MUST* be in MicroSoft REL format. When a terminating Carriage Return is entered. either the name you entered or the default will be located. If found. the size will be evaluated against the available space in the image file. Often when adding a B/P Bios to an older system generation program. the bit map portion of the program will be relocated. You will be notified with a message signifying the distance in bytes that the map was relocated. This is simply a diagnostic tool. and you should not be alarmed at the message. Following the message "...overlaying BIOS...". INSTAL12 will return to the main menu for the next command.

Replacement of the Command Processor with ZCPR33.REL or ZCPR34.ZRL is identical to Bios replacement. except that no relocation of the bit map is possible. The specified file will either fit and overlay the original. or it will be too large and the program will exit with an error message to that effect.

For ZSDOS installation. enter a 2. You will be asked for the name of a Disk Operating System file as:

```
          Name of DOS file (Default=ZSDOS.ZRL) : _
```

The default file type at this point is ZRL. but operating systems in MicroSoft REL format such as distribution versions of ZRDOS are also accepted. As above. you may respond with a full or partial file specification and INSTAL12 will fill in any missing items with the default disk. user. or filename.

Once the Disk Operating System file is found the following prompt appears:

```
          ZSDOS.ZRL Size OK...overlaying BDOS..
          Examine/Change ZSDOS parameters ([Y]/N)? : _
```

At this point, INSTAL12 allows you to change the startup settings of all ZSDOS options. If this is your initial installation of ZSDOS, we recommend that you press N for "No" to bypass this step, and skip the following paragraph.

If you enter any character other than N or n, the default option in brackets ([Y] for "Yes") is assumed, and INSTAL12 displays the current ZSDOS defaults as:

```
        1 - PUBlic Files            : YES
        2 - Pub/Path Write Enable   : NO
        3 - Read-Only Vector        : YES
        4 - Fast Fixed Disk Log     : YES
        5 - Disk Change Warning     : NO
        6 - Path w/o System Attr    : YES
        7 - DOS Search Path         : Disabled
        8 - Wheel Byte Protect      : Disabled..Assumed ON
        T - Time Routine (Clock)    : Disabled
        A - Stamp Last Access Time  : Disabled
        C - Stamp Create Time       : Disabled
        M - Stamp Modify Time       : Disabled
        G - Get Date/Time Stamp     : Disabled
        S - Set Date/Time Stamp     : Disabled
    Entry to Change ("X" if Finished) : _
```

These options are presented in the same manner by ZSCONFIG, and are fully described in Section 4.10 of the ZSDOS 1.0 manual.

Once you bypass the configuration step or exit by pressing X, one of the following prompts appears depending on whether you are installing an Image or Absolute Model file:

```
    Name to save new system (Default=MOVZSDOS.COM) : _
```

or

```
    Name to save new system (Default=ZSSYS.MDL) : _
```

Again, you may respond with a full or partial file specification and INSTAL12 will fill in any missing items with the default disk, user, or filename. If a file with the same name exists, INSTAL12 prompts you for a new name. When INSTALOS has a valid name, it creates your new system file and exits, displaying one of the following messages:

```
    ..Saving MOVZSDOS.COM                  - relocatable
```

or

```
    ..Saving ZSSYS.MDL                     - absolute
```

6.13.2 INSTAL12 Error Messages.

Occasionally INSTAL12 may issue error messages. Most errors result when the files you specified do not conform to INSTAL12' expectations. Often the solution is to run INSTAL12 again, specifying relocatable mode instead of absolute mode or vice-versa. Many INSTAL12 errors will also result from damaged files. If INSTAL12 gives errors in both absolute and relocatable modes, try recopying the source file from masters, or re-assemble the source program and execute INSTAL12 again.

If all of the above fail, your system files may contain information which INSTAL12 cannot recognize. You may be able to attempt an alternate installation with NZCOM or JetLDR for CPR and DOS segments, but you may need to contact the experts on Ladera Z-Node for assistance with Bios-related problems.

The following is a summary of all INSTAL12 error messages, their meanings, and some possible remedies.

***** SORRY! ZSDOS will only run on Z80 type computers!**
ZSDOS and its utilities will only operate on processors which execute the Z80 instruction set such as the Z80, NSC-800, Z180 or HD64180. There is no fix for this condition other than to run it on another system.

***** Unable to open [filename.typ]**
INSTAL12 cannot locate or open the system file you specified. First, ensure that the file is at the default or specified drive/user location. If you have specified the file correctly but this error persists, obtain a fresh copy of your system file and try again.

***** Can't find CCP/BDOS/BIOS at standard locations !!!**
The operating system contained in your system file is not a standard CP/M system. It contains a CCP which is not exactly 2 kilobytes long, a BDOS which is not exactly 3.5 kilobytes long, or both. If this message appears, first ensure that your system file has not been damaged. If you still receive this message, contact the authors on Ladera Z-Node or your distributor.

++ Image Vector does not match Calculations ++
INSTAL12 found an internal error in the image file while installing a MOVCPM-type file. If you did not use the /A option when running INSTAL12, you may be trying to perform a relative installation on an absolute file. Try running INSTAL12 again with the command INSTAL12 /A.

***** Cannot find legal Relocation Bit Map**
INSTAL12 was unable to locate a valid relocation bit map pattern in the MOVCPM-type file when installing in *Relocatable* mode. Non-standard relocatable image files are the general cause for this error. A workaround is to generate an Absolute Model with MOVCPM first, then use INSTAL12 in *Absolute* (/A) mode on the Absolute Model file.

---Can't find [filename.typ].. reenter (Y/[N]) :
The replacement file (CCP, BDOS or BIOS) specified cannot be located. Ensure that the drive, user and file name are correct.

***** Error in .REL sizing [filename.typ]**
Err Code : nn
An error occurred during the sizing operation of INSTAL12 on the REL or ZRL file. The REL or ZRL must be in MicroSoft relocatable format.

Named Common segments other than _CCP_, _BDOS_, and _BIOS_ are not allowed, and code and data segments (if any) must not overlap.

***** file too large to fit...**

The size of the relocatable CCP or BDOS is greater than the available space in the image file (2048 bytes for the CCP, 3584 bytes for the BDOS). This error may result if the relocatable file is not in proper MicroSoft REL format, or if a customized file is used. This error should never occur with the distribution ZSDOS.ZRL file, which is exactly 3584 bytes (3.5k) long.

***** Error opening : [filename.typ]**

INSTAL12 could not open the specified relocatable file. Ensure that you selected a valid REL file.

***** Error reading : [filename.typ]**

INSTAL12 detected an error when reading the specified relocatable file. Try recopying the file.

***** Error in .REL file : nn**

An error was found in a relocatable input file while attempting to replace the CCP, BDOS or BIOS portions of your operating system. **nn** is a hexadecimal code which may assist in locating the cause of the error. Contact your distributor if you need help in resolving an error of this nature with the code in the error message.

--- That file already exists. Overwrite it (Y/[N])?

The file you told INSTAL12 to write to already exists. If you enter Y here, INSTAL12 will erase the previous copy and create a fresh file with this name. Enter **N** to select a new name.

***** No Directory Space for [filename.typ]**

There was not enough directory space for the output file on the selected disk. Send the output file to a different drive by preceding the filename with a drive specifier, or change the disk in the output drive.

***** Error writing file. Try again with another disk (Y/[N])? :**

This message usually results from a lack of disk space on the drive you specified for output. Change disks and enter Y to try again.

6.14 IOPINIT - IO Package Initialization Utility

IOPINIT initializes an IOP Buffer defined in the Environment Descriptor to the standard Dummy IOP format and patches it into the Bios Jump Table. It serves the same basic function as the older ZCPR3 method of loading a SYS.IOP file, but was added as a stand-alone routine to do essentially the entire installation of the package. In so doing, additional space was freed in the B/P Bios core code allowing other routines to be added which cannot be removed to external programs.

6.14.1 Using IOPINIT

This program should be included near the beginning of the initial STARTUP script for any system in which an IOP is defined. NOTE: This routine *MUST* be run before any programs which change the Warm Boot Jump at location 0!

No arguments are expected when calling IOPINIT with all values determined from the executing system Environment. The routine responds to the normal double-slash help request as with all support routines.

6.14.2 IOPINIT Error Messages

--- No IOP Buffer defined in Environment ---

Self-Explanatory. If the IOP Buffer has been deliberately removed during configuration or assembly, no harm will be caused by executing IOPINIT.

--- No Z-System Environment Defined ---

This message should *NEVER* appear since a valid Environment Descriptor is REQUIRED in B/P Bios equipped systems. If it does, one possible cause is incorrect value(s) at critical points within the Descriptor that are used to validate the Environment.

*** IOP Already Installed! ***

Self-Explanatory. No harm is done to the system, this message is simply for information.

6.15 LDSYS - System Image File Loader

LDSYS is the primary utility to activate a System Image file prepared by BPBUILD (see 6.1). It first validates the currently-running system, then loads the image file, places the component parts where they belong in the computer's memory, and executes the Bios Cold Boot routine of the newly-loaded system. Image files may be either banked or unbanked and need not be placed in the currently-logged directory, since LDSYS can access files along the system path, or from Z3-style Path specifications.

6.15.1 Using LDSYS

This utility provides the only way to install a banked system in a B/P System, and a simple way to test non-banked systems before final conversion to bootable systems to be loaded onto system tracks using INSTAL12, MOVxSYS and BPSYSGEN. LDSYS expects only a single parameter to be passed on the Command Line, that being the name of an Image file to load. If no File Type is explicitly entered, a type of .IMGis assumed. The location of the desired file may be explicitly stated in normal ZCPR3 fashion with either DU: or DIR: prefixes. The overall syntax of LDSYS is therefore:

```
LDSYS [du¦dir:]name[.typ]
```

When loading, two summary screens are displayed, the first from LDSYS itself, and the second from the Cold Boot routine in the loaded system after control is transferred from LDSYS to the newly-loaded system. A sample display from a banked system during development when installed on a MicroMint SB-180 is:

```
B/P Bios System Loader   Vers 1.0  31 Aug 92
 Copyright (C) 1991 by H.F.Bower & C.W>Cotrill

 CCP starts at     : CC80  (0F80H Bytes)
 DOS starts at     : DC00  (0F00H Bytes)
   Banked Dos at   : 1080  (0500H Bytes)
 BIOS starts at    : EB00  (0880H Bytes)
   Banked Bios at  : 1580  (1269H Bytes)

 ...installing Banked System

 SB180 B/P 60.25k Bios  Ver 0.6  26 Jan 92 (Banked) with:

   ZCPR3+ Env
   ZSDOS Clock
   Hard Disk Support
   Warm Boot from RAM
   RAM Disk (M:)
   Full Error Messages

 _
```

All messages in the above sample screen through the line "...installing Banked System" are printed by LDSYS. All subsequent lines in the above screen are displayed from the newly-loaded Bios. During alteration, or modification of a new system, this subdivision in the display areas may be a clue to any difficulties encountered. The position of the cursor at the bottom of the sample screen is the point at which the new Bios, now in control, attempts to load the Startup file defined in the B/P Header. If none is found, additional initialization will not be performed, and you will see only the prompt for Drive A, User 0.

6.15.2 LDSYS Error Messages

***** No file specified ! *****

LDSYS was called without a specifying file to load. You may re-invoke it directly with the ZCPR "GO" command as:

```
GO filename
```

or call it in the normal fashion specifying an image file to load.

--- Ambiguous File: fn[.ft]

At least one question mark (or expanded asterisk) was detected in the specified file to load. Re-execute with an unambiguous file name.

--- Error Opening: fn[.ft]

The specified image file could not be located. Common causes for this are a mismatch in the file name, no file with the default type of "IMG" or an inability to find the file along the Dos Path or in a PUBlic directory.

6.16 MOVxSYS - Boot Track System Utility

This routine is a program to generate a bootable system image for the boot tracks of a Floppy or Hard Disk. It is customized for each type of hardware system using the B/P Bios. The generic name **MOVxSYS** translates to a specific name reflecting the standard name ID, examples of which are:

MOVYSYS	YASBEC
MOVAMSYS	Ampro Little Board
MOV18SYS	MicroMint SB-180

6.16.1 Using MOVxSYS

This program is patterned after the original MOVSYS.COM distributed with most Digital Research CP/M 2.2 systems, but extensively updated to reflect the Z-System standard Help, entry of a base address or system size in kilobytes, and additional checks needed to insure B/P standards.

Two basic parameters may be passed to this program as arguments on the command line. The first specifies the system size in either the equivalent number of kilobytes in an equivalent CP/M 2.2 system, or as the base address of the Command Processor. MOVxSYS parses the first element to determine if the value is a Hexadecimal number ending in the letter "H", and assumes that the value specifies a starting address if so. Valid addresses must be greater than 8000H to insure that the resident portion of the operating system in the Common Memory area. If the argument is not a Hexadecimal address, it is assumed to be a number of kilobytes in the range of 39 to 63. These sizes are based on the standard CP/M component elements of 2k bytes for the Command Processor, 3.5k bytes for the Basic Disk Operating System, and at least 1.5k bytes for the resident Bios portion. Several checks are performed within MOVxSYS and the initial executing portion of the Bios (Cold Boot) to detect invalid locations and incorrectly sized data areas.

The second parameter which may be specified on the command line is an optional asterisk ("*") or File Name and Type after the size specification. If an argument is present, one of two actions will be taken. The Asterisk instructs the program to relocate the system image to the specified size, and simply retain it in memory upon exiting without saving the image to disk. Any other characters will specify the name of a file under which name the image should be written to disk. If no second argument is given, the image will be written under a file name of SYSnnK.BIN where "nn" will be the number of kilobytes in the system size described above rounded down to the nearest even number.

Placing a system image on the Boot Tracks of a disk is done by BPSYSGEN (see 6.6) which may be done by immediately following MOVxSYS (invoked with the asterisk argument) by BPSYSGEN using the "Image in Memory" selection, or by specifying a file output from MOVxSYS, and invoking BPSYSGEN with the name of the resultant file.

If you reconfigure the Bios for your system with the goal of modifying the Boot System, you must assemble B/P Bios in the Non-Banked Mode by setting the

BANKED equate to NO. and setting the MOVCPM equate to YES in the DEF-xx.LIB file (see 4.2). The revised Bios may then be added to MOVxSYS.COM with INSTAL12 (see 6.13) to produce a customized Boot System reflecting your tailored needs. Refer to Chapter 3 for a more complete description of the installation process.

6.16.2 MOVxSYS Error Messages

****** Start < 8000H !!!**

A size value or CPR Starting address was specified which results in a base address less than 8000H. Since the lower 32k of memory (0..7FFFH) may be banked, the CPR *MUST* be in the upper half of memory. Execute the program again with an adjusted size specification.

****** Create Error !**

The program could not create the specified Binary output file. Possible causes are a full directory, Write Protected diskette, or bad media or drive.

****** Write Error...Exiting**

An error occurred while writing the specified Binary output file. Possible causes are a media or drive error, or a disk with inadequate storage space for the file.

****** Close Error !**

An error occurred while attempting to close the specified Binary output file. This is usually due to a media or drive error.

****** Size must be in 39..63 !!**

MOVxSYS was invoked with an invalid size (number of K) specification. Execute the program again with an adjusted size specification.

****** Bad Syntax !!**

The program became confused and could not properly decide what you wanted it to do. Review the built-in help by entering:

MOVxSYS //

and follow the syntax listed.

6.17 PARK - Hard Drive Head Park Utility

PARK is a simple B/P Utility routine that moves the Hard Drive heads on all drives to the designated landing zone (also called shipping or park zone) if defined for the type of Controller/Drive in your system. When all drives are parked, the utility executes a HALT instruction which requires a hardware reset or power-off/power-on sequence to overcome. To avoid the possibility of Hard Drive damage by removing power from drives while the heads are positioned over data storage portions of hard drives, PARK should always be executed prior to turning your computer off. This recommendation is particularly important for drives which do not feature automatic hard parking, or where such hardware features have failed.

6.17.1 Using PARK

This utility is simply called with no arguments, and sequentially scans all three possible Hard Drive units, executing the SCSI "Stop Unit" command on each. When all three units have been processed, the processor disables interrupts and executes a HALT instruction to prevent the units from becoming re-activated by subsequent instructions. Normally, only cycling the power or pressing the Reset button on the computer will allow processing to resume. Developing the habit of executing HALT before turning your computer off may result in increased life from your hard drives, and should become routine.

This utility is a specialized version of SPINUP (see 6.20) which permits individual units to be turned off and on during normal operation.

6.17.2 PARK Error Messages

+++ Not B/P Bios ... aborting +++

An attempt was made to execute PARK under a Bios other than B/P, or modifications made to the Bios altered the locations or values needed to correctly identify the system.

+++ Can't Park this type of Controller! +++

PARK was executed with a type of Controller or drive in the Bios that does not implement the "Stop Unit" SCSI function.

**** SCSI Block Length Error !

The Bios does not support the Extended Commands necessary to park the heads using the "Stop Unit" SCSI Function. This is most probably due to changes during an edit/assembly of the Bios which altered either the Command Descriptor Block size, or the Hard Drive function which returns the values.

6.18 SETCLOK - Real-Time Clock Set Utility

This utility provides a means of setting a B/P Bios clock from a physical clock contained in the ZSDOS CLOCKS.DAT library. It presents a similar interface to the ZSDOS 1 utility TESTCLOK from which it was derived.

6.18.1 Using SETCLOK.

SETCLOK is invoked by entering its name with an optional Clock Driver Number. For initial testing, or trying different clocks (always a dangerous procedure), simply enter the utility name as:

```
SETCLOK
```

You will be asked whether to extract clocks from a library. If you are using a custom clock, answer No. If you wish to use a clock from the prepared ZSDOS library, enter Yes which is the default setting if a Carriage Return is entered. You will also be asked for the location (Drive/User) of the clock file or library. This prompt sequence may appear as:

```
Extract Clock from Library ([Y]/N) : _
Location of CLOCKS.DAT [B0:] : _
```

Drive B, User 0 illustrated in the above sample prompt will probably differ in your system with the current drive and user always shown as the default location. If the file is on the currently-logged drive and PUBlic, it will also be found without specifying a unique User area. If the default reflects the location of the CLOCKS.DAT file, or a location accessible via the PUBlic feature, simply enter a carriage return, otherwise enter the location of CLOCKS.DAT, followed by a colon and a carriage return. A list of over 40 available clocks will appear. To select one of these clock drivers, enter the number corresponding to the clock, and the program will do the rest. Various messages will be displayed as the clock driver is loaded, linked and executed. If all goes well, the final message will be the Date and Time read from the clock followed by message that the B/P Bios Clock was Set Ok.

Alternatively, a clock driver may be selected for automatic execution according to a specification on the command line. To use this mode, the CLOCKS.DAT file must either be in the currently-logged Drive and User, or on the current drive with the PUBlic Attribute bit set. the syntax for this method of setting the B/P Bios clock is:

```
SETCLOK nn
```

where "nn" is a number corresponding to one of the clocks in the CLOCKS.DAT file. This method of operation may be used to set the Bios clock within an alias script such as STARTUP.COM commonly used when the computer is first booted.

6.18.2 SETCLOK Error Messages

Some of the errors in the SETCLOK utility are generated by the top-level program. These errors consist of:

+++ This is only for Z80 type computers!!!
This routine will only operate with Z80 or compatible processors since it is a B/P Bios utility which is also restricted to these types.

-- Error in locating Clock file
This routine could not find the CLOCKS.DAT Library. Insure that the library either exists in the currently-logged directory, can be found via the PUBlic feature, or is available along the DOS Path.

Other errors are generated in the process of extracting and validating the driver selected from the CLOCKS.DAT library. Such errors consist of:

-- Error Opening : clockname
The Selected Clock driver in the Library could not be opened. This is most often due to corruption of the CLOCKS.DAT file. Restore it from your ZSDOS backup disks and try again.

-- Error Reading : fn.ft
An error occurred while reading the Clock code from the library. This also is most often due to corruption of the CLOCKS.DAT file.

-- Error in : clockname
An error occurred in the logical relocatable structure of the selected clock driver.

-- Error initializing DAT file
An error was encountered in initializing the CLOCKS.DAT Library.

-- Memory overflow in DAT file
A memory allocation error occurred in the Clock routine which caused the allocated memory to be exceeded. This should not occur in any of the library clock drivers, but may be experienced if the guidelines are not followed when developing a custom clock.

Still other errors relate to the reading and linking of the selected clock routine whether it is from the Clock Library, or loaded as a Standalone driver. These errors include:

+++ Can't find : CLOCKS.DAT
SETCLOK could not find the referenced Clock file. This error will be seen if an attempt is made to use a standalone clock driver which could not be found in the current Drive/User, via the PUBlic attribute, or along the DOS path.

+++ Error on file open
An error occurred while trying to open a standalone clock file. Insure that the file was correctly assembled and try again.

+++ Error sizing : fn.ft

The selected clock file contained erroneous or invalid sizing information. If this is reported from the CLOCKS.DAT file, reload the file from your ZSDOS backup disk and try again. If it is reported while loading a standalone clock. it is most often due to incorrect specifications of the CSEG/DSEG/Named Common areas within the Clock template. Insure that the clock specifications were followed, reassemble the driver and try again.

+++ Link Error : nn in file : fn.ft

An error occurred while linking the relocatable code from a clock driver. The "nn" reported is an indicator to the exact nature of the error. Consult the authors if you cannot resolve the error.

The final two errors are indicators that errors occurred after the clock driver has been loaded, linked, and validated. If either of these occurs, it is most often due to selection of an incorrect clock driver, problems with the hardware controlling the selected clock. or alterations to the B/P Bios code which altered the specified interface.

-- Clock Not Ticking --

The selected clock driver could not detect an active clock. This is most often the result of selecting the incorrect driver. or setting incorrect values when asked for specific addresses or values when activated.

-- Error Setting B/P Bios Clock !!

The Bios reported an error while attempting to set the B/P Bios clock. This is most often caused by errors when modifying the module [illegible].

6.19 SHOWHD - Partition Display Utility

SHOWHD is a utility which is furnished with the B/P Bios package as an aid in converting existing systems to B/P Bios without losing data, particularly on Hard Drives. It is not specific to B/P Bios and should properly execute on any CP/M 2.2-compatible system. Its purpose is to display the current Hard Drive Partition settings so that you may configure a B/P Bios in either source code, or image form (with BPCNFG) to reflect the same partitioning data.

6.19.1 Using SHOWHD

This routine is a basic utility which is normally infrequently, so frills were not added. It expects no arguments and only operates in an interactive mode. To execute it, simply enter:

```
SHOWHD
```

You will be prompted to enter a drive letter. When entered, you will be presented with a display listing the logical parameters for the drive. A sample of execution is:

```
Show Hard Drive Partition Data - 2 Nov 91

 Enter Drive Letter [A..P] : C

 Drive: C
        DPH Info                 BPCNFG Info


   Sectors/Track  = 64           (same)
   Blk Shift Fctr = 5            4k/Block
   Block Mask     = 31
   Extent Mask    = 1
   Disk Blocks-1  = 4999         20000k Total (2500 Tracks)
   Max Dirs - 1   = 1023         1024 Dir Entries
   Alloc bytes    = FFH, 00H
   Check Size     = 0
   Track Offset   = 3500         (same)
 _
```

6.19.2 SHOWHD Error Message

Only one message may be displayed from the utility. It is:

+++ Invalid Drive : d

The Drive Letter selected was not a valid drive within the Bios.

6.20 SPINUP - Hard Disk Motor Control Utility

SPINUP is a generic B/P utility to directly control the heads and motors of newer SCSI drives. It moves the heads on the specified hard drive unit to the designated shipping or park zone and may turn the drive motor off if called to Stop the unit and that feature exists in the drive. If called to Start the unit, the drive motor is turned on (if applicable) for the specified drive unit and the heads are positioned to Cylinder 0. This routine may be used as a power conservation feature where operation can be continued for periods of time from RAM or Floppy drives without need to access the hard drive unit. Attempts to access a unit which has been "spun down" with SPINUP will result in an error. This routine is essentially a generic version of PARK (see 6.17) and is furnished in source code form to demonstrate methods of interfacing to Hard Drives from Application Programs.

6.20.1 Using SPINUP

This utility was written for use primarily in battery-operated systems where power conservation is desired. Generally, only the newer SCSI drives respond to the Stop/Start Unit commands by controlling the drive motors and positioning the heads over the designated "Landing Zone". SPINUP is Command-Line driven and expects an argument consisting of a valid Unit Number for the physical Hard Drive unit (see 5.2.1, CONFIG+61). Valid Unit Numbers are "0", "1" and "2".

Stopping the unit is indicated by preceding the Unit Number with a minus sign as:

```
SPINUP -0
```

which will cause Unit 0 to park the heads and remove power from the drive motor if possible. Starting the unit is indicated by simply passing the Unit Number as an argument as:

```
SPINUP 0
```

Whether starting or stopping a Hard Drive Unit, SPINUP monitors the unit status after issuing the command and reports the status of the drive when results of the operation are received. Results are returned as; the unit is Stopped, Started, or an error has occurred (see 6.20.2 below).

Prior to using SPINUP to stop a hard drive, you must insure that accesses to any logical drive on that unit will not occur while the unit is stopped by either exchanging logical drives with BPSWAP (see 6.5) or altering the Command Processor and Dos Search Paths with the ZSDOS Utility ZPATH. For example, if your system includes a partition on the subject Hard Drive as Drive A:, you have logged onto a RAM Drive as M: and the unit is stopped, ZCPR3 and ZSDOS may attempt to find a file on drive A: which will result in a Read Error. In this example, you may either swap drive A: with M:, insuring that M: is not in either Path, or set both paths to exclude drive A:

6.20.2 SPINUP Error Messages

+++ Not B/P Bios ... aborting +++

An attempt was made to execute SPINUP under a Bios other than B/P, or modifications made to the Bios altered the locations or values needed to correctly identify the system.

+++ Can't handle this Controller Type! +++

The Controller Type within the Bios cannot handle the "Stop/Start Unit" SCSI Commands.

****** SCSI Block Length Error !**

The Bios does not support the Extended Commands necessary to turn the drive on and off using the "Stop Unit" and "Start Unit" SCSI Functions. This is most probably due to changes during an edit/assembly of the Bios which altered either the Command Descriptor Block size, or the Hard Drive function which returns the values.

****** Invalid Unit # !**

The specified unit number was not "0", "1" or "2". Only three physical Hard Drive units are recognized in B/P Bios.

6.21 TDD - SmartWatch Support Utility

TDD is a customized version of the ZSDOS utility TD. With ZSDOS2 systems, it used to display, set or update the B/P Bios clock. This latter capability only exists with the Dallas SmartWatch (DS-1216E) or JDR No-Slot-Clock in the Ampro Little Board, SB180 or YASBEC.

6.21.1 Using TDD.

TDD obtains the system Time and Date with a DOS Function 98 and displays the information on your console. Your system Must have an installed clock driver to use this utility. If the clock driver supports a set function, TDD can set the Date and Time using DOS Function 99. When setting the clock, TDD will allow you to operate in either an Interactive or Command Line driven mode.

TDD responds to the standard Help request described in Section 1.7. You may obtain a brief usage description by entering:

```
TDD //
```

You may obtain the current Date and Time from the system clock by simply entering the program name as:

```
TDD
```

If you are using a Dallas 1216E-based clock, you can set the B/P Bios clock directly by entering:

```
TDD U
```

A continuous display may be obtained which will update every second until any key is depressed by entering:

```
TDD C
```

The system clock may be set in the interactive mode by entering the program name followed by the "S" parameter as:

```
TDD S
```

You will then be asked to enter the date. The prompt will display the format in which the date will be accepted (US or European) as either:

```
Enter today's date (MM/DD/YY):          - US
or
Enter today's date (DD.MM.YY):          - European
```

Date fields (month, day, and year) may be either one or two digits in each position. Invalid entries such as an invalid day for the entered month will cause the prompt to be re-displayed for a new entry.

When a valid date has been entered, you will be prompted for the current time.

The prompt will vary depending on whether you are using a Real Time Clock, or the Relative counter substitute for a clock. The two prompts are:

```
        Enter the time (HH:MM:SS):            - Real Time Clock
        Enter the relative time (+XXXX):      - Relative Counter
```

Time is assumed to be in 24 hour format when a Real Time Clock is being used. Seconds may be omitted when setting the clock. When the relative clock is used, a '+' must prefix the count to which you wish to set the relative Counter. Counts from +0 to +9999 are permitted.

When the time entry is ended with a carriage return, the date and time will not be automatically set. A message will prompt you to press any key. At this point, the next key depression (other than shift and control) will set the clock. This procedure allows you to accurately synchronize the time with one of the many accurate time sources.

Command Line setting of the clock is initiated by entering the program name followed by the date and optional time. The date must be in the correct form (US or European) for the configured TDD. If an error is detected in a Command Line clock set, TDD switches to the interactive mode and prompts for date and time as described above.

4.6.2 Configuring TD.

TD can be configured to present the time in either the US format of month, day, year as: *Sep 18, 1988*, or the European and military style as: *18 Sep 1988*. Likewise, the set function accepts a US format of MM/DD/YY or the European DD.MM.YY. The default may be set with Al Hawley's ZCNFG utility using data contained in the TDD.CFG file on the distribution disk.

4.6.3 TD Error Messages.

Error messages for TD are simple and are mostly self-explanatory. For clarity, however, they are covered here.

SORRY! ZSDOS or ZDDOS is required to run this program!
You tried to run this with someone else's DOS. Use ZSDOS or ZDDOS. This error aborts to the Command Processor.

***** NO Clock Driver installed!!!**
You tried to read a clock which does not exist. Install a clock with SETUPZST and try again. This error aborts to the Command Processor.

***** Clock does NOT Support SET!!!**
The clock driver on your computer will not permit you to set the time with TDD. This error aborts to the Command Processor.

***** Must have B/P Bios to use!!!**

This utility only functions under B/P Bios. If you are using B/P Bios and this message appears, it is most probably due to edit/reassembly of the Bios source which altered critical values in the data structures.

***** Hardware Type Not Supported ! *****

Since TDD is integrally tied to specified hardware platforms, it will only function if the running computer is one of the types for which the correct code has been implemented. This error should not appear unless the "U" or "S" command is issued.

++ Insufficient Memory! ++

The base of available memory has been reduced below that needed for the portion of TDD which is relocated to high memory. The most common cause of this is the addition of Resident System Extensions (RSXs) which cause the top of the TPA to be reported as below 8100H.

***** Error in Data Input**

An invalid character or number was entered when trying to set the date and time. This error will cause the Interactive mode to be entered and issue a prompt to re-enter correct date/time.

***** Must be wheel to set clock!**

An attempt was made to set the clock without Wheel access. [illegible] ZSCONFIG (ZSDOS 1) or ZSCFG2 (ZSDOS2) to set a valid Wheel byte, or disable it (see 6.22 for ZSCFG2 or the ZSDOS 1 manual for ZSCONFIG). This error [illegible] to the Command Processor.

***** Must have Z180 Processor!!!**

***** Must have Z180 to Set No-Slot-Clock!!!**

****** Can't find No-Slot-Clock!**

6.22 ZSCFG2 - ZSDOS2 Configuration Utility

ZSCFG2 is a program to configure various parameters of a ZSDOS2 Operating System. It is included in this manual due to the close interaction of all parts of an operating system, particularly the Banked and Portable BIOS. ZSCFG2 operates in either an interactive (novice) or command line driven (expert) mode for maximum flexibility and ease of use. If your computer is running ZCPR3, the Z3 Environment is automatically detected and ZSCFG2 will use video attributes such as reverse video and cursor addressing to enhance the display.

As in all of our support routines, a brief on-line help message is available by entering the name followed by two slashes as:

```
ZSCFG2 //
```

This configuration tool automatically tailors itself to the ZSDOS2 system and all messages, from Help to Interactive prompts will accurately reflect the options and status for the running configuration of ZSDOS2.

6.22.1 ZSCFG2 Interactive Use.

To start ZSCFG2 in the interactive mode, simply enter the program name as:

```
ZSCFG2
```

If a valid ZCPR3 environment is located, the screen is cleared, and you will see a screen containing needed addresses from the environment, and a tabular display of the current settings within the operating ZSDOS2 system. Reverse video is used to enhance the display if available. If you are using a computer which is not equipped with ZCPR3, or cannot support direct cursor addressing, the information (less ZCPR3 addresses) is simply scrolled up the screen, one line at a time. The only content differences between the two displays is that no data on the ZCPR3 environment will be displayed. An example of a ZSDOS2 display is:

```
 ...Configuring ZSDOS Ver 2.0        Z3 Environment at  : FE00H
                                     ZCPR Path Address  : FDF4H
                                         Wheel Byte at  : FDFFH

        1 - Public Files            : YES
        2 - Pub/Path Write Enable   : NO
        3 - Read-Only Vector        : YES
        4 - Fast Fixed Disk Log     : YES
        5 - Disk Change Warning     : NO
        6 - Path w/o System Attr    : YES
        7 - DOS Search Path         : Enabled - Internal
        8 - Wheel Byte Protect      : Enabled  Addr = FDFFH
        T - Time Routine (Clock)    : F168H
        A - Stamp Last Access Time  : Disabled
        C - Stamp Create Time       : EEB2H
        M - Stamp Modify Time       : EEBCH

 Entry to Change ("X" to EXIT) : _
```

The type of Operating system and version number are first displayed, followed by any ZCPR3 Environment information needed. If no environment is located, you will see a message to that effect. In such a case, certain options will be restricted as covered later in detailed descriptions.

Interactive operation consists simply of entering the number or letter in the left of each line to select a function. If you select numbers between one and six, the option is changed from *OFF* to *ON* or vice versa, and the menu and status are again displayed. If you select any of the other items, you will be asked for more detailed information. Section 6.22.2 below contains a detailed description of all options.

6.22.2 ZSCFG2 Command Line (Expert Mode) Use.

Command line entry (or Expert Mode) provides the ability to dynamically set ZSDOS options within STARTUP scripts, ZCPR Alias files, Submit files, or directly from the your console. This permits the operating system parameters to be tailored to specific applications, and restored upon completion. For example, a submit or alias command might feature the following sequence:

Disable/Enable ZSDOS features and set addresses
...Process application programs
Restore ZSDOS features and addresses and return

Tailoring of ZSDOS in this sequence would be via a call to ZSCFG with arguments passed on the command line within the script. In this fashion, you do not have to constantly attend the computer to change DOS parameters.

Settings are passed to ZSCFG as groups of characters separated by one or more tabs, spaces or commas. Each group of characters begins with a Command character which identifies the setting to be changed. In the case of the items related to time and date, a two-character sequence is used. A "+" sign identifies the Command as a Clock, or Time Stamp-related function, and the following Command character tells which parameter of the six is to be changed. Command Identifiers for ZSDOS are:

P	- Public File Support
W	- Public/Path Write Enable
R	- Read-Only Drive Sustain
F	- Fast Hard Disk Relog
!	- Disk Change Warning
S	- Path without SYStem Attribute
>	- ZSDOS Search Path
*	- Wheel Byte Write Protect
C	- Clock Routine Address
+A	- Stamp Access Time
+C	- Stamp Create Time
+M	- Stamp Modify Time

Options which are simply On/Off toggles require no arguments. You enable them merely by entering the Command Identifier(s). To disable such options, you simply append a minus sign ("-") to the end of the Identifier. For example, a command line entry to activate the Fast Relog capability for hard disks, turn PUBlic bit capability on, and disable the Disk Change warning would be:

```
ZSCFG2 F,!-
```

Certain options require additional parameters which are handled by a secondary prompt in the interactive mode. Since no prompt can be issued in the Command Line entry mode, the added parameters are passed by appending them to the Command Identifier. An example to set the Wheel Byte Write Protect to be the same as the ZCPR3 Wheel Byte, Activate the Internal path and set the Clock address to the standard B/P Bios Clock Vector is:

```
ZSCFG2 *Z,>I,CB
```

You must remember that NO spaced or other delimiters (spaces, tabs and commas) are permitted between the Command Identifier and added arguments. An "Invalid" error will generally be the result if you forget. When entering addresses or numbers as arguments, they are ***always*** in Hexadecimal (base 16) with optional leading zeros. The algorithm used to interpret the number entered only retains up to four digits. Therefore, if you enter the sequence "0036C921045", it would be interpreted as 1045H.

The following section describes each option and what alternatives are available to optimize it for your system.

6.22.3 ZSCFG2 Option Descriptions.

The two tools which permit tailoring of a ZSDOS2 system to your specific needs. INSTAL12 and ZSCFG. both present the same interactive display. This section, therefore, is applicable to installation as well as "on the fly" customization with ZSCFG2. Each option will also include specific arguments for Command Line entry of the option. Options will be covered in their order of appearance in the INSTAL12 and ZSCFG2 interactive menus.

6.22.3.1 Public Files.

```
Interactive Prompt     : 1 - Public Files
     (toggle)
Command Line Character: P
     Enable  - P
     Disable - D-
```

This flag controls recognition of the Plu*Perfect PUBlic attribute bit (Bit 7 of the second letter in the file name). When set to YES or activated. any file having this bit set will be accessible from any user area on the disk. This means that a search for the file will locate it on the first try. regardless of which User Area is currently selected (see 2.9.4. Public Access). If set to NO. the file will be private and can only be found if the user area matches that of the file. The default setting for this option is YES. to recognize PUBlic files.

6.22.3.2 Public/Path Write Enable.

```
Interactive            : 2 - Pub/Path Write Enable
     (toggle)
Command Line Character: W
     Enable  - W
     Disable - W-
```

When set to YES or activated in Command Line mode. ZSDOS2 will permit write operations to Public files. and in the case of ZSDOS2. files located along the Path. When set to NO or disabled. attempts to write to the file will result in a "Read-Only" error. The default setting for this option is NO.

6.22.3.3 Read-Only Vector Sustain.

```
Interactive Prompt     : 3 - Read-Only Vector
     (toggle)
Command Line Character: R
     Enable  - R
     Disable - R-
```

When set to YES or activated. the normal Write Protect vector set by ZSDOS2 function call 28 will not be cleared on a warm boot as with CP/M and ZRDOS. If set to NO or disabled. the Write Protect vector will function as in CP/M and ZRDOS. The default setting for this option is YES.

6.22.3.4 Fast Fixed Disk Relog.

```
Interactive Prompt     : 4 - Fast Fixed Disk Log
     (toggle)
Command Line Character: F
     Enable  - F
     Disable - F-
```

When set to YES or enabled, the allocation bit map for a fixed drive (one in which the WACD buffer is zero) will not be rebuilt after the initial drive logon. This results in much faster operation for systems with Hard Disks and RAM disks. If set to NO or disabled, the allocation map will be rebuilt each time fixed disk drives are initially selected after a warm boot. The default setting for this option is YES.

6.22.3.5 Disk Change Warning.

```
Interactive Prompt     : 5 - Disk Change Warning
     (toggle)
Command Line Character: !
     Enable  - !
     Disable - !-
```

When set to YES or enabled, a warning will be printed whenever ZSDOS2 detects that a disk in a removable-media drive (normally floppy disk drives) has been changed. If you press any key other than control-C, ZSDOS2 will automatically log in the new disk and continue. If set to NO or disabled, no warning will be given, and the disk will be automatically logged, and the operation in progress will continue. The default setting for this option is NO, for no displayed message.

6.22.3.6 Path Without System Attribute.

```
Interactive Prompt     : 6 - Path w/o System Attr
     (toggle)
Command Line Character: S
     Enable  - S
     Disable - S-
```

When set to YES or enabled, Public files on drives along the Path will be found without the System Attribute being set (see 2.9.2, Path Directory Access mode). If this option is set to NO or disabled, Public files on drives addressed along the Path will not be found unless the System Attribute Bit (bit 7 of the second character in the filetype) is set (see 2.9.3, Path File Access mode). The default setting for this option is NO or Disabled, requiring the Public Bit Set on accessible files.

6.22.3.7 DOS Search Path.

```
Interactive Prompt      : 7 - DOS Search Path
     Options - (D)isable. (S)et addr. (I)nternal
               (Z)CPR3 (only if running ZCPR3)
Command Line Character: >
     Enable  - >addr, >I
               >Z (only if running ZCPR3)
     Disable - >-
```

When this option is selected from the Interactive mode. you will be prompted for one of the three options. Contrary to the earlier ZSDOS1 configuration, a ZCPR3 style Environment Descriptor is required, so the following prompt will always be displayed:

```
DOS Path [(D)isable, (S)et, (I)nternal, (Z)CPR3] :
```

Operating ZSCFG2 in the Command Line mode permits you to select the same options directory from the command line as summarized above. No additional characters are required for Disable. Internal or ZCPR3 path selection. If you choose the (S)et option. you will be prompted for a Hexadecimal address with:

```
Enter PATH Address :
```

If you Disable the DOS Path option, ZSDOS2 functions just as CP/M 2.2 and ZRDOS for file searches. Requests for files will access only the currently logged disk and user, modified only by the Public capability, if active. This results in the familiar requirement to install utilities such as compilers, word processors and data base management systems to tell them where to go to find their overlays.

Proper use of the DOS Path overcomes the limitation in finding program overlays and other files by simply setting the DOS Path to the drive and user area where the relevant overlays and other files are stored. The Path may be set in three ways.

The first way is to assign a fixed address using the (S)et option from the Interactive Mode, or by appending the address to the Command Character in the Command Line mode. You will be responsible for insuring that any path at that address conforms to proper ZCPR3 path definitions.

The second way to set a DOS Path is to use the three element Internal path by selecting the (I)nternal option from the Interactive Mode, or adding an "I" after the Command Character in the Command Line mode. As distributed, ZSDOS contains a single path entry of 'A0:' to direct path searches to User Area 0 on Drive A. An alternative way to activate the Internal Path is with the ZPATH.COM utility made available with ZSDOS1. ZPATH will enable you to change the default path, and define up to three drive/user search elements.

The final method of setting a DOS Path is only available if you are operating a ZCPR3 system. It is chosen by selecting the (Z)CPR3 option from the Interactive Mode, or following the Command Character with a 'Z' in the Command Line mode. This Path mode will probably see little use, but is made available for systems which need more than three elements in a path. The principal disadvantage of using the ZCPR3 path is that requests from the command prompt (e.g. A0>) may result in n-squared searches where n is the number of elements in the path. The reason is that ZCPR3 will select the first path element, and ZSDOS will sequentially search along the entire path if the file is not found, returning to ZCPR3 with a "file not found" error. ZCPR3 will then select the second element with ZSDOS again searching along the entire file. This situation does not occur once an application program is started, since the ZCPR3 Command Processor is no longer active.

The default setting for the DOS Path is "Internal".

6.22.3.8 Wheel Byte Write Protect.

```
Interactive Prompt      : 8 - Wheel Byte Protect
     Options - (D)isable, (S)et addr (Z)CPR3
Command Line Character: *
     Enable  - *addr
               *Z
     Disable - *-
```

When you select this option from the Interactive mode of ZSCFG, you will presented with an additional lines containing available choices as:

```
Wheel [(D)isable, (S)et, (Z)CPR3] :
```

Selecting the (D)isable option by entering a "D" or disabling the Wheel Byte with the "*-" parameter string in the Command Line mode will cause ZSDOS2 to assume that the Wheel byte is always ON giving the user full privileges in file control (Writes, Renames and Erasures). Entering an "S" for (S)et from the Interactive mode will allow you to enter a Hexadecimal address for a Wheel Byte. It is your responsibility to insure that the byte is protected as necessary from unintentional alteration. Setting a Wheel Byte address from the Command line simply requires appending a Hexadecimal address after the Wheel Command Character. Entering a "Z" from the Interactive mode, or a "*Z" parameter string in the Command Line mode will set the address to the address defined for the Wheel byte in the ZCPR3-compatible B/P environment. The default for this option is OFF or Disabled to assume that the user has full privileges.

6.22.3.9 Time Routine (Clock Driver).

```
Interactive Prompt     : T - Time Routine (Clock)
     Options - (D)isable, (S)et addr
Command Line Character: C
     Enable  - C addr
     Disable - C-
```

This option allows the user to enter the address of a clock driver routine conforming to ZSDOS standards, or disable an existing clock routine. When you enter a "T" at the prompt in the Interactive mode, the following appears:

Time (Clock) Routine [(D)isable, (S)et, (B)ios+4EH] :

Entering a "D" in the Interactive mode or the Command Line sequence "C-" will disable any existing clock. The primary effect of this is to cause an error return to DOS function calls 104 and 105 as well as disabling Date/Time Stamping functions. If you enter an "S" at this point in the Interactive mode, you will be further prompted for a Hexadecimal address of a clock driver. The same effect of setting a Clock Driver address is achieved in the Command Line mode by entering the "C" Command character followed by a valid Hexadecimal address beginning with a Number. Selecting "B", or entering "CB" as a command line argument, will Set the DOS Clock Driver address to the base of B/P Bios offset by 4EH which corresponds to the Jump Table entry for the B/P ZSDOS-compatible Clock driver. *Do NOT enter unknown values since unpredictable results can occur!*

6.22.3.10 Stamp Last Accessed Time.

```
Interactive Prompt     : A - Stamp Last Access Time
     Options - (D)isable, (E)nable
Command Line Character: +A
     Enable  - +A
     Disable - +A-
```

This option is only available with DateStamper type of Date/Time Stamps. For P2DOS, the function is not defined and is ignored within ZSDOS2. As stated in Section 3.4.4.2, Unless you have a definite need to retain a record of the last time files are accessed, we recommend that you disable this option to reduce unnecessary overhead. To Select the Last Access Time option, enter an "A" at the Interactive main prompt. This will display the following prompt:

Stamp Last Access Time Routine [(D)isable, (E)nable] :

If you enter a "D" at this point in the Interactive mode or disable the function with the sequence "+A-" in the Command Line mode, no times will be entered in the "Last Accessed" field in the DateStamper file. This option may be re-enabled by selecting the "E" option in the Interactive mode from this secondary prompt, or the sequence "+A" from the Command Line mode.

6.22.3.11 Stamp Create Time.

```
     Interactive Prompt     : C - Stamp Create Time
     Command Line Character: +C
          Enable  - +C addr
          Disable - +C-
```

Entry of a "C" from the main menu in the Interactive mode will allow you to enable or disable the Create Time stamping feature. A secondary prompt will be displayed as:

Stamp Create Time Routine [(D)isable, (E)nable] :

To disable the Create time, enter a "D" from the secondary prompt, or the sequence "+C-" in the Command Line mode. To enable stamping of Create Times, enter an "E" from the secondary prompt from the Interactive mode, or by entering the Argument Sequence "+C" from the Command Line mode.

6.22.3.12 Stamp Modify Time.

```
     Interactive Prompt     : C - Stamp Modify Time
          Options - (D)isable, (E)nable
     Command Line Character: +M
          Enable  - +M
          Disable - +M-
```

The time of last Modification of a file is probably the most valuable of the times offered in a ZSDOS system. As such, you will probably never have a need to disable this feature. Should the need arise, however, enter an "M" at the Interactive main prompt. You will then be presented with:

Stamp Modify Time Routine [(D)isable, (E)nable] :

If you enter a "D" at this point in the Interactive mode or disable the function with the sequence "+M-" in the Command Line mode, no times will be stored in the "Modify" field of any active Time Stamp activity.

This option may be re-enabled by selecting the "E" option in the Interactive mode from this secondary prompt, or the sequence "+M" from the Command Line mode.

6.22.4 ZSCFG2 Error Messages.

Only two error messages exist in ZSCFG2. For the most part, any error you see will deal with invalid parameters or entry mistakes. The two error messages are:

-- Invalid --

An invalid address or character was entered in a parameter.

*** ERROR: DOS is not ZSDOS2!

An attempt was made to run ZSCFG2 on an Operating system which was not ZSDOS2. This program cannot function under any other operating system.

6.23 ZXD - File Lister Utility for ZSDOS2

ZXD Version 1.60 is a modification of an earlier version released with our ZSDOS 1 package. It is the ZSDOS Extended Directory listing program derived from the ZCPR3 tool XD III written by Richard Conn and now modified to properly return disk sizing information from a banked ZSDOS 2 Operating System. Many additional capabilities were added over the original XD III, not the least of which is the ability to display time stamps for each file in a variety of formats. ZXD can display file Dates and times from DateStamper, P2DOS, and Plu*Perfect Systems' DosDisk stamp methods. In ZCPR3 systems, the Wheel byte is used to disable some functions as a security precaution in remote access systems.

6.23.1 Using ZXD.

ZXD is activated by entering its name at the command prompt, and may be followed by optional drive and user specifications to obtain the directory of another drive or user area. It may also be followed by various parameters which alter the format and/or content of the display. If options are listed without being preceded with a File Specification (drive, user, file name), then they must be preceded by the standard option character, a slash. A help message may be obtained in the standard manner by entering:

```
ZXD //
```

ZXD accepts directions in a natural form popularized with the ZCPR series of Command Processor replacements. Using the conventions described in Section 1.2, the syntax is summarized by:

```
ZXD [dir:][afn] [/][options]
```

If ZXD is called with no arguments, a display of only those files satisfying built-in default conditions will be displayed. Normally these defaults select only non-system files in the current drive and user. The default selections may be modified by option parameters detailed below. If option parameters are desired without drive, user or file specifications, then the options must be prefixed with a slash. The slash is optional if any redirection or file specifications are entered.

6.23.2 ZXD Options.

Option parameters, consisting of one or two characters, allow you to obtain selected information from files on a disk, or to tailor the display to your particular needs. Each of these options is also reflected as a permanent default. After deciding which parameters you use most by using the command line options, we recommend configuring ZXD to reflect those parameters as defaults. The results will be the requirement to enter fewer keystrokes, and consequently faster operation when a directory scan is required.

The option characters are described in alphabetical order in the following sections.

6.23.2.1 Select Files by Attribute.

In order to avoid cluttering a directory display with unwanted file names. ZXD features a flag which controls addition of those files marked with the SYStem Attribute Bit. The **A** Option controls this feature. It requires a second character of **S**, **N**, or **A**. Control offered by these characters is:

- **S** - Include Only Files marked with the SYStem Attribute
- **N** - Include Only Files Not marked with the SYStem Attribute (this is the default condition)
- **A** - Include All Files

Since listing of all Non-SYStem files is the default condition, you will probably not use the **N** option very often. The **A** option, on the other hand, offers a simple way of viewing All files within the current directory, including SYStem files which are normally invisible due to the Attribute bit.

In a ZCPR3 system where Wheel access has not been granted (Wheel byte is Off), this option is forced to Non-SYStem files only and the **A** option character is not permitted.

6.23.2.2 Date Display Format.

The Dates for a ZXD display may be displayed in either US form of MM/DD/YY or European form of DD.MM.YY. You may override the default form with the **D** option. Here is an example of the two types of date displays:

US Form:

```
ZXD  Ver 1.66        3 Apr 1993   15:43:17
Filename.Typ  Size      Modified       Filename.Typ  Size      Modified
-------- ---  ----     ---------       -------- ---  ----     ---------
INITDIR .COM    4k  07:01-09/17/88     ZPATH   .COM    4k  07:50-09/17/88
ZXD     .COM    8k  08:01-09/17/88
  C2: -- 3 Files Using 16K (324K Free)
```

European Form:

```
ZXD  Ver 1.66      3 Apr 1993  15:43:11
Filename.Typ  Size     Modified       Filename.Typ  Size     Modified
-------- ---  ----    ---------       -------- ---  ----     --------
INITDIR .COM    4k  07:01-17.09.88    ZPATH   .COM    4k  07:50-17.09.88
ZXD     .COM    8k  08:01-17.09.88
  C2: -- 3 Files Using 16K (324K Free)
```

6.23.2.3 Disable Date (NoDate) Display.

While the display of date and time information is the default mode of ZXD, this may be disabled with the N option to display more file names on a screen.

6.23.2.4 Output Control Option.

The O option controls ZXD's printer or screen output, and requires a second character which adds additional control to output formats. The second characters recognized are:

F - Send a Form Feed character at the end of the list
H - Toggle Horizontal/Vertical display of sorted listing

6.23.2.5 Output to Printer.

Option P controls output to the printer. When this option is given the sorted directory listing is sent to both the console screen and the printer. This option is disabled and not available in a ZCPR3 system where Wheel access has not been granted (Wheel byte is Off).

6.23.2.6 Sort by Name or Type.

The default sort condition for ZXD is to first sort by File Name, then by File Type within matching Names. Option S reverses the sequence.

6.23.2.7 Primary DateStamp.

ZXD features an algorithm which will attempt to find one of several types of Date/Time Stamps for each file. The default conditions tell ZXD to first attempt to locate DateStamper type of Stamps. If that fails, a search is made for DosDisk stamps from MS/PC-DOS disks, and finally to check for P2DOS type stamps. The T option causes the DateStamper checks to be bypassed, thereby speeding response if DateStamper type stamping is never used.

6.23.2.8 All User Areas.

The distribution version of ZXD will only search a single User area, either the currently logged or the explicitly stated area, for files. The U option will locate files in *all* user areas on the disk. Combining the U with the AA options will list all files in all user areas, both system and non-system, on a disk. This option is disabled and not available in a ZCPR3 system where Wheel access has not been granted (Wheel byte is Off).

6.23.2.9 Wide Display.

ZXD only displays the "Last Modified" Date/Time Stamp. This may be reversed by appending the **W** option to the Command Line, which generates a ***Wide*** of all available Stamps. Only DateStamper has provisions for all three stamp categories: P2DOS contains only ***Created*** and ***Modified*** stamps, while the single MS/PC-DOS stamp accessed through DosDisk best corresponds to "Modified". A display created with this option is:

```
Filename.Typ  Size     Created         Last Access        Modified
------------  ----     -------         ---- -------       --------
BU16     .COm   8k  17:26-06/12/88  08:42-08/21/88  17:26-06/12/88
COPY     .COM   8k  15:06-09/17/88                  15:06-09/17/88
ZPATH    .COM   4k  07:50-09/17/88  15:02-09/17/88  07:50-09/17/88
ZXD      .COM   8k  08:00-09/17/88                  08:01-09/17/88
```

6.23.3 Customizing ZXD.

The configuration utility ZCNFG.COM is used to alter the default settings in ZXD. Settings and text prompts for ZXD are contained in the file ZXD.CFG which must be accessible to ZCNFG for any configuration change. All options are simple ON/OFF, or reversible settings and correspond to the options discussed in Section 6.23.2 above. See Section 4.5 of the ZSDOS 1.0 User Manual or ZCNFG documentation on the Ladera Z-Node for details on using ZCNFG.

7.0 ZSDOS Version 2.

Version 2 of ZSDOS is currently in a developmental phase. The version provided with this package is preliminary and should not be considered a final work. Be sure you back up any files which you don't mind sacrificing, and please let us know in as much detail as possible any problems you experience.

In addition to the ZSDOS Version call (Function 48) returning 20H signifying ZSDOS2, three new Operating System functions have been added. They are:

Function 46	Return Disk Free Space
Enter: C = 46 (function #) E = Drive # (A=0..P=15)	Exit: A = 0 if Ok, <>0 if Error Disk Free Space in Kilobytes is placed in DMA+0 (LSB) thru DMA+3 (MSB).

This function returns Disk Free Space from fully-banked systems where the ALV buffers are not directly accessible by applications programs. It *MUST* be used to reliably determine free space since there is no way for programs to ascertain which System Bank (if more than one) contains the Allocation Bit Map. For most reasonably-sized systems, only the lower two or three bytes will be used, but four bytes are allocated to accommodate a maximally-sized system.

Function 49	Return Environment Descriptor Address
Enter: C = 49 (function number)	Exit: HL = Address of Env Desc.

This function returns the address of a ZCPR 3.4 "type" Environment Descriptor needed in B/P Bios systems. Rather than rely on the Command Processor inserting the ENV address into application programs upon execution, this function may be used to reliably acquire the ENV address at any time.

Function 152	Parse File Name
Enter: C = 152 (function number) DE = Pointer to dest FCB DMA --> start of parse string	Exit: A = Number of "?" in fn.ft DE = points to delimiter FCB+15 will be 0 if parse Ok, 0FFH if errors occurred.

This function may be used to replace Z3LIB library routines in a more robust manner and produce consequently smaller applications programs. It is fully compliant with ZCPR 3.4 parse specifications.

8.0 ZCPR Version 4.

Z40.ZRL is a consolidation of ZCPR34 and many of the RCP features commonly in use, modified by the need to bank as much of the Command Processor as possible. When Z40 is used in a Fully-Banked system, you may not need much of, or any Resident Command Processor with your system. Z40 relys on ZSDOS2 and will *NOT* work without it since the Command Line Parser and disk free space calculations have been removed in favor of ZSDOS2 services. Additionally, the prompt line displays the time and will only function correctly if he ZSDOS2 clock is enabled. Comments on how these new System components work would be appreciated.

More complete documentation is provided in the Z40.HLP files included with the distribution diskettes, and a list of active functions is available with the H command at the prompt. To read the On-line help files, use HELP.COM available for downloading from any Z-Node.

Application Programs. In contrast to *utility programs* (see), application programs or *applications* are larger programs such as word processors which function interactively with the user.

BDOS. Basic Disk Operating System. The machine-independent, but usually processor-dependent, program which controls the interface between application programs and the machine-dependent hardware devices such as printers, disk drives, clocks, etc. It also establishes the concept of files on media and controls the opening, reading, writing, and closing of such constructs.

BGii. BackGrounder ii from Plu*Perfect Systems, a windowing task-switching system for CP/M users with hard or RAM disks.

BIOS. Basic Input/Output System. Machine-dependent routines which perform actual peripheral device control such as sending and receiving characters to the console, reading and writing to disk drives, etc.

Bit. Binary digit. An element which can have only a single on or off state.

Bit Map. An array of bits used to represent or *map* large arrays of binary information in a compact form.

Boot. The term used for the starting sequence of a computer. Generally applies to starting from a 'cold,' or power-off state, and includes the loading of Operating System, and configuration steps.

Byte. A grouping of eight bits.

CPR. Command Processor Replacement. Replaces CCP (see below). Example: ZCPR

CCP. Console Command Processor. The portion of the operating system that interprets user's commands and either executes them directly or loads application programs from disk for execution. The CCP may be overwritten by applications, and is reloaded by the "Warm Boot" function of the BIOS.

Checksum. An value which arithmetically summarizes the contents of a series of memory locations, and used to check the current contents for errors.

Clock Driver. A software link between a Non-banked ZSDOS and the clock on your system. The clock driver allows ZSDOS and its utilities to read the clock which is normally inherent in the B/P Bios.

Command Script. Sometimes called simply *scripts*, command scripts allow you to create a single command which issues other commands to perform a unique set of actions. CP/M *submit files* are one kind of command script familiar to all CP/M users. ZCPR also offers more sophisticated types of scripts such as *aliases* and *command files* (e.g., *ALIAS.CMD*).

DateStamper. A software package developed by Plu*Perfect Systems to allow time and date stamping of files. The Boot System uses an external module in the file LDDS.COM to implement DateStamper, while ZSDOS2 automatically supports this stamping method. DateStamper is unique among file stampers for microcomputers for two reasons: first, it maintains all file stamps within a file; second, it maintains stamps for create, access, and modify time/date for each file.

DDT. Dynamic Debugging Tool. A utility distributed with CP/M 2.2 which can display, disassemble, or alter disk files or areas of memory using opcodes or hexadecimal values.

DOS. Disk Operating System. Often used term for the BDOS, but generally refers to the aggregate of CCP, BDOS and BIOS.

DosDisk. A software package from Plu*Perfect Systems which allows users of CP/M and compatible computers to write and read files directly to and from standard 5-1/4" 40-track Double-Sided, Double-Density MS-DOS format diskettes. This is the standard '360k' disk format used in IBM-PC compatible computers.

FCB. File Control Block. A standard memory structure used by CP/M and compatible operating systems to regulate disk file operations.

File Attributes. Also known as *file attributes*, reserved bits stored along with file names in disk directories which control how the files are accessed.

Hexadecimal. A base-16 numbering system consisting of the numbers 0-9 and letters A-F. Often used to represent bytes as two digits (00 to FF). Hexadecimal numbers are usually represented by suffixing the number with an 'H' as in "01H".

IOBYTE. Input/Output Byte. A reserved byte at location 3 which is used by some CP/M BIOSes to redirect input and output between devices such as terminals and printers.

K. Usually refers to Kilobyte or 1024 (2 10th power) bytes.

P2D. P2Dos Datestamps. An alternative form of file stamping used in HAJ Ten Brugge's P2DOS. P2D stamps are compatible with CP/M Plus time and date stamps. This format is supported in a B/P Boot system with the LDP2D.COM Stamp module, and automatically in ZSDOS2.

RAM. Random Access Memory. As opposed to Read Only Memory (ROM) the area of a computer's memory which may be both read from and written to.

RSX. Resident System Extension. A program module complying with a standard developed by Plu*Perfect Systems for extending the functionality of a CP/M 2.2 compatible Operating System. The module must be loaded at the top of the Transient Program Area, and below the Console Command Processor.

System Prompt. The familiar A> prompt which appears soon after CP/M computers are started up.

TPA. Transient Program Area. That addressable memory space from the lowest available address to the highest available address. Usually this extends from 100H to the base of the BDOS (assuming that the Command Processor is overwritten), or the base of the lowest RSX.

Utility Programs. In contrast to ***application programs*** (see), utility programs or ***utilities*** are shorter programs, such as directory programs, which accept a single command from the user.

Wheel Byte. Taking its name from the colloquial "Big Wheel," the Wheel byte controls security under ZCPR and ZSDOS. When the byte is set to a non-zero value, the user has "Wheel status" and may execute commands unavailable to other users.

Word. In the computer context, a fixed number of bytes. For 8- bit microcomputers, a word is usually two bytes, or 16 bits.

Z-System. An operating system which completely replaces CP/M by substituting ZCPR for Digital Research's command processor and ZSDOS for Digital Research's disk operating system. ZCPR and ZSDOS complement one another in several ways to enhance performance.

ZCPR. Z80 Command Processor Replacement. Originally developed as a group effort of the Special Interest Group for Microcomputers (SIG/M), but refined by Richard Conn to ZCPR version 3.0 and Jay Sage to versions 3.3 and 3.4.

ZRL. A form of Relocatable file image using specified 'Named common' bases. For ZSDOS, files of this type are MicroSoft-compatible REL files using only the Common Relative segment " BIOS ".

This manual is not intended as a complete reference to Z80 assembly language, CP/M, or ZCPR operating systems. Since many books have been written on these topics we include here only those we have found most useful. The sources in each category are listed in order of difficulty.

1. Information on CP/M

Waite, Mitchell and Robert Lafore. *Soul of CP/M: How to Use the Hidden Power of Your CP/M System*. Indianapolis: Howard W. Sams & Co., 1983. Well written basic introduction to the principles and design of CP/M and the use of 8080 assembly language. Many programming examples and several useful appendices included. Illustrated, appendices, index, 391 pp.

Johnson-Laird, Andy. *The Programmer's CP/M Handbook*. Berkeley: Osborne/Mcgraw-Hill, 1983. Highly detailed, complete description of the structure of CP/M for programmers. Instructions on patching, designing and debugging a new CP/M system. Complete example BIOS source included. Illustrated, appendices, index, 501 pp.

Digital Research. *CP/M Operating System Manual*. Pacific Grove, California: Digital Research, 1982. Original documentation by Digital Research, developers of CP/M. Concise and authoritative, and technically written, this guide is mainly of use as a reference guide for those who already understand CP/M. Includes coverage of the original CP/M utilities and a skeletal example BIOS. Illustrated, appendices, index, 250 pp.

2. Z80 Assembly Language

Mostek Corporation. *Programming Manual for Z80 Microcomputer*. Publication No. MK78515. MOSTEK Corp., 1977. The "Bible" of Z80 assembly language programming with detailed descriptions and examples of each opcode.

Hitachi America, Ltd. *HD64180 8-Bit High Integration CMOS Microprocessor Data Book*. Publication #U77. Hitachi America, Ltd., 1985. The reference book for the 64180. No programming examples, but very detailed listings of all instructions.

Zaks, Rodnay. *Programming the Z80*. Berkeley: SYBEX, Inc., 1982. Highly detailed information about every aspect of the Z80 microprocessor and Z80 assembly language. Covered are basic programming concepts, Z80 hardware organization, programming, addressing, and I/O techniques. Programming examples are included, and over 200 pages of information are devoted to the Z80 instruction set itself. Illustrated, appendices, index, 624 pp.

3. Information on ZCPR, ZSDOS, the Z-System and Accessories

Morgen, Bruce and Jay Sage. The Z-System User's Guide. An introduction to Z-System for the less technical user.

Sage, Jay. The ZCPR 3.3 User's Guide. Manual for ZCPR version 3.3, including many examples of how the features of Z-System can be put to work. Extended command processing and security features are highlighted.

Conn, Richard. ZCPR3: The Manual. New York: New York Zoetrope, 1985. The "bible" of ZCPR3, written by ZCPR's original author. Much information is now out of date, but still very useful. Illustrated, index, 351 pp.

Conn, Richard. ZCPR3: The Libraries. Manual for the SYSLIB, Z3LIB, and VLIB assembly language libraries, which allow advanced ZCPR3 programmers to create their own Z-System programs.

Mitchell, Bridger. Backgrounder ii. Idyllwild, CA, 1986. (Available from Plu*Perfect Systems and others). Manual for Backgrounder ii includes many advanced techniques and a description of the Plu*Perfect RSX standard.

Bower, Harold; Cotrill, Cameron and Wilson, Carson. ZSDOS 1.0 User's Guide. 1988. Detailed User instructions for the Non-banked version of ZSDOS, along with detailed references. This is a must if you do not move up to the banked ZSDOS2.

Bower, Harold; Cotrill, Cameron and Wilson, Carson. Programmer's Manual for ZSDOS 1.1. 1990. Details data areas, interfaces, and programming examples for writing programs for systems running the ZSDOS system.

Number	Fcn Name	Input Parameters	Returned Values	Uses
0	CBOOT	None	None	All Regs
1	WBOOT	None	None	All Regs
2	CONST	None	A=FFH if Ready A=00H if No Char	AF
3	CONIN	None	A=Char from CON: (masked per ICFG)	AF
4	CONOUT	C=Char to send (masked per ICFG)	None	AF
5	LIST	C=Char to send (masked per ICFG)	None	AF
6	AUXOUT	C=Char to Send (masked per ICFG)	None	AF
7	AUXIN	None	A=Char from AUX: (masked per ICFG)	AF
8	HOME	None	< Status Code >	All Regs
9	SELDSK	C=Drive (0=A..15=P)	HL=Disk Param Hdr Addr HL=0 if No Drive	All Regs
10	SETTRK	BC=Track Number	None	None
11	SETSEC	BC=Sector Number	None	None
12	SETDMA	BC=DMA Address	None	None
13	READ	None	< Status Code >	All Regs
14	WRITE	C=Write Type (0=Unallocated, 1=Directory, Force Write)	< Status Code >	All Regs
15	LISTST	None	A=FFH if PTR Ready A=00H if PTR Not Ready	AF
16	SECTRN	BC=Logical Sector #	HL=Physical Sector # DE=Translation Tbl Addr	All Regs

------ <<< End of CP/M 2.2 Vectors >>> --

Number	Fcn Name	Input Parameters	Returned Values	Uses
17	CONOST	None	A=0FFH if CON: Ready A=00H if CON: Busy	AF
18	AUXIST	None	A=0FFH if AUXin Ready A=00H if No Char	AF
19	AUXOST	None	A=0FFH if AUXout Ready A=00H if AUXout Busy	AF
20	DEVTBL	None	HL--> Char IO Table	HL
21	DEVINI	None	None	All Regs
22	DRVTBL	None	HL--> DPH Table if Ok (0 entries for No Drv)	HL
23	<Reserved for MULTIO>			
24	FLUSH	None	< Status Code >	All Regs
25	MOVE	HL=Source Addr DE=Dest Addr BC=Length to move	None	All Regs
26	TIME	C=0(Read), C=1(Set) DE--> 6-byte Time	A=1 if Ok, A=0 if Errs E=Orig 6th buffer byte D=1/10th Secs (Read)	All Regs
27	SELMEM	A=Bank Number	None	None
28	SETBNK	A=Bank Number	None	None

29	XMOVE	C=Source Bank #
		B=Dest Bank #
			None	None

----- <<< End of CP/M 3 "Type" Vectors >>> -----

30	RETBIO	None	A=Bios Version # BC--> Bios Jump Table DE--> Config Data Area HL--> Device Config Table	A,BC,DE,HL

31	DIRDIO	<-- See details below for Direct Device IO -->
		B=Driver Type
		C=Function #
			<see table>	<see table>

32	STFARC	A=Bank Number	None	None
33	FRJP	HL=Dest Address	??	??
34	FRCLR	HL=Return Address	??	None
35	FRGETB	HL--> Byte to Get C=Bank Number	A=Byte at (C:HL)	AF
36	FRGETW	HL--> Word to Get C=Bank Number	DE=Word at (C:HL)	F,DE

37	FRPUTB	HL--> Byte Dest
		C=Bank Number
		A=Byte to Put
			None
			(A saved to (C:HL))
				F

38	FRPUTW	HL--> Word Dest
		C=Bank Number
		DE=Word to Put
			None
			(DE saved to (C:HL))
				F

39	RETMEM	None	A=Current Bank Number	AF

================= FLOPPY DISK SUBFUNCTIONS (Function 31) =================

0	STMODE	A=0 (Double Dens)
		0FFH (Single Dens)
		C=0 (Fubfcn #)
		B=1 (Floppy)
			None	AF

1	STSIZE	A=0 (Normal Speed)
		0FFH (Hi capable)
		D=0 (Motor On Cont)
		0FFH (Motor Contr)
			None	AF

		E=0 (Hard), 1 (8")
		2 (5.25"), 3 (3.5")
		C=1 (Subfcn #)
		B=1 (Floppy)
			None	AF

2	STHDRV	A=Unit (B0,1)
		Head (D2)
		C=2 (Subfcn #)
		B=1 (Floppy)
			None	AF

3	STSECT	A=Phys Track #	None	AF

		D=0 (128 Sctrs..
		3=1024 Sctrs)
		E=Last Sector #
		C=3 (Subfcn #)
		B=1 (Floppy)
			None	AF

4	SPEC	A=Step rate in mS	None	AF
		(B7=1 for 8" Drv)		
		D=Head Unload in mS		
		E=Head Load in mS		
		C=4 (Subfcn #)		
		B=1 (Floppy)		
5	RECAL	C=5 (Subfcn #)	< Status Code >	AF
		B=1 (Floppy)		
6	SEEK	A=Desired Trk #		
		D=OFFH Verify,	< Status Code >	AF
		E=0 (Single-step)		
		<>0 (Double-step)		
		C=6 (Subfcn #)		
		B=1 (Floppy)		
7	SREAD	HL--> Read Buffer	< Status Code >	AF,HL
		C=7 (Subfcn #)		
		B=1 (Floppy)		
8	SWRITE	HL--> Write Buffer	< Status Code >	AF,HL
		C=8 (Subfcn #)		
		B=1 (Floppy)		
9	READID	C=2 (Subfcn #)	< Status Code >	AF
		B=1 (Floppy)		
10	RETDST	C=10 (Subfcn #)	A=Status Byte	AF,BC,HL
		B=1 (Floppy)	BC=Controller Type	
			HL--> Status Byte	
11	FMTTRK	HL--> Format Data	< Status Code >	All Regs
		D=Sctrs/Trk		
		E=Gap 3 Byte Count		
		C=11 (Subfcn #)		
		B=1 (Floppy)		

================== HARD DISK SUBFUNCTIONS (Function 31) ==================

0	HDVALS	DE--> 512 byte Buff	A=# Bytes in CDB	AF,HL
		C=0 (Subfcn #)		
		B=2 (Hard)		
1	HDSLCT	A=Device Byte	A=Physical Device Bit	AF
		C=1 (Subfcn #)		
		B=2 (Hard)		
2	DOSCSI	DE--> Comnd Desc Blk	H=Message Byte	All Regs
		A=0 (No Write Data)	L=Status Byte	
		C=2 (Subfcn #)	A=Masked Status Byte	
		B=2 (Hard)		

==

Status Code: A = 0, Zero Set (Z) if Operation successfully performed
A <> 0, Zero Clear (NZ) if Errors occured in Operation

Number	Function name	Input Parameters	Returned Values
0	Boot	None	None
1	Console Input	None	A=Character
2	Console Output	E=Character	A=00H
3	Reader Input	None	A=Character
4	Punch Output	E=Character	A=00H
5	List Output	E=Character	A=00H
6	Direct Console I/O	E=0FFH (In)	A=Input Character
		E=0FEH (In)	A=Console Status
		E=0FDH (In)	A=Input Character
		E=00H..0FCH (Out)	A=00H
7	Get I/O Byte	None	A=I/O Byte (0003H)
8	Set I/O Byte	E=I/O Byte	A=00H
9	Print String	DE=Address String	A=00H
10	Read Console Buffer	DE=Address Buffer	A=00H
11	Get Console Status	None	A=00H = No character
			A=01H = Char. present
12	Get Version Number	None	A=Version Number (22H)
13	Reset Disk System	None	A=00H No $*.* on A
			A=FFH $*.* on A
14	Select Disk	E=Disk Number	A=00H No $*.* File
			A=FFH $*.* File
15	Open File	DE=Address of FCB	A=Directory Code
16	Close File	DE=Address of FCB	A=Directory Code
17	Search for First	DE=Address of FCB	A=Directory Code
18	Search for Next	DE=Address of FCB	A=Directory Code
19	Delete File	DE=Address of FCB	A=Error Code
20	Read Sequential	DE=Address of FCB	A=Read/Write Code
21	Write Sequential	DE=Address of FCB	A=Read/Write Code
22	Make File	DE=Address of FCB	A=Directory Code
23	Rename File	DE=Address of FCB	A=Error Code
24	Get Login Vector	None	HL=Login Vector
25	Get Current Disk	None	A=Current Disk
26	Set DMA Address	DE=DMA Address	A=00H
27	Get Alloc. Address	None	HL=Addr Alloc Vector
28	Write Protect Disk	None	A=00H
29	Get R/O Vector	None	HL=R/O Vector
30	Set File Attributes	DE=Address FCB	A=Error Code
31	Get DPB Address	None	HL=Address of DPB
32	Set/Get User Code	E=FFH (Get)	A=User Number
		E=User Number (Set)	A=00H
33	Read Random	DE=Address of FCB	A=Read/Write Code
34	Write Random	DE=Address of FCB	A=Read/Write Code
35	Compute File Size	DE=Address of FCB	A=Error Code
36	Set Random Record	DE=Address of FCB	A=00H
37	Reset Mult Drive	DE=Mask	A=00H
38	*Not Implemented*		
39	Get fixed disk vector	None	HL=Fixed Disk Vector
40	Write random, 00 fill	DE=Addr of FCB	A=Read/Write Code
41-44	*Not Implemented*		

45	Set error mode	E=FFH (Get)	A=00H
		E=FEH (Get Err/Disp)	A=00H
		E=01H (Set ZSDOS)	A=00H
		E=00H (Set CP/M)	A=00H
46	Return Free Space	E=Disk Number	A=Error Code
			Space in DMA+0..DMA+3
47	Get DMA address	None	HL=Current DMA Address
48	Get DOS & version	None	H=DOS type: "S"=ZSDOS
			"D"=ZDDOS
			L=BCD Version Number
49	Return ENV Address	None	HL=Env. Descriptor Addr
50-97	*Not Implemented*		
98	Get time	DE=Address to Put Time	A=Time/Date Code
99	Set time	DE=Address of Time	A=Time/Date Code
100	Get flags	None	HL=Flags
101	Set flags	DE=Flags	None
102	Get file stamp	DE=Addr of FCB	A=Time/Date Code,
			Stamp in DMA Buffer
103	Set file stamp	DE=FCB Address,	A=Time/Date Code
		Stamp in DMA Buffer	
104-151	*Not Implemented*		
152	Parse FileSpec	DE=FCB Address,	A=No. of "?"s in Fn.Ft
		String in DMA Buffer	DE=Addr of delimit chr
			FCB+15=0 if Parse Ok
			0FFH if Error(s)
153-255	*Not Implemented*		

Directory Codes: A=00H, 01H, 02H, 03H if No Error
A=FFH if Error

Error Codes: A=00H if No Error A=0FFH if error

Time/Date Codes: A=01H if No error A=0FFH if error

Read/Write Codes: A=00H if No error
A=01H Read => End of File
Write =>Directory Full
A=02H Disk Full
A=03H Close Error in Random Record Read/Write
A=04H Read Empty Record during Random Record Read
A=05H Directory Full during Random Record Write
A=06H Record too big during Random Record Read/Write

Extended Error codes in Return Error Mode:
A=0FFH Extended Error Flag
H=01H Disk I/O Error (Bad Sector)
H=02H Read Only Disk
H=03H Write Protected File
H=04H Invalid Drive (Select)

The universal stamp and time formats used by ZSDOS are based on packed BCD digits. It was decided that these were the easiest format for Z80 applications programs to work with, and were compatible with most real time clocks. The format for the stamps and for the clock functions are identical to the Plu*Perfect DateStamper's formats for these functions.

Some file stamping formats (for example CP/M Plus type) do not store all the information present in the universal format to disk. In the case of CP/M Plus type stamps, there is no provision for stamping the Last Access time. The ZSDOS interface routines fill unimplemented fields in the stamp with 0 when the Get Stamp function is used, and ignore the contents of the unused fields when the Put Stamp function is used.

Depending on the stamping method selected, the format of the stamps on the disk may differ from the universal format. These differences are effectively hidden from users by ZSDOS and the Stamp routines so long as ZSDOS's functions are used to get or manipulate the stamps.

Time format (6 bytes packed BCD):

```
TIME+0 - last 2 digits of year
         (prefix 19 assumed for 78 to 99, else 20 assumed)
TIME+1 - month  [1..12]
TIME+2 - day    [1..31]
TIME+3 - hour   [0..23]
TIME+4 - minute [0..59]
TIME+5 - second [0..59]
```

File Stamp format (15 bytes packed BCD):

```
DMA+0  - Create field (first 5 bytes of time format)
DMA+5  - Access field (first 5 bytes of time format)
DMA+10 - Modify field (first 5 bytes of time format)
```

TABLE OF CONTENTS

1.0 Introduction.

The Banked and Portable (B/P) Basic I/O System (BIOS) is an effort to standardize many of the logical to physical mapping mechanisms on Microcomputers running Z-Systems with ZSDOS. In expanding the capabilities of such systems, it became apparent that standard BIOSes do not contain the functionality necessary, adequate standardization in extended BIOS calls, nor an internal structure to fully support external determination of system parameters. B/P Bios provides a method of achieving these goals, while also possessing the flexibility to operate on a wide range of hardware systems with a much smaller level of systems programming than previously required.

1.1 About This Manual.

Documentation on B/P Bios consists of this manual plus the latest addendum on the distribution disk in the file **README.2ND**. This manual is divided into the following sections:

- o *The Features of B/P Bios* summarizes the significant features of B/P Bios in general, highlighting advantages and the few limitations in the system.
- o *Tailoring B/P Bios* contains details on altering the many options to generate a customized .REL file tailored to your system.
- o *Installing a B/P Bios* details the installation of B/P Bios in both *Unbanked* and *Banked* configurations in a "how to" fashion.
- o *Programming for B/P Bios* describes the interfaces, data structures and recommended programming practices to insure the maximum benefit and performance from systems with B/P Bios.
- o *The B/P Bios Utilities* describes the purpose, operation, and customization of all supplied B/P Bios utilities and support routines.
- o *Appendices* which summarize various technical information.
- o A *glossary* defining many technical terms used in this Manual.
- o An *index* of key words and phrases used in this Manual.

For those not interested in the technical details, or who want to bring the system up with a pre-configured version as quickly as possible, Section 4, *Installing a B/P Bios*, will lead you through the installation steps needed to perform the final tailoring to your specific computer. Other chapters cover details of the individual software modules comprising the B/P Bios, and specifics on the utilities provided to ease you use of this product.

1.2 Notational Conventions

Various shorthand terms and notations are used throughout this manual. Terms are listed in the Glossary at the end of this manual.

Though the symbols seem cryptic at first, they are a consistent way of briefly summarizing program syntax. Once you learn to read them you can tell at a glance how to enter even the most complicated commands.

Several *special symbols* are used in program syntax descriptions. By convention, square brackets ([]) indicate optional command line items. You may or may not include items shown between brackets in your command, but if you do not, programs usually substitute a default value of their own. If items between brackets are used in a command, all other items between the brackets must also be used, unless these items are themselves bracketed.

All of the support utilities developed to support the B/P Bios system contain built-in help screens which use the above conventions to display helpful syntax summaries. Help is always invoked by following the command with two slashes (//). So for example,

```
ZXD //
```

invokes help for ZXD, the ZSDOS extended directory program. Interactive ZSDOS programs such as BPCNFG2 also contain more detailed help messages which appear as a session progresses.

Many utilities may be invoked from the command line with *options* which command the programs to behave in slightly different ways. By convention, options are given after other command parameters. For example, the P option in the command

```
ZXD *.* P
```

causes the ZXD directory utility to list all files (*.*) and send its output to the printer (P). For convenience, a single slash character (/) can often be used in place of leading parameters to signify that the rest of the command line consists of option characters. Therefore, the command

```
ZXD /P
```

is identical in meaning to the previous example (see 6.23 for more on ZXD).

1.3 What is B/P Bios?

B/P Bios is a set of software subroutines which directly control the chips and other hardware in your computer and present a standard software interface to the Operating System such as our ZSDOS/ZDDOS, Echelon's ZRDOS, or even Digital Research's CP/M 2.2. These routines comply with the CP/M 2.2 standards for a Basic IO System (BIOS) with many extensions; some based on CP/M 3.x (aka CP/M

Plus), and others developed to provide necessary capabilities of modern software. When properly coded, the modules comprising a B/P Bios perform with all the standard support utilities, nearly all Z-System utilities, and most application programs without alteration.

The ability to operate Banked, Non-banked and Boot System versions of the Bios with a single suite of software, across a number of different hardware machines, plus the maximization of Transient Program Area for application programs in banked systems are features which are offered by no other system of which we are aware.

1.3 The History of B/P Bios.

Our earlier work developing ZSDOS convinced us that we needed to attack the machine-dependent software in Z80-compatible computers and develop some standard enhancements in order to exercise the full potential of our machines. This premise is even more true today with large Hard Disks (over 100 Megabytes) being very common, needs for large RAM Drives, and an ever shrinking Transient Program Area. Attempts to gain flexibility with normal operating systems were constrained by the 64k addressable memory range in Z80-compatible systems, and forced frequent operating system changes exemplified by NZCOM and NZBLITZ where different operating configurations could be quickly changed to accommodate application program needs.

In the mid to late 1980's, several efforts had been made to bank portions of CP/M 2.2 "type" systems. XBIOS was a banked Bios for only the HD64180-based MicroMint SB-180 family. While it displayed an excellent and flexible interface and the ability to operate with a variety of peripherals, it had several quirks and noticeably degraded the computer performance. A banked Bios was also produced for the XLM-180 single board S-100 computer, but required special versions of many Z-System utilities, and was not produced in any significant quantity. Other spinoffs, such as the Epson portable, attempted banking of the Bios, but most failed to achieve our comprehensive goals of compatibility with the existing software base, high performance, and portability.

In 1989, Cam developed the first prototype of B/P Bios in a Non-banked mode on his TeleTek while Hal concentrated on extending ZSDOS and the Command Processor. As of mid-1992, B/P Bios has been installed on:

YASBEC -	Z180 CPU, FD1772 FDC, DP8490 SCSI, 1MB RAM
Ampro LB w/MDISK -	Z80 CPU, FD1770 FDC, MDISK 1MB RAM
MicroMint SB-180 -	HD64180 CPU, SMS9266 FDC, 256KB RAM
Compu/Time S-100 -	Z80 CPU, FD1795 FDC, 1 MB RAM
Teletek -	Z80 CPU, NEC765 FDC, 64KB RAM

2 Features of B/P Bios.

B/P BIOS is designed to be completely compatible with the CP/M 2.2 standards for a Basic IO System, as well as to provide many extensions needed for banked memory which is becoming so prevalent with newer systems and processors. Additionally, strict coding standards used in the various modules forming the BIOS ease interface problems with applications programs and provide a more robust framework for future development. The extensions added to the basic CP/M 2.2 foundation include many elements from Digital Research's CP/M 3 (aka CP/M Plus), but in a more logically consistent manner. Also included in banked versions are provisions for managing up to 8 MB of extended memory for banked applications, RAM Drives and potentially multitasking in future versions. To provide insight into the methodology used, let us now examine some of the features in a generic B/P Bios.

2.1 Character IO.

As defined by Digital Research in their CP/M 2.2 standards, character IO consisted of logical devices referred to as TTY, CRT, UC1, CON, etc. B/P Bios extends and generalizes these interfaces using the IOBYTE to define four physical devices called COM1, COM2, PIO and NUL. The first two, COM1 and COM2, are serial ports; PIO is a Parallel port, while NUL is a "bit-bucket" which can be replaced by a customized driver, or used in lieu of an actual device. Digital Research provided only a limited interface capability to the character devices in CP/M 2.2, consisting of a Console (CON), an auxiliary Input and Output (RDR/PUN), and a Printer (LST). The ability to sense Input and Output Status with these devices was extremely limited and was enhanced in CP/M 3. These enhanced capabilities are completely incorporated into B/P Bios with the addition of strict register usage so that only relevant registers may be altered in the respective routines. By manipulating the IOBYTE, any of the four physical devices may be used in the three logical devices of CONsole, AUXiliary, and Printer (LST).

Also featured in B/P Bios are modifications of CP/M 3 functions to initialize (or re-initialize) all devices and parameters, and return the address of a table which contains names and parameters of the defined character devices. While not totally compatible with CP/M 3 equivalents, these functions are consistent with the spirit and functionality needed with this advanced system. Included in the device table are; flags defining whether the device is capable of Input, Output or Both, Data rates for serial devices (Maximum and Set), Serial data format where applicable, and Handshaking method (CTS/RTS, XON/XOFF or None), as well as Input and Output Data masks for stripping unneeded bits from characters during IO.

2.2 Mass Storage IO.

All versions of Digital Research's CP/M BIOSes define only a generic Disk driver with implementations of Floppy, Hard, RAM and Tape drives left to the user or developer. In B/P Bios, we went several steps further to ease many problems. First, we retained all standard CP/M 2.2 functions and parameters, added CP/M 3 features for returning the Disk Parameter Header (DPH) table address, and flushing of the software deblocking code segment, and added a new vector to the BIOS jump table to provide a standard method of directly addressing low-level device functions. Several standard low-level Floppy Disk functions are supported and used by the standard utilities, including a function to return the type of Disk Controller in use which permits a single support utility to adapt to a wide variety of hardware platforms. In a like manner, low-level functions are provided for SCSI/SASI Hard Disk drives, and provisions for RAM Disk drives in the event special hardware is implemented. The methods used to implement these access mechanisms may be logically extended to handle Tape Drives or Network Interfaces.

2.3 Clock Support for Time and Date.

Many Hardware vendors have added provisions for Time and Date as non-standard extensions to CP/M 2.2 BIOSes, and more have incorporated such support into CP/M 3 BIOSes. We opted to define the CP/M 3 clock vector as a ZSDOS-standard clock building on our previous Operating System work. This entry point into the Bios completely complies with our ZSDOS standards and can completely replace the separate clock driver when used with ZSDOS. For systems capable of returning tenths-of-seconds, such as the YASBEC and SB-180, the standard has been enhanced to support this capability as well.

2.4 Banked Memory Support.

While Digital Research added banked memory support to their CP/M 3, it was in a manner incompatible with Bios interface standards defined for earlier CP/M standards. The method used in B/P Bios is compliant with CP/M 2.2 in direct accessing of Bios functions with only one minor exception when using the Banked ZSDOS2, and contains many of the CP/M 3 extensions added for banked memory support, with some being modified to be consistent with standards adopted for Z-System software. The exception to CP/M 2.2 accesses occurs when the Operating System can access certain buffers in the System Memory Bank. With ZSDOS 2, Allocation Bit Buffers (ALV), Check Buffers (CSV), and the Disk Host Buffer are all contained in the System Bank and not directly accessible from Transient Programs. To compensate for this, we have added a command to ZSDOS 2 to return the free space on disks (the most common reason for accessing these buffers) and tailored several utilities to adapt to banked and non-banked systems.

In addition to the primitives initiated by Digital Research, we added functions to directly access Words and Bytes in extended banks of memory, Directly accessing software routines contained in alternate memory banks, and properly managing the system when errors occur. These features make B/P Bios much more robust and resilient than other products. These features are implemented by methods transparent to the system utilities so that the same functions are available in both banked and non-banked versions.

2.5 Other Features.

B/P Bios contains a standardized identification method which may be used to determine the hardware on which the software is operating. This allows applications to "adapt" to the environment in a manner similar to that used in the rest of the Z-System community. It also minimizes system "crashes" by executing programs which assume certain hardware features which may be detrimental if executed on other systems. The effects of identification of physical system parameters is most readily noticed by virtue of a single suite of support programs performing low-level functions such as formatting and diagnostics which function across widely differing hardware platforms. Portability on this scale can rarely be seen in other computer systems.

The ZCPR 3.4 Environment with extensions is mandatory in a B/P Bios system. Beginning with the addition of System Segment address and size information for CPR, DOS and BIOS which were added in the ZCPR 3.4 Environment, B/P Bios also adds a Resident User Space which may be used to locate unique routines for custom applications in a manner similar to, but more consistent than NZ-COM. An Environment Version number of 90H identifies the Z3 Environment as being compliant with B/P definitions.

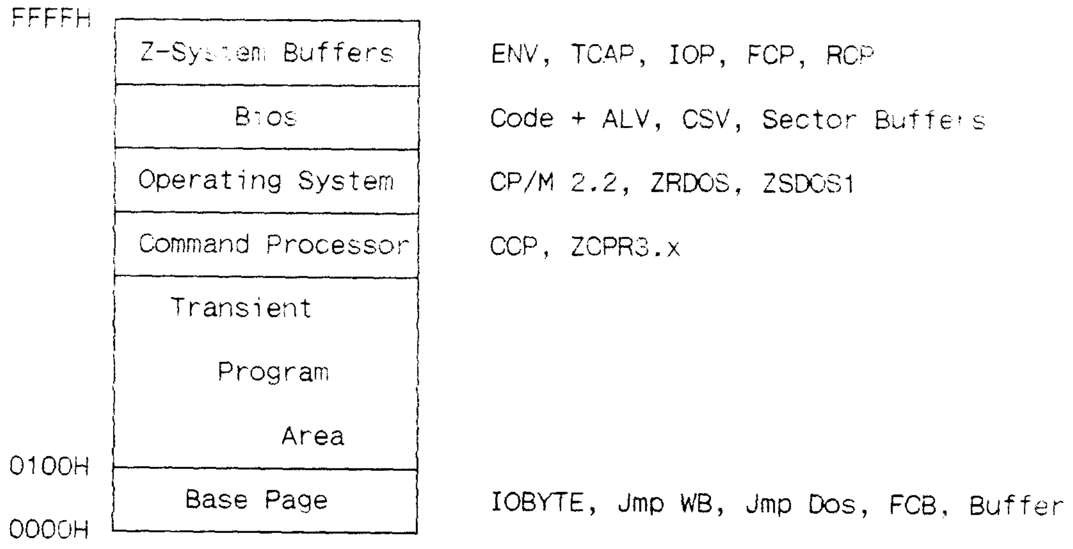
In Banked systems, application programs may also be placed in alternate memory banks using location and sizing information contained at standard positions within the Bios Header Structure. This feature permits significantly greater functionality without sacrificing precious Transient Program Area. While the scheme employed in the initial distribution is subject to minor adjustments as the banked ZSDOS2 becomes more firmly developed, experimentation and suggestions into this realm are encouraged.

3. Tailoring a B/P Bios.

To customize a B/P Bios for your use, or adapt it to a new hardware set, you will need an editor and an assembler capable of producing standard Microsoft Relocatable files. Systems using the Hitachi HD64180 or Zilog Z180 must be assembled with either ZMAC or SLR180 which recognize the extended mnemonic set, or with a Z80 assembler and MACRO file which permits assembly of the extended instructions. For Z80 and compatible processors, suitable assemblers include ZMAC and Z80ASM. For any assembler, failure to produce standard Microsoft Relocatable code will preclude the ability of our Standard utilities to properly install B/P Bios systems.

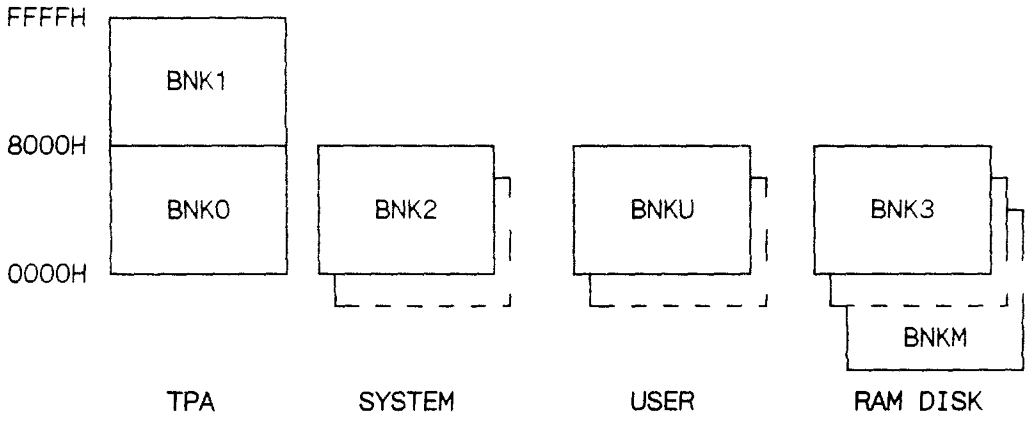
3.1 Theory of Operation.

In order to understand the need for, and principles behind B/P Bios, you must understand the way in which CP/M 2.2, as modified by the Z-System, uses the available memory address space of a Z80 microprocessor. For standard versions of CP/M and compatible systems, the only absolute memory addresses are contained in the *Base Page* which is the range of 0 to 100H. All addresses above this point are variable (within certain limits). User programs are normally run from the Transient Program Area (TPA) which is the remaining space after all Operating System components have been allocated. The following depicts the assigned areas pictorially along with some common elements assigned to each memory area:



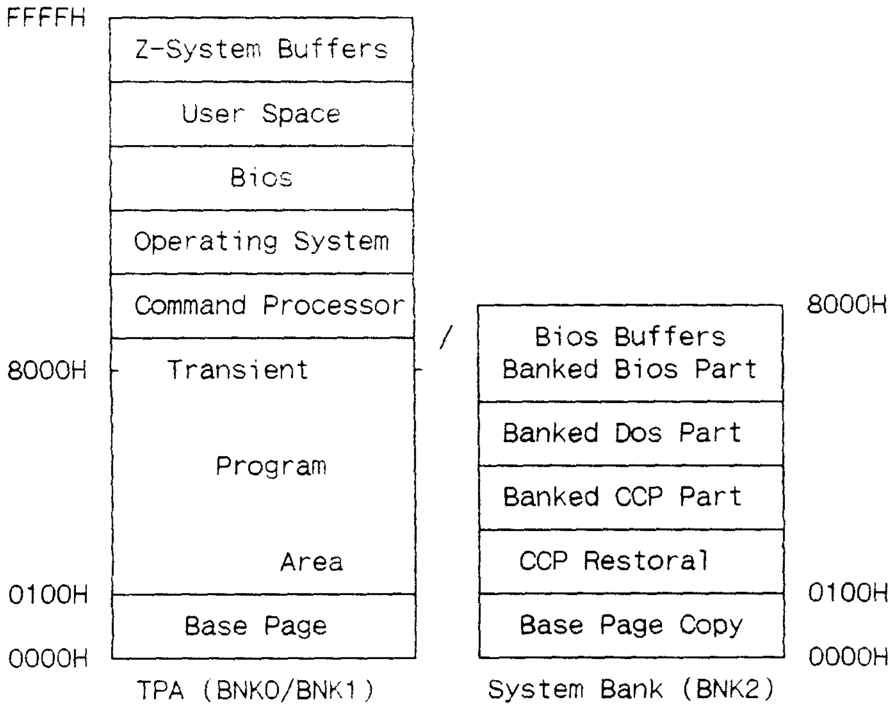
As more and more functionality was added to the Z-System Buffers, bigger drives were added using more ALV space, and additional functionality was added to Bios code in recent systems, the available TPA space has become increasingly scarce.

B/P Bios attacks this problem at the source in a manner which is easily adaptable to different hardware platforms. It uses additional memory for more than the traditional role of simple RAM Disks, it moves much of the added overhead to alternate memory banks. The generic scheme appears pictorially as:



As can be seen from the above diagram, multiple banks of memory may be assigned to different functional regions of memory, with each 32k bank (except for the one defined as BNK1) being switched in and out of the lower 32k of the processor's memory map. The bank defined as BNK1 is ***ALWAYS*** present and is referred to as the *Common Bank*. This bank holds the portions of the Operating System (Command Processor, Operating System, BIOS, and Z-System tables) which may be accessed from other areas, and which therefore must always be "visible" in the processor's memory. It also contains the code to control the Bank switching mechanisms within the B/P Bios.

To illustrate this functional division, the memory map of a basic B/P Bios system is divided as:



The B/P Bios banking concept defines a one byte Bank Number permitting up to 8 Megabytes to be directly controlled. Certain assumptions are made in the numbering scheme, the foremost of which is that BNK0 is the lowest physical RAM bank, BNK1 is the next incremental RAM bank, with others follow in incre-menting sequential order. A couple of examples may serve to illustrate this process. The YASBEC is offered with a couple of options in the Memory Map. Units with the MEM-1, 2 or 3 decoder PALs assign the first 128k bytes of physical memory to the Boot ROM, so BNK0 is set to 4 (Banks 0-3 are the ROM). The MEM-4 PAL only uses the first 32k (Physical Bank 0) for the ROM which means that BNK0 is assigned to 1, BNK1 to 2 and so on up to the 1 Megabyte maximum where BNKM is 31.

The Ampro Little Board equipped with MDISK, on the other hand, completely removes the Boot ROM from the memory map leaving a maximum of 1 MB of contigu-ous RAM space. In this system, BNK0 is set to 0 and BNKM to 31 of a fully equipped 1 MB MDISK board.

The region beginning after BNK1 is referred to as the System Bank. It begins at the bank number assigned to BNK2 and ends at the bank number immediately before that assigned to the User Bank, BNKU if present, or BNK3 if no User Bank area is defined.

If present, one or more 32k banks of memory may be defined with the BNKU equate for unique user programs or storage areas. This area begins with the bank number set to the label and ends at the bank number immediately before the BNK3 label. BNK3 defines a high area of physical memory which is most often used for a RAM Disk providing fast temporary workspace in the form of an emulated disk drive.

B/P Bios contains protection mechanisms in the form of software checks to insure that critical portions of the memory map are enforced. In the case of Non-banked systems, a check is made to insure that the system size is not so great that the Bios may overwrite reserved Z-System areas in high memory (RCP, IOP, etc). If a possible overflow condition is detected, the message

```
++ mem ovfl ++
```

will be issued when the system is started. In Banked Bios systems, this message will be displayed if the top of the system portions in the SYStem Bank exceeds the 32k bank size. For most systems, this space still permits drives of several hundred megabytes to be accommodated.

Since the Common portions of the operating system components must remain visible to applications, a similar check is made to insure that the lowest address used by the Command Processor is equal to or greater than 8000H. This factor is checked both in both MOVxSYS and BPBUILD with either a warning issued in the case of the former, or validity checks on entry in the case of the latter.

3.2 B/P Bios Files.

This BIOS is divided into a number of files, some of which depend highly on the specific hardware used on the computer, and some of which are generic and need not be edited to assemble a working system. Much use is made of conditional assembly to tailor the resulting Bios file to the desired configuration. The Basic file, BPBIO-xx.Z80, specifies which files are used to assemble the Bios image under the direction of an included file, DEF-xx.LIB. It is this file which selects features and contains the Hardware-dependent mnemonic equates. By maintaining the maximum possible code in common modules which require no alterations, versions of B/P Bios are relatively easy to convert to different machines. The independent modules used in the B/P Bios system are:

BOOTRAM.Z80	- (only needed in BOOT ROM applications)
BOOTROM.Z80	- (only needed in BOOT ROM applications)
BYTEIO.Z80	- Character IO per IOBYTE using IIO-xx routines
DEBLOCK.Z80	- Disk Deblocking routines
DPB.LIB	- 3.5/5.25" Floppy Format Definitions (if AutoSelect)
DPB8.LIB	- 8"/Hi-Density Floppy Format Definitions (if AutoSelect)
DPB2.LIB	- Additional Floppy Definitions (optional if AutoSelect)
DPBRAM.LIB	- Fixed Floppy Format Definitions (if Not AutoSelect)
DPH.LIB	- Disk Parameter Header Table & Floppy definitions
FLOPPY.Z80	- Floppy Disk High-Level Control
SECTRAN.Z80	- Sector Translate routines
SELFLP1.Z80	- Floppy Select routine (if Not auto selecting)
SELFLP2.Z80	- Floppy Select routine (if auto selecting)
SELRWD.Z80	- Generic Read/Write routines
Z3BASE.LIB	- ZCPR 3.x file equate for Environment settings

Other files are hardware version dependent to varying extents. These modules requiring customization for different hardware systems are given names which end with a generic "-xx" designator to identify specific versions. Tailoring these modules ranges from simple prompt line customization to complete re-writes. Versions of B/P Bios generated to date are identified as:

"-18"	- MicroMint SB-180	(64180 CPU, 9266 FDC, 5380 SCSI)
"-YS"	- YASBEC	(Z180 CPU, 1772 FDC, DP8490 SCSI)
"-AM"	- Ampro Little Board	(Z80 CPU, 1770 FDC, 1MB MDISK)
"-CT"	- Compu/Time S-100 board set	(Z80 CPU, 1795 FDC, 1MB Memory)
"-TT"	- Teletek	(Z80 CPU, 765 FDC)

Files associated with specific hardware versions or require tailoring are:

BPBIO-xx.Z80	- Basic file, tailored for included file names
CBOOT-xx.Z80	- Cold Boot routines, Sign-on prompts
DEF-xx.LIB	- Equates for option settings, mode, speed, etc.
DPBHD-xx.LIB	- Hard Drive Partition Definitions (optional)
DPBM-xx.LIB	- Ram Drive Definition (optional)
DPHHD-xx.LIB	- Hard Drive DPH definitions (optional)
DPHM-xx.LIB	- Ram Drive DPH Definition (optional)
FDC-xx.Z80	- Floppy Disk Low-Level interface/driver routines
HARD-xx.Z80	- Hard Drive Low-Level interface/driver routines (optional)

IBMV-xx.Z80	- Banking Support Routines (if banked)
ICFG-xx.Z80	- Configuration file for speed, Physical Disks, etc
IIO-xx.Z80	- Character IO definitions and routines
RAMD-xx.Z80	- Ram Drive interface/driver routines (optional)
TIM-xx.Z80	- Counter/Timer routines and ZSDOS Clock Driver
WBOOT-xx.Z80	- Warm Boot and re-initialization routines

3.3 B/P Bios Options.

The most logical starting point in beginning a configuration is to edit the DEF-xx.LIB file to select your desired options. This file is the basic guide to choosing the options for your system, and some careful choices here will minimize the Bios size and maximize your functionality. Some of the more important options and a brief description of them are:

MOVCPM - Integrate into MOVCPM "type" loader? If the system is to be integrated into a MOVCPM system, the Environment descriptor contained in the CBOOT routine is always moved into position as part of the Cold Start process. If set to *NO*, a check will be made to see if an Environment Descriptor is already loaded, and the Bios copy will not be loaded if one is present.

NOTE: When assembling a Bios for Boot Track Installation (MOVCPM set to YES), many options are deleted to conserve space and the Bios Version Number is forced to 1.1.

BANKED - Is this a banked BIOS? If set to *YES*, the Bank control module, IBMV, is included in the assembly, and much of the code is relocated to the system bank. Note that a Banked system CANNOT be placed on the System Tracks, or integrated into a MOVCPM image.

IBMOVS - Are Direct Inter-Bank Moves possible? If set to *YES*, direct transfer of data between banks is possible such as with the Zilog Z180/Hitachi 64180. If *NO*, a 256-byte transfer buffer is included in high Common Memory and Inter-bank moves require transfer of bytes through this buffer.

ZSDOS2 - Assemble this for a Banked ZSDOS2 system? If *YES*, the ALV and CSV buffers will be placed in the System bank invisible to normal programs. This has the side effect that many CP/M programs which perform sizing of files (Directory Listers, DATSWEEP, MEX, etc) which do not know about this function will report erroneous sizes. The advantage is that no sacrifice in TPA is required for large Hard Disks. Set this to *NO* if you want strict CP/M 2.2 compatibility.

FASTWB - Restore the Command Processor from the System Bank RAM? If set to *YES*, Warm Boots will restore the Command Processor from a reserved area in the System RAM bank rather than from the boot tracks. For the maximum benefit of B/P Bios, always attempt to set this to *YES*. In systems without extended memory, it MUST be set to *NO*.

MHZ - Set to Processor Speed in closest even Megahertz (e.g. for a 9.216 MHz clock rate, set to 9). The value entered here is used in many systems to compute Timing values and/or serial data rate parameters.

CALCSK - Calculate Diskette Skew Table? If *NO*, a Skew table is used for each floppy format included in the image. Calculating Skew is generally more efficient from a size perspective, although slightly slower by factors which are so small as to be practically unmeasurable.

HAVIOP - Include IOP code into Jump table? If the IOPINIT routine satisfies your IOP initialization requirements, you may turn this off by setting to *NO* and save a little space. This typically will be turned off when generating a system for MOVCPM integration to conserve space.

INROM - Is the Alternate Bank in ROM? Set to *NO* for Normal Disk-based systems. Please contact the authors if you need additional information concerning ROM-based system components.

BIOERM - Print BIOS error messages? Set this to *YES* if you desire direct BIOS printing of Floppy Disk Error Messages. If you are building a BIOS for placement on Boot Tracks, however, you will probably not have room and must turn this Off. Set to *NO* to simply return the normal Success/Fail error flag with no Message printout.

FLOPY8 - Include 8"/Hi-Density Floppy Formats? Some systems (SB-180, Compu/Time) can handle both 5.25" and 8" disks. If your hardware supports the capability and you want use 8" disks as well as the normal 3.5 and 5.25" diskettes, setting this to *YES* will add formats contained in DPB8.LIB and control logic to the assembly. Future systems may take advantage of the "High-Density" 3.5 and 5.25" Floppy Disks which use higher data rates. Their definitions will be controlled by this flag as well.

NOTE: If AUTOSL is set to *NO*, this option will probably cause the BIOS to be larger than necessary since these additional formats may not be accessible.

MORDPB - Use more Floppy DPB's (in addition to normal 4-5.25" and optional 8")? If *YES*, the file DPB2.LIB is included. Many of the formats are Dummies and may be filled with any non-conflicting formats you desire.

NOTE: If AUTOSL if set to *NO*, this option will probably cause the BIOS to be larger than necessary since these additional formats may not be accessible.

MORDEV - Include Additional Character Device Drivers? Is set to *YES*, user-defined drivers are added to the Character IO table, and associated driver code is assembled. Systems featuring expansion board such as the SB-180 and YASBEC may now take advantage of additional serial and parallel interfaces within the basic Bios. Set to *NO* to limit code to the basic 4 drivers.

NOTE: When assembling a Bios for Boot Track Installation (MOVCPM set to YES), MORDEV is overridden to conserve space, and the Bios Version Number is forced to 1.1 in the distribution files.

BUFCON - Use type ahead buffer for the Console? If set to *YES*, code is added to create and manage a type-ahead buffer for the driver assembled as the console. This device will be controlled by either interrupts (in systems such as the YASBEC and SB-180) or background polling (in Ampro and Compu/Time). This means that characters typed while the computer is doing something else will not be lost, but will be held until requested.

BUFAUX - Use type ahead buffer on Auxiliary Port? As with BUFCON above, setting to *YES* will add code to create and manage a type ahead buffer for the auxiliary device. Since the AUX port typically is used for Modem connections, buffering the input will minimize the loss of characters from the remote end.

AUTOSL - Auto-select floppy formats? If set to *YES*, selection of Floppy disks will use an algorithm in SELFLP2.Z80 to identify the format of the disk from the DPB files included (DPB.LIB, optional DPB8.LIB, and optional DPB2.LIB) and log the disk if a match is found. There must be *NO* conflicting definitions included in the various files for this to function properly. See the notes in the various files to clarify the restrictions. If set to *NO*, the single file DPBRAM.LIB is included which may be tailored to contain only the fixed format or formats desired per disk drive. This results in the smallest code requirement, but least flexibility.

RAMDSK - Include code for a RAM-Disk? If set to *YES*, any memory above the System or User bank may be used for a RAM Drive (default is drive M:) by including the file RAMD-xx.Z80. Parameters to determine the size and configuration are also included in the files DPHM-xx.LIB and DPBM-xx.LIB. In systems without extended memory, or to conserve space such as when building a system for the boot tracks, this may be disabled by setting to *NO*.

HARDDSK - Include SCSI Hard Disk Driver? Set to *YES* if you wish to include the ability to access Hard Disk Drives. In a floppy-only system, a *NO* entry will minimize BIOS code.

HDINTS - (System Dependent) In some systems such as the YASBEC, Interrupt-driven Hard Disk Controllers using DMA transfer capabilities may be used. If you wish to use this type of driver specified in the file HARDI-xx.Z80 instead of the normal polled routines included in HARD-xx.Z80, set this option to *TRUE*. In most cases, this driver will require more Transient Program Area since the Interrupt Handling routine must be in Common Memory.

CLOCK - Include ZSDOS Clock Driver Code? If set to *YES*, the vector at BIOS+4EH will contain a ZSDOS-compatible clock driver with the physical code contained in the TIM-xx.Z80 module. If set to *NO*, calls to BIOS+4EH return an error code.

TICTOC - (System Dependent) Use pseudo heartbeat counter? This feature is used in systems such as the Ampro Little Board and Compu/Time SBC880 which do not have an Interrupt scheme to control a Real Time Clock. Instead, a series of traps are included in the code (Character IO Status polls, Floppy Disk Status polls) to check for overflow of a 1-Second Counter. It is less desirable than an Interrupt based system, but suffices when no other method is available. Set to *NO* if not needed.

QSIZE - Size in bytes of type ahead buffers controlled by BUFCON and BUFAUX.

REFRSH - Activate Dynamic Refresh features of Z180/HD64180 processors? In some computers using these processors such as the YASBEC, refresh is not needed and merely slows down processing. Set to *NO* if you do not need this feature. If your processor uses dynamic memory, or needs the signal for other purposes (e.g. The SB180 uses Refresh for Floppy Disk DMA), Set this to *YES*.

Z3 - Include ZCPR init code? Since a Z3 Environment is mandatory in a B/P Bios (which now "owns" the Environment), this option has little effect.

For assembly of a Banked version of B/P Bios, the identification of various banks of memory must be made so that the various system components "know" where things are located. Refer to Section 3.1 above for a description of these areas. The BNK0 value should be the first bank of RAM in the System unless other decoding is done. The following equates must be set:

BNK0	- First 32k TPA Bank (switched in/out)
BNK1	- Second 32k TPA Bank (Common Bank)
BNK2	- Beginning of System Bank (BIOS, DOS, CPR) area
BNKU	- Beginning of Bank sequence for User Applications
BNK3	- Beginning of Extra Banks (first bank to use for RAM Disk)
BNKM	- Maximum Bank Number assigned

3.4 Configuration Considerations.

When assembling a version of B/P Bios for integration into an IMG file, size of the resulting image is not much of a concern, so you need not worry about minor issues of size. For integration into a system for loading onto diskette boot tracks, however, the limitation is very real in order to insure that the CPR/DOS/BIOS and Boot Sector(s) can fit on the reserved system tracks. Typically, a limit of slightly under 4.5k exists for the Bios component. When the MOVCPM flag is set to YES for this type of assembly, warnings will be issued when the image exceeds 4352 bytes (the maximum for systems with 2 boot records), and 4480 bytes (the maximum for systems with a single boot record). Achieving these limits often requires disabling many of the features.

The first thing you should do before assembling the BIOS is to back up the entire disk, then copy only the necessary files onto a work disk for any editing. After setting the options as desired, edit the hardware definitions in ICFG-xx.Z80 to reflect the physical characteristics of your floppy and hard drives, as well as any other pertinent items. Then edit the logical characteristics for your Hard and Ram Drives (if any) in DPBHD-xx.LIB and DPBM-xx.LIB. If you do not desire any of the standard floppy formats or want to change them, edit DPB.LIB and/or DPB2.LIB (if using auto selection) or DPBRAM.LIB if you are using fixed floppy formats. Finally edit the DPH files to place the logical drives where desired in the range A..P.

Decide whether you want to generate a system using the Image file construct developed in support of B/P Bios (BPBUILD/LDSYS), or for integration on a floppy disk's boot tracks. If the latter, you probably will not be able to have all options turned on. For example, with the MicroMint SB-180, the following options must be turned Off: BANKED, ZSDOS2, BIOERM, FLOPYS, MORDPB, BUFAUX and usually either CLOCK or RAMDSK. As an aid to space reduction, conditional assembly based on the MOVCPM flag automatically inhibits all but double-sided Floppy formats from DPB.LIB. If configuring for Floppy Boot tracks (MOVCPM flag set to TRUE), a warning will be printed during assembly if the size exceeds that available for a One or Two-sector boot record. Using the BPBUILD/LDSYS method, you may vary nearly all system parameters, even making different systems for later dynamic loading.

If you are using a version of the B/P Bios already set for your type of computer, you are now ready to assemble, build a system and execute it. The only remaining task would be an optional tailoring of the sign on banner in the file CBOOT-xx.Z80 and reassembly to a .REL file.

For those converting a standard version of the B/P Bios to a new hardware system, we recommend that you begin with a Floppy-only system in Non-Banked mode then expand from there. The easiest way to test out new versions is to use the System Image (IMG file) mode, then advance to boot track installations if that is desired. Enhancements that can be added after testing previous versions may be to add Hard Drives, RAM Drive, and finally Banking.

4 Installing a B/P Bios.

The Distribution diskette(s) on which B/P Bios is furnished are configured for booting from the vanilla hardware for the version ordered. A 9600 bps serial terminal is standard, and will allow you to immediately bring up a minimal non-banked floppy disk system. Due to the variety of different system configurations and size restrictions in some versions, only the Floppy Disk Mass Storage capability can be assured on the initial boot disk. Where space remained on the boot tracks, limited Hard Drive support is also provided, and in some configurations, even RAM Drive support exists.

After booting from either an established system, or the boot tracks of the distribution disk, format one or more fresh diskettes and copy the distribution diskette(s) contents to the backup diskette(s). Copy the boot tracks from the master to the copies using BPSYSGEN (see 6.6). Remove the master diskette(s) for safekeeping and work only with the copies you just made.

Using the backup diskette with the B/P utilities on it, execute BPCNFG in the Boot Track configuration mode (see 6.2), adjusting all the options to your specific operating environment. When you have completed tailoring the system, it is ready for booting by placing the diskette in drive A: and resetting the system.

The sample STARTUP.COM file on the distribution disk will automatically execute a sequence of instructions when the system is booted. It contains various instructions which further tailor the system and load portions of the operating system which are too big to fit on the boot tracks. The default instruction sequence is:

```
LDDS                              <-- Load the DateStamper style File
                                      Stamp routine and clock
LDR SYS.RCP,SYS.FCP,SYS.NDR       <-- Load ZCPR 3 Environment segments
                                      for Resident Command Processor,
                                      Flow Control Pkg and Named Dirs
IOPINIT                           <-- Initialize the IO Processor Pkg
TD S                              <-- Prompt for Date and Time. Set Clk
                                      Alternatives are to use TDD
                                       (6.21) or SETCLOK (6.18)
IF ~EX MYTERM.Z3T                 <-- If the file MYTERM.Z3T does Not
                                      exist...
TCSELECT MYTERM.Z3T               <-- ..select which terminal you have
                                      creating a MYTERM.Z3T file
FI                                <-- ...end of the IF
LDR MYTERM.Z3T                    <-- Load the Terminal Definition data
```

If you wish to alter any of these initial instructions to, for example, initialize the RAM drive using INITRAMD, add File Time Stamp capabilities to it with INITDIR or PUTDS and copy some files there with COPY, these may be added with ALIAS, VALIAS, SALIAS or other compatible files available from the ZSYSTEM or ZCPR33 areas on Z-Nodes.

After the initial system is up and running from the Default Boot Track system, you may expand the operation by generating systems for different purposes in order to gain the most advantage from your system. Many types of installation are possible, the simplest of which is a Non-Banked system using only 64k of the systems memory, all of which is in primary memory. Such a system uses a normal Command Processor such as the ZCPR3.x family, and a Non-Banked Operating System such as our ZSDOS Version 1. Non-Banked systems may be installed on a Disk's Boot Tracks, or created as an Image File for dynamic loading using the LDSYS Utility (see 6.15).

Banked systems *MUST* be created with the BPBUILD Utility (see 6.1) and loaded with LDSYS (see 6.15). The techniques to manage different memory banks to form a complete Operating Environment are rather intricate and are best handled by our utilities. Many Image files may be created and loaded as needed to tailor your system for optimum performance. The following sections describe these various types of installations in detail.

4.1 Boot Track Installation.

For most of the existing CP/M compatible computers to begin executing a Disk Operating System, a program must be placed on a specified area of a Floppy or Hard Disk Drive. Normally, the first two or three tracks on the disk are reserved for this purpose and are referred to as the "Boot Tracks". Since the space so defined is generally restricted, neither a complete B/P Bios nor a banked installation is possible. Instead, a scaled-down system roughly equivalent to those currently in use is used to start the computer and serve as the Operating System, with larger systems loaded later as needed.

If you are using a pre-configured version of B/P Bios for your hardware, you may simply continue to use the Boot Track system from the distribution disk(s) by copying the system as described in Section 4 above using BPSYSGEN (see 6.6). If you elect to alter or otherwise customize the Boot Track system, you must assemble the B/P Bios source setting certain of the equates in the DEF-xx.LIB file to insure a correct type of system. To assemble a Boot Track system, the most important equates are:

MOVCPM	Set to *YES*
BANKED	Set to *NO*
ZSDOS2	Set to *NO*

One element of Banked Systems is available in a Boot Track installation if additional memory is available, and your B/P Bios routines support such a feature. This feature reloads the Command Processor from Banked memory instead of from the Boot Tracks of a disk, and generally produces less code (taking less space on the Boot Tracks) and executes faster. It is set with:

FASTWB	Set to *YES* if desired, *NO* if Warm Boot from disk

Some of the features that generally need to be disabled to scale a smaller system are set as:

MORDPB	Set to *NO*
DPB8	Set to *NO*
MORDEV	Set to *NO*

When at least these equates and any others you desire to change (see section 4) have been made to the component files of the system, assemble your BPBIO-xx file to a MicroSoft standard REL file. This output file may be used to overlay the Bios portion of the MOVxSYS.COM system generation utility (see 6.16) furnished with your distribution disk, or an equivalent program provided with your computer. MOVxSYS or its equivalent (MOVCPM, MOVZSYS, etc) is a special program customized for your particular hardware containing all the Operating System components which will be placed on the Boot Tracks, along with a routine to alter the internal addresses to correspond to a specified memory size.

To Add the new Bios you just assembled, execute INSTAL12 (see procedures in 6.13) specifying your computer's MOVxSYS or equivalent program and follow the prompts to overlay the new Bios. Once INSTAL12 has saved a relocatable or absolute file, you are ready to create a boot disk containing the modified system.

If you used the command INSTAL12 to install system segments on MOVxSYS or equivalent program, you must first create an Absolute System Model file. Since the functional portion of your new program is identical to the original MOVxSYS or equivalent, use the method explained in your original documentation to generate a new system. With MOVxSYS, the command is:

```
MOVxSYS nn *                <-- replace MOVxSYS with your version
```

Where nn is the size of the system (typically 51 for a moderate boot system). The asterisk tells the program to retain the image in memory and not write it to a disk file. You may now use BPSYSGEN to write the new image to the system tracks of your boot diskette. Do this by executing BPSYSGEN with no arguments and issue a single Carriage Return when asked for the source of the image.

If you used the command INSTAL12 /A to install replacement system segments over a System Image file, or used a utility which wrote the new image to a disk file, use BPSYSGEN to write the image file to the system tracks of your boot disk. The proper command is

```
BPSYSGEN filename
```

where **filename** is the name of the disk file you just created by executing MOVxSYS or equivalent with output to a disk file, or with INSTAL12 on an existing image file.

If the system is written to a Hard Disk, and your system supports booting from a Hard Disk such as the YASBEC, you normally must alter the default Boot Sector from the default Floppy Disk Boot Sector contained in MOVxSYS or equivalent. This alteration is accomplished by HDBOOT (see 6.9) which must be customized to the specific Hardware System used.

After the above actions have been completed as appropriate, tailor the Boot Track system to reflect the desired starting configurations with BPCNFG (see 6.2). Such items as the desired Startup file name, Bank Numbers (critical if FASTWB is used), and drive types and assignments are routinely tailored at this point. When the you have finished this step, test your new system by resetting the system, or cycling the power and you should be up and running!

4.2 Non-Banked Image Installation.

A Non-Banked system may be installed as an Image File as opposed to the basic Boot Track installation covered in 4.1 above. To create an Image File, you must have REL or ZRL versions of a Command Processor (ZCPR3.x or equivalent recommended), an Operating (ZSDOS.ZRL recommended), and a REL version of B/P Bios for your system assembled with the **MOVCPM** equate in DEF-xx.LIB set to *NO*. Other equates in this file may be set as described above for the Boot Track system. Since Image Files are not as constrained in size as is installation for Boot Tracks, more features may generally be activated such as Error Messages, RAM Drive, additional Hard Drive partitions, and complete Floppy Format suites. The main precaution here is that large Hard Drives will rapidly cause significant loss of Transient Program Area since all Drive parameters must be in protected high memory above the Bios.

After the Bios has been assembled, an Image file must be produced. This is accomplished with the BPBUILD Utility (see 6.1). Set the File names in Menu 1 to reflect only Non-banked files (or minimally banked Bios if FASTWB is set to *YES*), and let BPBUILD do the work. Since the standard Non-Banked System segments are normally set to the "standard" CP/M 2.2 sizes, you may answer the "autosize" query with a *Y* to obtain the maximum Transient Program Area in the resulting system. When BPBUILD completes its work, a file, normally with the default type of .IMG, will have been placed in the currently logged Drive/User area and you are ready to perform the next step in preparation of the Non-Banked Image.

As with the Boot Track installation covered above, several system items must be tailored before the Image may be safely loaded and executed. This is done by calling BPCNFG with the Image file name as an argument, or specify Image configuration from the interactive menu (see 6.2). Set all items as you desire them in the operating system, particularly the Bank Numbers (if FASTWB is active), and the Disk Drive characteristics and assignments. When this has been satisfactorily completed, you are ready to load and execute the newly-created system.

Installing an Image File (default file type of .IMG) is extremely easy. Only the utility LDSYS.COM (see 6.15) is needed. If the file type has not been changed from the default .IMG, only the basic name of the Image File need be passed to LDSYS when executed as:

```
LDSYS IMGFILE          <-- Where IMGFILE is your Image file name
```

The operating parameters of the currently-executing system are first examined for suitability of loading the Image File. If it is possible to proceed, the Image File is loaded, placed in the proper memory locations, and commanded to begin execution by calling the B/P Bios Cold Boot Vector. The Cold Boot (Bios Function 0) performs final installation, displays any desired opening prompt and transfers control to the Command Processor with any specified Startup file for use by a ZCPR3.x Command Processor Replacement.

Since a non-banked Image File will probably closely resemble that contained on the Boot Tracks, the same STARTUP file may generally be used to complete the initial tailoring sequence. If a different file is desired, the Image File may be altered to specify a different file using BPCNFG.

4.3 Banked Bios, Non-banked System Installation.

With the B/P Bios system, an Image system may be created and loaded which places portions of the Bios *Only* in the System bank, retaining a non-banked Operating System and therefore maximum compatibility with existing applications software. A few thousand bytes can normally be reclaimed for Transient Programs in this manner, although large and/or increasing numbers of logical drives will still reduce TPA space because of the need to store Allocation Vector information in Common Memory.

To prepare such a system, simply edit the needed Bios files if necessary with particular emphasis on the DEF-xx.LIB file where the following equates must be set as:

MOVCPM	Set to *NO*
BANKED	Set to *YES*
ZSDOS2	Set to *NO*

Since banked memory *MUST* be available for this type of installation, you will probably want the Fast Warm Boot feature available to maximize system performance. To activate this option, set the following equate as:

FASTWB	Set to *YES*

When the editing is complete, assemble the Bios to a MicroSoft .REL file with an appropriate assembler such as ZMAC and build an Image system with BPBUILD (see 6.1) changing the Bios file name in menu 1 to the name of the newly created Bios file. Next, configure the default conditions if necessary with BPCNFG (see 6.2) and you are ready to activate the new system in the same manner as all Image files by calling LDSYS with the Image file argument as:

```
LDSYS BBSYS          <-- where BBSYS is your Image File Name
```

As with the completely Non-Banked system described above in Section 4.2, no new requirements are established for a Startup file over that used for the initial Boot System, since both the Command Processor and Disk Operating System are unbanked, and no data areas needed by application programs are placed in the System Bank. As with all Image Files, additional features such as full Bios Error Messages, more extensive Floppy Disk Formats and RAM drive may generally be included in the System definition prior to assembly since the size constraints of Boot Track systems do not apply.

4.4 Fully Banked Image Installation.

To create a system taking maximum advantage of banked memory, a special banked Operating System and Command Processor are needed. These have been furnished in initial form with this package as ZSDOS20.ZRL and Z40.ZRL respectively. They use the Banking features of B/P Bios and locate the maximum practicable amount of executable code and data in the System Bank. Of significant importance to maximizing the Transient Program Area is that the Drive Allocation Bit maps are placed in the System Bank meaning that adding large hard drives, or multiple drives produce only minimal expansion to the resident portion of the Bios.

A Fully banked Bios is created by editing the B/P Bios files as needed to customize the system to your desires. Insure that the following DEF-xx.LIB equates are set as:

MOVCPM	Set to *NO*
BANKED	Set to *YES*
ZSDOS2	Set to *YES*

Assemble the resultant B/P Bios to a MicroSoft .REL file. Build an Image file with BPBUILD (see 6.1) and configure the produced Image file with BPCNFG (see 6.2). When you are confident that all default settings have been made, activate the file by entering:

```
LDSYS FBANKSYS          <-- where FBANKSYS is your Image file name
```

Several differences may exist in the Startup file used for a Fully banked system. Generally the changes amount to deleting items such as a File Stamp module for the Non-banked ZSDOS1 which is not necessary with the fully-banked ZSDOS 2 and Z40. Only the type of clock need be specified for ZSDOS2. Furthermore, since the Z40 Command Processor Replacement contains most commonly-used commands gathered from a number of Resident Command Processor (RCP) packages, there is normally no need to load an RCP. A simple Startup file found adequate during development of the fully-banked B/P system is:

```
ZSCFG2 CB                  <-- Set ZSDOS 2 clock to Bios+4EH
LDR SYS.FCP,SYS.NDR        <-- Load ZCPR 3 Environment segments
                               for Flow Control and Named Dirs
IOPINIT                    <-- Initialize the IO Processor Pkg
TD S                       <-- Prompt for Date and Time. Set Clk
                               Alternatives are to use TDD
                                (6.21) or SETCLOK (6.18)
IF ~EX MYTERM.Z3T          <-- If the file MYTERM.Z3T does Not
                               exist...
TCSELECT MYTERM.Z3T        <-- ..select which terminal you have
```

```
                                        creating a MYTERM.Z3T file
    FI                                  <-- ...end if the IF
    LDR MYTERM.Z3T                      <-- Load the Terminal Definition data
```

Since the requirements for a fully-banked system differ significantly from a non-banked one, we recommend that you use a different name for the Startup file. For example, STARTUP.COM is the default name used with Boot Track systems for initial operation, and with Non-banked Image Files, while STARTB may be a suitable name for the script to be executed upon loading a fully-banked system. The name of the desired Startup file may be easily altered in either Boot Track or Image systems from Option 1 in BPCNFG (see 6.2).

An option available to start from a large Image File is to configure a Startup file for execution by the Boot Track system containing a single command. The command would simply invoke LDSYS with the desired Banked Image File as an argument such as:

```
            LDSYS BANKSYS          <-- Where BANKSYS.IMG is your Image file
```

In this case, none of the normal initialization sequences cited above would be executed by the Boot Track system, and only those contained in the Startup for BANKSYS.IMG would occur. Other options abound and are left to the community to invent new combinations and sequences.

4.5 In Case of Problems...

While we attempted to outline procedures for the majority of installations we considered feasible, there may be occasions where you inadvertently find yourself in a position where you seem to have lost the ability to get your system up and running.

PROBLEM: When loading an .IMG file with LDSYS, the screen displays the LDSYS banner, system addresses, and halts with the last screen displaying:
"...loading banked system".

-- **SOLUTION** Something is not set correctly in the Bios, since all lines after the last one displayed are printed from the newly-loaded Bios. One of the most common causes for this problem is incorrect bank number settings. Use the hidden selection in Menu 1 of BPCNFG (see 6.2) to verify that the correct bank numbers have been set for TPA and SYStem banks. Another common cause of this problem is incorrect settings for the Console port, or a setting in the IOBYTE which directs Console data to a device other than the one intended. Use Menu 2 BPCNFG to properly set the IOBYTE and the console parameters.

PROBLEM: You boot from or load a B/P Bios system from a Hard Drive, and immediately after starting, the system attempts to log onto Floppy Drive 0.

-- **SOLUTION:** The most common cause for this symptom is that the desired Hard Drive and Floppy Drive definitions were not swapped to define a Hard Drive Partition as the A: drive. Use BPCNFG (see 6.2), Menu 5 to exchange drives to the desired configuration. A similar situation may exist where a Hard Drive is activated immediately after booting when a Floppy drive is desired as the A: Drive.

PROBLEM: The computer seems to boot satisfactorily, but after a few programs or any program which executes a Warm Boot (or entering Control-C), the system goes into "Never-never Land" and must be reset.

-- **SOLUTION:** This symptom is most often caused by an inability to access and load the Command Processor. This is most probably caused by assembling B/P Bios with the FASTWB equate in DEF-xx.LIB set to *YES* when the system contains no extended memory, or incorrect settings of the Bank Numbers. To check Bank Number settings, use the hidden function in BPCNFG, Menu 1 (see 6.2).

PROBLEM: When doing a Cold Boot from a Hard Drive (from Power up or Reset), the system goes to a Floppy Drive *before* displaying the initial sign on messages, and remains logged on the Floppy.

-- **SOLUTION:** This is most often due to your forgetting to run the HDBOOT utility on the Hard Drive Boot system after applying it with BPSYSGEN. Normally, systems created with MOVxSYS contain a Floppy Disk Boot sector which will load the initial Operating System from a Floppy. HDBOOT (see 6.9) modifies this record on a specified Hard Drive Unit so that the Operating System is loaded from a Hard Drive. Run HDBOOT on the Desired Hard Drive, then use BPCNFG (see 6.2) to insure that the logical drives are positioned as desired (Menu 5).

PROBLEM: When Booting, the system console either doesn't display anything, or prints strange characters.

-- **SOLUTION:** This is most often due to incorrect settings for the current Console, most probably the Data rate, or CPU Clock Frequency. Boot from a good system, then use BPCNFG (see 6.2) to adjust the settings on the problem system. Pay particular attention to Menu 1 (CPU Clock Rate) and Menu 2 (IOBYTE and Serial Port Data Rates).

PROBLEM: When running a fully-banked system with ZSDOS 2, some programs seem to "hang" or "lock up" the system on exit.

-- **SOLUTION:** One of the most common sources of this symptom is with the application program where the author used code which assumes that the BDOS and Command Processor are of a certain size, or bear a fixed relationship to the addresses in page 0. You may experience this most often when using an IMG system built by answering *YES* to the Autosizing query in BPBUILD (see 6.1). To compensate for such ill-behaved programs, you may use a two-step build process as:

1) Use BPBUILD to create an IMG file answering *YES* to Autosizing on exit. This maximizes TPA placing the Resident Bios as high as possible in memory.

2) Execute BPBUILD again with an argument of the name you gave to the file

just created above. This loads the definition from the IMG file. Immediately exit with a Carriage Return, and answer *NO* to Autosizing, and *YES* to placing system segments at standard locations. This procedure keeps the Bios address constant, but will move the starting addresses of BDOS and Command Processor down, if possible, to simulate "standard" sizes used in CP/M 2.2.

5. Programming for B/P Bios.

For most existing purposes, programming for B/P Bios is no different than for standard CP/M 2.2 BIOSes. Even adapting CP/M 3 programs for a B/P Bios should present no great hurdle due to the close similarity retained with the corresponding extended functions. The power of a B/P Bios interface, however, is in using the combined features to produce portable software across a wide variety of hardware platforms by exercising all of the B/P Bios features in concert. This section describes the interfaces available to the programmer of a system using the B/P Bios, and the functions available to ease direct floppy and hard drive accesses for specialized programming in a consistent manner.

One of the architectural flaws which we considered in CP/M Plus was the odd way in which direct BIOS access was handled. We designed B/P Bios to be as compatible with CP/M 2.2 as possible, yet provide the expanded functionality needed in Banked applications. To that end, direct interface with BIOS calls follows CP/M 2.2 conventions as much as possible.

The following pages on programming assume some familiarity with the basic CP/M fundamentals, and with Z80/Z180 assembly language, since it is beyond the intent of this manual, and our literary writing skills, to present an assembly programming tutorial. Should you need additional assistance in this area, please refer to the annotated bibliography for reference material.

5.1 Bios Jump Table.

The BIOS Jump table consists of 40 Jumps to various functions within the BIOS and provides the basic functionality. It includes the complete CP/M 2.2 sequence, most of the CP/M 3 (aka CP/M Plus) entry points (although some differ in parameter ordering and/or register usage), and new entry points needed to handle banking in a consistent and logical manner.

Bios entry points consist of a Table of Absolute 3-byte jumps placed at the beginning of the executable Image. Parameters are passed to the Bios in registers as needed for the specific operation. To avoid future compatibility problems, some of the ground rules for Bios construction include: No alteration of Alternate or Index registers as a result of Bios calls, and all registers listed in the documentation as being Preserved/Unaffected MUST be returned to the calling program in their entry state. Bios entry points are:

Function 0 (xx00)	Cold Boot
Enter: None	Exit: None. Execution resumes at CPR Uses: All Registers

Execute Cold Start initialization on the first execution. The jump argument is later overwritten, and points to the IOP Device jump table. The reason for this is that code to perform the initialization is often placed in areas of memory which are later used to store system information as a memory conservation measure. Attempts to re-execute the initialization code would then encounter data bytes instead of executable instructions, and the system would most assuredly "crash".

Among other functions performed during initial execution of the Cold Boot code are: Establishing an initial Z3 Environment if necessary, initializing any Z3 system segments such as an Extended Path, Flow Control Package, Named Directory Buffer and such: setting system-specific values such as the locations of Allocation Vector buffers for RAM and Hard Drives: and executing the Device Initialization routine (see Function 21). The Cold Boot routine usually exits by chaining to the Warm Boot Function (Function 1) to set vectors on Page 0 of the TPA memory bank.

Function 1 (xx03)	Warm Boot
Enter: None	Exit: None. Execution returns to CPR Uses: All Registers

This function re-initializes the Operating System and returns to the Command Processor after reloading it from the default drive boot tracks, or banked memory if the Bios was assembled with the Fast Warm Boot option.

Unless altered by an ill-behaved Resident System Extension (RSX) or other operating transient program, the Warm Boot Vector at location 0 in memory points to this vector. Well-behaved programs will not alter this address but should, instead, alter the destination argument of the Jump vector in the Bios header. There is a singular exception to this in the case of NZCOM where the Warm Boot vector points to the NZBIOS, and Not the 'Real' Bios. In such a case, the address of the "Real" Bios must be separately determined (See Function 30).

Function 2 (xx06)	Console Input Status
Enter: None	Exit: A = 0FFH if Char Ready, NZ A = 0 if No Char Ready, Z Uses: AF

This function returns a flag indicating whether or not a character has been entered from the Console device selected by the IOBYTE on Page 0 of the TPA bank. The return status is often used by Transient Programs to determine if the user has attempted to start or stop program execution.

Function 3 (xx09)	Console Input
Enter: None	Exit: A = Masked Input Character Uses: AF

This function waits for a character to be entered from the Console device selected by the IOBYTE on Page 0 of the TPA Bank, and returns it to the calling routine. According to strict CP/M 2.2 standards, the Most Significant bit of the input byte must be set to Zero, but this may be altered by the input mask for the Console Device.

Function 4 (xx0C)	Console Output
Enter: C = Character to send to Console	Exit: None. Uses: AF

This function sends a specified character to the Console Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask bits as specified in the Character Device Configuration for the device as an Output.

Function 5 (xx0F)	List Output
Enter: C = Character to send to List Dev (Printer)	Exit: None. Uses: AF

This function will send a specified character to the List Device (Printer) defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask it as specified in the Character Device Configuration for the Output device.

Function 6 (xx12)	Auxiliary Output
Enter: C = Character to send to Auxiliary Device	Exit: None. Uses: AF

This function will send a specified character to the Auxiliary Output Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for the device to become ready, if necessary, before sending the character, and will mask it as specified in the Character Device Configuration for the Output device.

Function 7 (xx15)	Auxiliary Input
Enter: None	Exit: A = Masked Input Character Uses: AF

This function will read a character from the Auxiliary Input Device defined by the IOBYTE on Page 0 of the TPA Bank. It will wait for a character to be received, and will mask it as specified in the Character Device Configuration for the Input device.

Function 8 (xx18)	Home Drive
Enter: None	Exit: None. Heads on selected drive moved to Track 0. Uses: All Primary Registers

This function will position the head(s) on the selected drive to Track 0. In B/P Bios, This operation performs no useful action, and is simply a Return. Pending Write purges and head repositioning is handled by the individual device drivers (Specifically Select Drive functions).

Function 9 (xx1B)	Select Logical Drive
Enter: C = Desired Drive (A=0..P=15)	Exit: (Success) A <> 0, NZ HL = DPH Address (No Drive) A = 0, Zero (Z) HL = 0 Uses: All Primary Registers

This function selects a specified logical drive as the current drive to which disk operations refer. If the operation is successful, the Disk Parameter Header (DPH) address is returned for later determination of the unit parameters If the operation fails for any reason (non-existant drive, unknown or bad media, etc), a Zero value pointer is returned to signify that the drive cannot be accessed through the Bios.

Function 10 (xx1E)	Select Track
Enter: BC = Desired Track Number	Exit: None. Track Number saved Uses: No Registers

This function stores a specified Logical Track number for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes.

NOTE: While a 16-bit value is specified for this function, only the lower byte (8-bits) is used in most drivers.

Function 11 (xx21)	Select Sector
Enter: BC = Desired Sector Num	Exit: None. Sector Number saved Uses: No Registers

This function stores a specified Logical Sector Number for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes.

NOTE: While a 16-bit value is specified for this function, only the lower byte (8-bits) is used in all Floppy Disk and most Hard and RAM Disk drivers.

Function 12 (xx24)	Set DMA Address for Transfer
Enter: BC = Buffer Starting Addr	Exit: None. DMA Address saved Uses: No Registers

This Function stores a specified address to be used as the Source/Destination for a future disk operation. The last value stored with this function will be the one used in Disk Reads and Writes. In banked systems, the Bank selected for the transfer may be altered by Function 28.

Function 13 (xx27)	Disk Read
Enter: None	Exit: A = 0, Z if No Errors A = Non-Zero if Errors, NZ Uses: All Primary Registers

This function reads a Logical 128-byte sector from the Disk, Track and Sector set by Functions 9-11 to the address set with Function 12. On return, Register A=0 if the operation was successful, Non-Zero if Errors occurred.

Function 14 (xx2A)	Disk Write
Enter: C = 1 for immediate write C = 0 for buffered write	Exit: A = 0, Z if No Errors A = Non-Zero if Errors, NZ Uses: All Primary Registers

This function writes a logical 128-byte sector to the Disk. Track and Sector set by Functions 9-11 from the address set with Function 12. If Register C=1, an immediate write and flush of the Bios buffer is performed. If C=0, the write may be delayed due to the deblocking.

Function 15 (xx2D)	List Output Status
Enter: None	Exit: A = 0FFH, NZ if ready for Output Character A = 0, Z if Printer Busy Uses: AF

This function returns a flag indicating whether or not the printer is ready to accept a character. It uses the IOBYTE on Page 0 of the TPA Bank to determine which physical device to access.

Function 16 (xx30)	Perform Sector Translation
Enter: BC = Logical Sector Num DE = Addr of Trans Table	Exit: HL = Physical Sector Num Uses: All Primary Registers

This function translates the Logical Sector Number in register BC (Only C used at present) to a Physical Sector number using the Translation Table obtained from the DPH and addressed by DE.

This ends the strict CP/M 2.2-compliant portion of the Bios Jump Table. The next series of entry Jumps roughly follows those used in CP/M 3. but with corrections to what we perceived to be deficiencies and inconsistencies in the calling parameters and structures.

Function 17 (xx33)	Console Output Status
Enter: None	Exit: A = 0FFH, NZ if Console Ready for output char A = 0, Z if Console Busy Uses: AF

This function returns a flag indicating whether or not the Console Device selected by the IOBYTE on Page 0 of the TPA Bank is ready to accept another output character.

Function 18 (xx36)	Auxiliary Input Status
Enter: None	Exit: A = 0FFH, NZ if Aux Input has character waiting A = 0, Z if No char ready Uses: AF

This function returns a flag indicating whether or not the Auxiliary Input selected by the IOBYTE on Page 0 of the TPA Bank has a character waiting.

Function 19 (xx39)	Auxiliary Output Status
Enter: None	Exit: A = 0FFH, NZ if Aux Output Ready for output char A = 0, Z if Aux Out Busy Uses: AF

This function return a flag indicating whether or not the Auxiliary Output selected by the IOBYTE on Page 0 of the TPA Bank is ready to accept another character for output.

Function 20 (xx3C)	Return Pointer to Device Table
Enter: None	Exit: HL = Addr of Device Table Uses: HL

This function roughly corresponds to an analogous CP/M Plus function although precise bit definitions vary somewhat. The Character IO table consists of four devices: COM1, COM2, PIO, and NUL. Each has an input and output mask, data rate settings and protocol flags. Not all defined settings (e.g. ACK/NAK and XON/XOFF handshaking, etc) may be fully implemented in each version, but are available for later expansion and use.

Function 21 (xx3F)	Initialize Devices
Enter: None	Exit: None. Initialization done Uses: All Primary Registers

This function initializes Character IO settings and other functions which may be varied by a Configuration Utility. It is an extended version of the corresponding CP/M Plus function. Its primary use is to restore IO configurations, system parameters such as clock rate, wait states, etc, after alteration by programs which directly access hardware such as many modem programs and the configuration utility, BPCNFG (see 6.2).

Function 22 (xx42)	Return DPH Pointer
Enter: None	Exit: HL = Address of start of Table of DPH Pointers Uses: HL

This function returns a Pointer to a table of 16-bit pointers to Disk Parameter Headers for Drives A-P. A Null (0000H) entry means that no drive is defined at that logical position.

Function 23 (xx45)	<Reserved for Multiple Sector IO>
Enter: None	Exit: None. Uses: No Registers

This function is Reserved in the initial B/P Bios release and simply returns.

Function 24 (xx48)	Flush Deblocker
Enter: None	Exit: None. Pending Disk Writes Executed. Uses: All Primary Registers

This function writes any pending Data to disk from deblocking buffers as mentioned in Function 14 above. This function should be called in critical areas where tasks are being swapped, or media is being exchanged when it is possible that the Operating System will not detect the change.

Function 25 (xx4B)	Perform Possible Inter-Bank Move

Enter: HL = Start Source Address
DE = Start Dest Address
BC = Number Bytes to Move
	Exit: None. Data is moved
	Uses: All Primary Registers

This function moves the specified number of bytes between specified locations. For banked moves, the Source and Destination banks must have been previously specified with an XMOVE call (function 29). Note that the B/P implementation of this function reverses the use of the DE and HL register pairs from the CP/M 3 equivalent function.

Function 26 (xx4E)	Get/Set Date and Time

Enter: DE = Start of 6-byte Buff
C = 0 (to Get Date/Time)
C = 1 (to Set Date/Time)
	Exit: A = 1 of Successful
	A = 0 if Error or No Clock
	Uses: All Primary Registers

This function provides an interface to programs for a Real-Time Clock driver in the Bios. The function uses a 6-byte Date/Time string in ZSDOS format as opposed to Digital Research's format used in CP/M Plus for this function. Also, This function must conform to additional requirements of DateStamper(tm) in that on exit, register E must contain the entry contents of (DE+5) and HL must point to the entry (DE)+5. If the actual hardware implementing the clock supports 1/10 second increments, the current 1/10 second count may be returned in register D.

Function 27 (xx51)	Select Memory Bank
Enter: A = Desired Memory Bank	Exit: None. Bank is in Context in range 0..7FFFH Uses: AF

This function selects the Memory Bank specified in the A register and make it active in the address range 0-7FFFH. Since character IO may be used when a bank other than the TPA (which contains the IOBYTE) is activated with this function, the B/P Bios automatically obtains the IOBYTE from the TPA bank to insure that Character IO occurs with the desired devices.

Function 28 (xx54)	Select Memory Bank for DMA
Enter: A = Memory Bank for Disk DMA Transfers	Exit: None. Bank Number saved for later Disk IO Uses: No Registers

This function selects a memory Bank with which to perform Disk IO. Function 12 (Set DMA Transfer Address) operates in conjunction with this selection for subsequent Disk IO.

Function 29 (xx57)	Set Source & Dest Banks for Move
Enter: B = Destination Bank Num C = Source Bank Number	Exit: None. Bank Nums saved for MOVE operation Uses: No Registers

This function sets the Source and Destination Bank numbers for the next Move (Function 25). After a Move is performed, the Source and Destination Banks are automatically reset to TPA Bank values.

This marks the end of the CP/M Plus "Type" jumps and begins the unique additions to the B/P Bios table to support Banking, Direct IO and interfacing.

Function 30 (xx5A)	Return BIOS Addresses

Enter: None	Exit: A = Bios Version (Hex)
	BC = Addr of Bios Base
	DE = Addr of Bios Config
	HL = Addr of Device Table
	Uses: All Primary Registers

This function returns various pointers to internal BIOS data areas and the Bios Version Number as indicated above. The Bios Version may be used to determine currency of the system software, and will be used by various support utilities to minimize the possibility of data corruption and/or as an indicator of supported features.

The Base Address of the Bios Jump Table returned in register BC is often used to insure that the proper indexing is achieved into the B/P data structures in the event that a Bios 'shell' has been added such as when running NZCOM. While the Warm Boot jump at memory location 0000H normally points to the Bios Base+3, it is not always reliable, whereas this function will always return a true value with B/P Bios.

Registers DE and HL return pointers which are of value to programs which alter or configure various Bios parameters. The pointer to the configuration area of the Bios should be used in utilities as opposed to indexing from the start of the Bios Jump Table since additions to the Jump Table or insertion of other data will affect the Configuration Area starting address. The pointer in HL is available for use in systems which may contain more than four character IO devices. This pointer enables exchanges of devices to place desired devices in the first four positions of the table making them available for selection via the IOBYTE. After any alterations are made to the devices, a call to the Device Configuration Bios Function 21 should be made to activate the features.

FLOPPY DISK SUBFUNCTIONS.

Function 31 permits low-level access to Floppy and Hard Disks (via SCSI interface) by specifying a Driver Number and desired Function. While some hardware types do not support all of the parameters specified, particularly for Floppy Drives, this architecture supports all types, although specific systems may ignore certain functions. In this manner, for example, a single Format program supports NEC765, SMC9266, WD1770/1772/179x and other controller types with widely differing interfaces. Floppy Disk functions are accessed by entering a 1 value into Register B (Floppy Driver Number) and the desired function number in Register C, then jumping to or calling BIOS Entry jump number 31.

Function 31 (xx5D) Floppy SubFunction 0	Set Floppy Read/Write Mode

Enter: A = 0 for Double Density
FF for Single Density
B = 1 (Floppy Driver)
C = 0 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine establishes the Density mode of operation of the Floppy Disk Controller for Read and Write accesses. It assumes that SubFunctions 1 (Set Size and Motor) and 3 (Set Sector) have been called first.

Function 31 (xx5D) Floppy SubFunction 1	Set Floppy Disk & Motor Parms

Enter: A = 0 for 300 rpm (normal)
FF for 360 rpm (8"/HD)
D = FF for Motor Control,
0 if Motor always on
E = Disk Drive Size
B = 1 (Floppy Driver)
C = 1 (Subfunction #)
	Exit: None.
	
	Uses: AF

This routine establishes some of the physical parameters for a Floppy Drive. The normal 5.25" and 3.5" disk drives holding 400 or 800 kb or less rotate at 300 rpm. Many of the newer drives can increase this speed to 360 rpm which is the rate used on older 8" floppy drives. This is the speed used on the "High Density" 1.2 MB (IBM formatted) 5.25" drives. The A register is used to indicate the fastest speed capable on the specified drive. Register D is used to indicate whether the Motor is always On, or will start and stop periodically. This is normally used by the Bios to delay for a period before writing if the motor is stopped to allow the diskette to come up to speed thereby minimizing chances of data corruption. Register E is used to indicate the phys-cal media size as: 0=Hard Disk, 001B=8" Drive, 010B=5.25" Drive, and 011B=3.5". Nothing is returned from this command.

While all of these functions may not be supported on any specific computer type, the interface from using programs should always pass the necessary parameters for compatibility.

NOTE: This routine assumes that SubFunction 2 (Set Head and Drive) has been called first. Call this routine before calling Function 0 (Set Mode).

```
Function 31 (xx5D)                                   Set Head and Drive
Floppy SubFunction 2

Enter: A = Drive # (Bits 0,1)            Exit: None.
           Head # (Bit 2)
       B = 1 (Floppy Driver)             Uses: AF
       C = 2 (Subfunction #)
```

This routine is entered with register A containing the Floppy unit number coded in bits 0 and 1 (Unit 0 = 00, 1 = 01 .. 3 = 11), and the Head in Bit 2 (0 = Head 0, 1 = Head 1). Nothing is returned from this function. Call this Subfunction before most of the others to minimize problems in Floppy accesses.

```
Function 31 (xx5D)                                   Set Floppy Disk Mode
Floppy SubFunction 3

Enter: A = Physical Sector Number        Exit: None.
       D = Physical Sector Size
       E = Last Sctr # on Side           Uses: AF
       B = 1 (Floppy Driver)
       C = 3 (Subfunction #)
```

This routine establishes information needed to properly access a specified sector unambiguously with a number of different controller types. On entry, Register A contains the desired physical sector number desired. D contains the sector size where 0 = 128 byte sectors, 1 = 256 .. 3 = 1024 byte sectors, and E contains the last sector number on a side. Normally register E is unused in Western Digital controllers, but is needed with 765 and 9266 units. Nothing is returned from this subfunction.

Function 31 (xx5D) Floppy SubFunction 4	Specify Drive Times

Enter: A = Step Rate in milliSec
D = Head Unload Time in mS
E = Head Load Time in mS
B = 1 (Floppy Driver)
C = 4 (Subfunction #)
	Exit: None.
	
	Uses: AF

This subfunction set various timing values used for the physical drive selected. On entry, the A register contains the drive step rate in milliseconds. Within the Bios, this rate is rounded up to the nearest controller rate if the specified rate is not an even match. Register D should contain the desired Head Unload time in milliseconds, and E to the desired Head Load time in mS.

NOTE: With Western Digital type controllers, only the Step Rate is universally variable. In these systems, rates signaled by the Bios settings are rounded up to the closest fixed step rate such as the 2, 3, 5, or 6 milliSecond rates in the WD1772 or 6, 10, 20, or 30 milliSecond rates used in the older WD1770 and WD1795. Nothing is returned from this function.

Function 31 (xx5D) Floppy SubFunction 5	Home Disk Drive Heads
Enter: B = 1 (Floppy Driver) C = 5 (Subfunction #)	Exit: A = 0, Zero Set (Z) if Ok A <> 0, NZ if Errors Uses: AF NOTE: Subfcns 1, 2, & 4 Needed

This subfunction moves the head(s) on the selected drive to track 0 (home). Only success/failure is indicated by the value in the A register. No other registers may be altered by this function (especially BC).

NOTE: This function requires that Subfunctions 1 (Set Disk and Motor Parameters), 2 (Set Head and Drive) and 4 (Specify Drive Times) be called first in order to establish the physical characteristics of the Drive.

Function 31 (xx5D) Floppy SubFunction 6	Seek Track

Enter: A = Desired Track Number
D = 0FFH to Verify,
0 for No Verification
E = 0 for No Double-Step
<>0 for Double-Step
B = 1 (Floppy Driver)
C = 6 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	<> 0, NZ if Error
	
	Uses: AF
	
	NOTE: Subfcns 2, 3, & 4 Needed

This subfunction moves the head(s) for the selected drive to a specified track on the media. If the Double-Step flag (Register E) is set to a Non-Zero value, then the controller will issue two step pulses for every track increment or decrement which is required. After the Seek, a Read ID function will be performed to verify that the desired track was found if the Verification Flag (Register D) is set to a Non-Zero Number, preferably 0FFH. Only the AF registers may be altered by this function.

NOTE: This function requires that Subfunctions 2 (Set Head and Drive), 3 (Set Floppy Disk Mode) and 4 (Specify Drive Times) be called first in order to establish the physical characteristics of the Drive.

Function 31 (xx5D) Floppy SubFunction 7	Read Floppy Disk Sector

Enter: HL = Dest Buffer Address
B = 1 (Floppy Driver)
C = 7 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	<> 0, NZ if Error
	
	Uses: AF, HL
	
	NOTE: Subfcns 0,1,2,4 & 6 Needed

This subfunction Reads a physical sector of data from the selected drive and places it in the buffer at the specified address. It is important that an appropriately sized buffer is provided for this task. The Value in the A register will indicate the success or failure of the function as indicated in the above chart. Only the AF and HL registers may be altered by this function.

NOTE: This function requires that Subfunctions 0 (Set Read/Write Mode), 1 (Set Disk & Motor Parms), 2 (Set Head & Drive), 4 (Specify Drive Times) and 6 (Seek Track) be called first in order to establish the physical and logical characteristics of the data transfer.

Function 31 (xx5D) Floppy SubFunction 8	Write Floppy Disk Sector

Enter: HL = Source Buffer Address
B = 1 (Floppy Driver)
C = 8 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	A <> 0, NZ if Error
	
	Uses: AF,HL
	
	NOTE: Subfcns 0,1,2,4 & 6 Needed

This subfunction writes data from the buffer beginning at the specified address to the track, sector and head selected by other subfunctions. The value in the A register along with the setting of the Zero Flag will indicate whether the operation succeeded or not. Only the AF and HL registers may be altered by this function.

NOTE: This function requires that Subfunctions 0 (Set Read/Write Mode), 1 (Set Disk & Motor Parms), 2 (Set Head & Drive), 4 (Specify Drive Times) and 6 (Seek Track) be called first in order to establish the physical and logical characteristics of the data transfer.

Function 31 (xx5D) Floppy SubFunction 9	Read Disk Sector ID
Enter: B = 1 (Floppy Driver) C = 9 (Subfunction #)	Exit: A = 0, Zero Set (Z) if Ok A <> 0, NZ if Error Uses: AF NOTE: Subfcns 0 & 2 Needed

This Subfunction reads the first correct ID information encountered on a track. There are no entry parameters for this function other than the Driver and Subfunction number. A flag is returned indicating whether or not errors occurred. An error indicates that no recognizable Sector ID could be read on the disk. In most cases, this is due to an incorrect Density setting in the Bios.

NOTE: This function requires that Subfunctions 2 (Set Head & Drive) and 3 (Set Floppy Disk Mode) are called first in order to establish the physical characteristics of the disk.

Function 31 (xx5D) Floppy SubFunction 10	Return Floppy Drive Status
Enter: B = 1 (Floppy Driver) C = 10 (Subfunction #)	Exit: A = Status Byte of last Opn BC = FDC Controller Type HL = Address of Status Byte Uses: AF, BC, HL NOTE: Subfcn 2 Needed

This function returns the status of the currently-selected drive. There are no entry parameters for this function other than the Floppy Driver and Function number. On exit, the raw unmasked status byte of the drive, or the last operation depending on the controller type, is returned along with a binary number representing the FDC controller type (e.g. 765, 9266, 1772, etc).

NOTE: This routine assumes that Subfunction 2 (Set Head & Drive) has been called before this routine to select the Physical Parameters.

Function 31 (xx5D) Floppy SubFunction 11	Format Floppy Disk Track

Enter: HL = Pointer to Data Block
D = # of Sectors/Track
E = # of Bytes in Gap 3
B = 1 (Floppy Driver)
C = 11 (Subfunction #)
	Exit: A = 0, Zero Set (Z) if Ok
	A <> 0, NZ if Error
	
	Uses: AF,BC,DE,HL
	NOTE: Use Subfcn 10 for Cont Type

This Sub function formats a complete track on one side of a Floppy Disk. It assumes that the Mode, Head/Drive, Track, and Sector have already been set. On entry, HL points to data required by the controller to format a track. This varies between controllers, so RETDST should be called to determine controller type before setting up data structures. On entry, D must also contain the number of Sectors per Track, and E must contain the number of bytes to use for Gap 3 in the floppy format. On exit, A=0 and the Zero flag is Set (Z) if the operation was satisfactorily completed. A <> 0 and the Zero flag cleared (NZ) if errors occurred. This routine may alter all primary registers (AF,BC,DE,HL).

NOTE: This routine assumes that Subfunction 10 (Return Floppy Drive Status) has been called first to determine the Controller type and insert the correct information in the Format Data Block.

HARD DISK SUBFUNCTIONS.

These functions are available to directly access Hard Drives connected by a SCSI type interface. They are accessed by loading the desired function number in the C register, loading a 2 (SCSI driver) into the B register and calling or jumping to Jump number 31 in the Bios entry jump table. Since this interface is not as standardized as Floppy functions in order to handle SASI as well as SCSI devices, the interface has only basic functions with the precise operations specified by the User in the Command Descriptor Block passed with Function 2. While this places a greater burden on User programs, it allows more flexibility to take advantage of changing features in the newer SCSI drives.

Function 31 (xx5D) Hard Disk SubFunction 0	Set Hard Disk Addresses

Enter: DE = Address of Data Area
B = 2 (Hard Disk Driver)
C = 0 (Subfunction #)
	Exit: A = # Bytes in Comnd Block
	
	Uses: AF

This Subfunction sets the User Data Area Address for Direct SCSI IO, and returns the number of bytes available in the SCSI Command Descriptor Block. The Data Area must be AT LEAST 512 bytes long and is used to store data to be written, and to receive data read from the selected drive. This Data Area size is mandatory since 512 bytes are always returned from a direct access in order to handle the wide variety of controller types recognized in the B/P Bios drivers. The number of bytes available in the Command Descriptor Block within the physical driver is usually 10 in order to handle the extended SCSI commands, but may be scaled back to 6 in limited applications.

Function 31 (xx5D) Hard Disk SubFunction 1	Set Physical & Logical Drive

Enter: A = Device Byte (5.2.1)
B = 2 (Hard Disk Driver)
C = 1 (Subfunction #)
	Exit: A = Physical Device Bit
	
	Uses: AF

This Subfunction sets the Physical Device bit in the Bios for SCSI accesses and the Logical Unit Number in the SCSI Command Block (Byte 1, bits 7-5). The format of the Device Byte provided to this routine is defined in the Configuration Data, Section 5.2.1, CONFIG+61, and is available from the Extended Disk Parameter Header DPH-1. On exiting this routine, a byte is returned with a "One" bit in the proper position (Bit 7 = Device 7...Bit 0 = Device 0) to select the desired unit via a SCSI command.

Function 31 (xx5D) Hard Disk SubFunction 2	Direct SCSI Driver

Enter: DE = Ptr to Comnd Desc Blk
A = 0 if No Write Data
A = FF if Data to Write
B = 2 (Hard Disk Driver)
C = 2 (Subfunction #)
	Exit: A = Bit1 Status, Flags Set
	H = Message Byte Value
	L = Status Byte Value
	
	Uses: AF,BC,DE,HL
	NOTE: Subfcns 0 & 1 Needed

This Subfunction performs the actions required by the command in the specified Command Descriptor Block. The flag provided in Register A signifies whether or not user data is to be written by this command. If set to a Non-Zero value. Data from the area specified with Function 0 will be positioned for SCSI Write operations. At the end of the routine, 512 bytes are always transferred from the Bios IO Buffer to the Users Space set by Subfunction 0. This may be inefficient, but was the only way we could accommodate the wide variety of different SASI/SCSI controllers within reasonable code constraints. The status returned at completion of this function is the Status byte masked with the Check Bit, Bit 1. The full Status Byte and Message Byte from SCSI operations are also provided for more definition of any errors.

NOTE: This routine assumes that the Command Descriptor Block has been properly configured for the type of Hard Disk Controller set in B/P Bios, and that the selected disk is properly described (if necessary) in the Bios Unit definitions. Errors in phasing result in program exit and Warm Boot. It assumes the user has called Functions 0 (Set Hard Disk Addresses) and 1 (Set Physical & Logical Drives) before using this Subfunction.

Function 32 (xx60)	Set Bank for Far Jump/Call
Enter: A = Desired Bank Number	Exit: None. Uses: No Registers

This Function sets the bank number for a later Function 33 Jump to a routine in an alternate Memory Bank.

Function 33 (xx63)	Jump to (HL) in Alternate Bank
Enter: HL = Address to execute in Bank set w/Fn 32	Exit: <Unknown> Called routine sets return status Uses: All Primary Regs (assumed)

This Function switches to the bank number previously specified with Function 32, then calls the routine addressed by HL. Upon completion. operation returns to the bank from which called, and the address on the top of the stack.

Function 34 (xx66)	Clear Stack Switcher
Enter: HL = Addr to resume exec in entry bank	Exit: None. Execution resumes at addr in HL in entry bank Uses: No Registers

This Function is used for error exits from banked routines to return to the entry bank.

Function 35 (xx69)	Load A,(HL) from Alternate Bank
Enter: HL = Addr of desired byte C = Desired Bank Number	Exit: A = Byte from C:HL Uses: AF

This Function gets a byte (8-bits) from the specified Bank and Address. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 36 (xx6C)	Load DE,(HL) from Alternate Bank
Enter: HL = Addr of desired word C = Desired Bank Number	Exit: DE = Word from C:HL Uses: AF,DE

This Function gets a Word (16-bits) from the specified Bank and Address. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 37 (xx6F)	Load (HL),A to Alternate Bank
Enter: HL = Addr of Dest Byte C = Desired Bank Number A = Byte to save at C:HL	Exit: None. Byte stored at C:HL Uses: AF

This Function saves a Byte (8-bits) to the specified Address and Bank. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 38 (xx72)	Load (HL),DE to Alternate Bank

Enter: DE = Word to store at C:HL
HL = Addr of Dest Word
C = Desired Bank Number
	Exit: None. Word stored at C:HL
	
	Uses: AF

This Function saves a Word (16-bits) to the specified Address and Bank. The bank is temporarily switched in context for the access (if required), then restored to entry conditions. Interrupts are temporarily disabled during the brief access time.

Function 39 (xx75)	Return Current Bank in Context
Enter: None	Exit: A = Bank currently active in Addr 0..7FFFH Uses: AF

This Function returns the Memory Bank currently in Context in the address range of 0..7FFFH. It may be used in the "where am I" role in application programs to track memory accesses.

5.2 Bios Data Structures.

5.2.1 Configuration Area.

Much of the ability to tailor B/P Bioses to your specific operating needs is due to the standardized location of many discrete elements of data. and a facility to easily locate and change them, regardless of the particular hardware platform in operation. Bios Function 30, Return Bios Addresses, reports the base address of the Configuration Area in the DE register pair. In this section, we will review each of the specified elements, their functions, and which parts of the data must be rigidly controlled to insure that the supplied utilities continue to function. as well as guarantee the portability of other programs.

CONFIG-6 - Bios ID. **Character String, 6 bytes.**
This character string MUST begin with the three characters "B/P" in Uppercase Ascii. followed by three Version-specific identifying characters. As of March 1993, the following identifiers have been assigned to systems:

"B/P-YS"	YASBEC
"B/P-AM"	Ampro Little Board 100
"B/P-18"	MicroMint SB-180
"B/P-CT"	Compu/Time S100 Board Set
"B/P-TT"	Teletek
"B/P-XL"	Intelligent Computer Designs XL-M180

CONFIG+0 - IOBYTE. **Byte.**
This byte contains the initial definition of the byte placed at offset 3 on the Base Page (0003H) during a Cold Boot and determines which of the four defined character IO devices will be used as the Console. Auxiliary and Printer devices. The default setting may be altered by BPCNFG to reflect changed device configurations. or by reassembly of the Bios.
The bit definitions in this byte are:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | | └─┴── Console Device
    | | | | └─┴────── Auxiliary Input Device
    | | └─┴────────── Auxiliary Output Device
    └─┴────────────── Printer Device
```

CONFIG+1 - System Drive. **Byte.**
This byte contains the drive which will be accessed after a Cold Boot and is assumed to contain the Operating System files. It is a binary value where A = 0. B = 1...P = 15.

CONFIG+2 - Bios Option Flags. Byte.
This byte consists of individually mapped bits which display which options are active in the assembled Bios. The bits listed as <reserved> should not be defined without prior coordination with the system developers to preclude conflicts with planned enhancements. The byte is currently defined as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | | | |
    | | | | | | | +---- 0 = Unbanked Bios      1 = Banked Bios
    | | | | | | +------ 0 = Bank in RAM        1 = Bank in ROM
    | | | | | +-------- 0 = DPBs Fixed         1 = DPBs Assignable
    | | | | +---------- 0 = ALV/CSV in TPA     1 = ALV/CSV in Bank (ZSDOS2)
    +-+-+-+------------  <reserved>
```

The next five bytes define the memory map of a banked system in 32k slices. For a complete description of Bank allocations, please refer to Section 4. In non-banked systems, all except the RAM Drive Bank should all be set to 0. If no memory is available for re-assignment as a RAM drive, this byte as well should be set to 0.

CONFIG+3 - User Bank. Byte.
This Byte reflects the Bank number reserved for User Applications.

CONFIG+4 - TPA Bank. Byte.
This Byte reflects the Bank number reserved for the Transient Program Area in the address range of 0..7FFFH. The next sequential bank number is normally the Common Bank which always remains in Context in the addressing range of 8000..FFFFH and contains the Operating System, Bios and Z-System tables.

CONFIG+5 - SYStem Bank. Byte.
This byte reflects the Bank number containing any executable code and data specified for the System Bank.

CONFIG+6 - RAM Drive Bank. Byte.
This byte reflects the starting Bank number available for use as a RAM Drive. It is assumed that all RAM from this Bank through the Maximum Bank Number is contiguous and available as a RAM Drive.

CONFIG+7 - Maximum Bank Number. Byte.
This byte reflects the number of the last available bank of RAM in the system. In many systems, it may be set to different numbers depending on the number of RAM chips installed in the system.

CONFIG+8 - Common Page Base. Byte.
This byte reflects the Base Page of the Common area in systems which do not fully comply with the 32k Memory Banking architecture of B/P Bios, but can be made somewhat compliant. This Byte must be AT LEAST 80H, but may be higher if needed.

CONFIG+9 - DPB Size. Byte.
This byte contains the length of Disk Parameter Block allocations within the Bios. Since more information is needed than the 15 bytes defined by Digital

Research in CP/M 2.2, an extended format is used. All re-assignments of Disk Parameter data should use this byte to determine the size of records.

CONFIG+10 - Number of DPBs in Common RAM. Byte.
CONFIG+11 - Number of DPBs in System Bank.. Byte.
These two bytes indicate the complete complement of Floppy Disk formats available within the Bios. In most cases, one of these two bytes will reflect a zero value with all Disk Parameter Blocks resident either in the Common area or in the System Bank. The provisions are available, however with these two bytes to split the definitions for custom versions without voiding the support tools provided.

CONFIG+12 - Pointer to first Common DPB. Word.
CONFIG+14 - Pointer to first Banked DPB. Word.
These two words point to the first DPB in a sequential list within the respective memory banks for Disk Parameter Blocks defined in the preceding bytes. In most cases one of these two words will be a Null pointer (0000H) corresponding to no data as described in the count bytes above.

CONFIG+16 - Initial Startup Command. String.
This string contains the first command which will be initiated on a Cold Boot. It is loaded into the Multiple Command Buffer defined in the Environment Descriptor (See 5.2.4) and calls a file of the specified name with a type of "COM". The string may have up to eight characters and must be Null-terminated (end with a Binary 0). The string is defined as:

Byte	-	Number of Characters (0..8)
String	-	8 bytes for Ascii characters (usually Uppercase)
Byte	-	Terminating Null (binary 0)

CONFIG+26 - Pointer to Environment Descriptor. Word.
This Word points to the first byte of an extended Z34 Environment which MUST begin on a Page boundary (xx00H). See Section 5.3.1 for a complete description of the Environment Descriptor and B/P Bios unique features.

CONFIG+28 - Banked User Flag/Bank Number. Byte.
This Byte may be used as a flag to indicate whether or not a User Bank is defined. Bank 0 cannot be used as a User bank by decree of the system authors. Therefore, if this byte contains a binary 0, no User Bank is available.

CONFIG+29 - Pointer to Start of Banked User Area. Word.
This word contains the address of the first available byte in the Banked User Area, if one exists. Routines loaded into the User Bank should contain a standard RSX header structure to link sequential programs and provide a primitive memory management function.

CONFIG+31 - CPU Clock Rate in Megahertz. Byte.
This byte must contain the processor speed rounded to the nearest Megahertz. It may be used by software timing loops in application and utility programs to adapt to the clock speed of the host computer and provide an approximate time.

This byte is reflected in the Environment Descriptor (see 5.2.4) as well for programs which are Z-System "aware".

CONFIG+32 - Additional Wait State Requirements. Byte.
This byte is "nibble-mapped" to reflect the number of wait states needed for memory and IO accesses when these functions can be set via software. In the Z80/Z180, IO port accesses have one wait state inserted within the processor. This byte does not account for this fact, and reflects wait states IN ADDITION TO any which are built into the hardware. For older processors such as the Z80, these bytes normally have no effect since additional wait states must be added with hardware.

CONFIG+33 - Timer Reload Value. Word.
In many systems, Interrupts or Timer values are set by software-configurable countdown timers. the 16-bit value at this location is reserved for setting the timer value and may be "fine tuned" to allow the system to maintain correct time in the presence of clock frequencies which may deviate from precise frequencies needed for accurate clocks.

CONFIG+35 - Floppy Disk Physical Parameters. Table.
This table consists of four 5-byte entries which contain information on up to four physical drives. Each entry is defined as:

Byte 0 - Provides base for XDPH byte. Bit mapped as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | |_|_|__ Disk Size 000=Fixed Disk, 001=8", 010=5.25", 011=3.5"
    | | | | |________ 0 = Single-Sided,       1 = Double-Sided
    | | | |__________  <reserved>
    | | |____________ 0 = Motor Always On     1 = Motor Control Needed
    | |______________ 0 = 300 RPM Max Speed   1 = 360 RPM (8" & HD)
    |________________  <reserved>
```

Byte 1 - Step Rate in milliseconds.
Byte 2 - Head Load Time in milliseconds.
Byte 3 - Head Unload Time in milliseconds.
Byte 4 - Number of Tracks (Cylinders) on drive.

Those bits in Byte 0 which are listed as unused must be set to 0 since this byte provides the initial value stored in the XDPH when assignable drives are used. For controllers which do not need the available information (e.g. Western Digital controllers do not need Byte 3), these values may be set to any arbitrary value, but MUST remain present in the structure to prevent changing subsequent addresses.

CONFIG+55 - Motor On Time in 1/10th Seconds. Byte.
This time may be used in some types of Floppy Disk controllers to keep the drive motors spinning for a specified time after the last access to avoid delays in bringing the spindle up to speed. Some controllers, notably the Western Digital 17xx and 19xx series to not support this feature. In this case, the byte may be set to any arbitrary value, but MUST remain present.

CONFIG+56 - Motor Spinup Time in 1/10th Seconds. Byte.
This time is the delay which will be imposed by the Bios before attempting to access a Floppy Disk drive when it senses that the motor is in a stopped condition. Providing such a delay will minimize the probability of data corruption by writing to disk which is rotating at the incorrect speed.

CONFIG+57 - Maximum Number of Retries. Byte.
This byte specifies the number of attempts which will be made on a Floppy Disk access before returning an error code. In some cases, such as diagnostic programs, it may be desirable to set this value to 1 to identify soft errors, or ones which fail on the first attempt, but succeed on a subsequent try. We recommend a value of 3 or 4 based on our experience. Larger values may result in inordinately long delays when errors are detected.

CONFIG+58 - Pointer to Interrupt Vector Table. Word.
This Word contains the address of the base of an Interrupt Vector Table which, when used, contains pointers to service routines. The precise definition of the table is not standardized and may vary considerably between systems. This pointer serves only to provide an easy and standardized method of locating the table for re-definition of services or system features.

CONFIG+60 - SCSI Controller Type. Byte.
To accommodate the widest variety of different controllers including the older SASI models, this byte is defined as containing a byte code to the specific model being used. In most cases, this byte has little if any effect within the Bios, but may have significant effects on Hard Disk Diagnostic programs, or User-developed utilities. Any additions to this table should be coordinated with the authors to insure that the standard support utilities continue to function. Current definitions are:

0 - Owl
1 - Adaptec ACB-4000A
2 - Xebec 1410A/Shugart 1610-3 (SASI)
3 - Seagate SCSI
4 - Shugart 1610-4 (Minimal SCSI subset)
5 - SCSI-2 (Newer Conner, Quantum and Maxtor drives)

CONFIG+61 - Hard Drive Physical Parameters. **Table.**
This table consists of three 9-byte entries defining up to three physical Hard Drives. While the SCSI definition allows for more units, three was considered adequate for most systems. If additional drives are needed, please contact the authors for methods of including them without invalidating any of the standard utilities or interfaces. Each of the three entries is defined as:

Byte - Physical and Logical Address as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | | └─┴─┴── Physical Device (000-110B, 111B reserved for Host)
    | | | | └──────── <reserved>
    | | | └────────── 0 = Drive NOT Present, 1 = Drive Present
    └─┴─┴──────────── Logical Unit Number (000-111B) for controllers
                        capable of handling multiple drives
```

Word - Number of Physical Cylinders on Drive
Byte - Number of Usable Physical Heads on Drive
Word - Cylinder Number to begin Reduced Write Current
Word - Cylinder Number to begin Write Precompensation
Byte - Step Rate. This byte may either be an absolute rate in mS or a code based on controller-specific definitions

For many of the newer controllers, the last three items may not have any meaning in which case they can be set to any arbitrary value. Also in newer drives, the physical characteristics such as the number of cylinders and heads may be hidden within the drive electronics with re-mapped values provided to the controller via various SCSI commands. As with the last three entries, in this case, they may be set to any arbitrary value.

CONFIG+88 (Reserved Bytes). Five Bytes are reserved for future expansion.

CONFIG+93 - Character Device Definitions. **Table.**
This table consists of four or more 16-byte (8-byte in B/P versions prior to 1.1) entries and must be terminated by a Null (binary Zero) byte. Each entry defines the name and characteristics of a character device in the system. The first four of these are directly available for selection by the IOBYTE as the Console, Auxiliary IO and Printer. Other entries may be defined and exchanged with the first four to make them accessible to the system. The entries are defined as:

String - Four Ascii character Name as: COM1, PIO1, NULL, etc.

Byte - Data Rate capabilities as:

```
Bit 7 6 5 4 3 2 1 0
    | | | | └─┴─┴─┴── Current Data Rate Setting
    └─┴─┴─┴────────── Maximum Rate Available (Bits-per-Second) as:
```

0000 = None	0001 = 134.5	0010 = 50	0011 = 75
0100 = 150	0101 = 300	0110 = 600	0111 = 1200
1000 = 2400	1001 = 4800	1010 = 9600	1011 = 19200
1100 = 38400	1101 = 76800	1110 = 115200	1111 = Fixed

Byte - Configuration Byte defined as:

```
Bit 7 6 5 4 3 2 1 0
    │ │ │ │ │ │ │ └── 0 = 2 Stop Bits,        1 = 1 Stop Bit
    │ │ │ │ │ │ └──── 0 = No Parity,          1 = Parity Enabled
    │ │ │ │ │ └────── 0 = Odd Parity,         1 = Even Parity
    │ │ │ │ └──────── 0 = 8-bit Data,         1 = 7-bit Data
    │ │ │ └────────── 0 = No XON/XOFF,        1 = XON/XOFF Control Enabled
    │ │ └──────────── 0 = No CTS/RTS,         1 = CTS/RTS Control Enabled
    │ └────────────── 0 = Device NOT Input,   1 = Device can be read
    └──────────────── 0 = Device NOT Output,  1 = Can Write Device
```

Byte - Input Data Mask. Bit-mapped byte used to logically AND with bytes read from Device Input.

Byte - Output Data Mask. Bit-mapped byte used to logically AND with bytes before being output to device.
Word - Pointer to Character Output routine.
Word - Pointer to Output Status routine.
Word - Pointer to Character Input routine.
Word - Pointer to Input Status routine.

NOTE: The last four pointers are not at these locations in B/P Bios versions prior to 1.1, but were accessed by a pointer returned by Bios Function 30.

5.2.2 Disk Parameter Header.

The Disk Parameter Header (DPH) is a logical data structure required for each disk drive in a CP/M compatible Disk Operating System. It consists of a series of eight pointers which contain addresses of other items needed by the DOS as well as some scratchpad space. The Address of the DPH associated with a given drive is returned by the Bios after a successful selection with Bios Function 9. If Errors occur during selection, or the drive does not exist, a Null Pointer (0000H) is returned.

For B/P Bios, it was necessary to add an additional four bytes to each DPH which contain additional information on physical and logical parameters as well as flag information. These additional bytes are referred to as the Extended DPH, or XDPH. While similar in concept to the extension added to CP/M 3, the implementation is different. The XDPH prepends the DPH and may be accessed by decrementing the returned address. As a convention, DPHs in B/P Bios source code have reserved certain label sequences for specific types of units with DPH00-DPH49 used for Floppy Drives, DPH50-DPH89 for Hard Drive Partitions and DPH90-DPH99 for RAM Drives.

An entire DPH/XDPH block is required for each logical drive in a B/P Bios system. While some pointers, such as the pointer to the Directory Buffer, may be common across a number of drives, for most systems, the other items will point to unique areas.

The DPH/XDPH elements as indexed from the DPH addresses accessible to application programs are:

DPH-4 - Format Lock Flag. Byte.

A Zero value indicates that the format of the disk is not fixed, but may be changed. If the Bios was assembled with the Auto-select option, the Bios will scan a number of different formats in order to identify the disk. If a 0FFH value is placed in this byte, it indicates that the format is fixed and cannot be changed. This is normally the case for RAM and Hard disk drives, as well as for alien floppy formats which have been selected in the emulation mode. If the Auto-select option was not chosen during assembly of the Bios, all Floppy Disk drives will also have a 0FFH byte in this position showing that the formats cannot be changed.

DPH-3 - Disk Drive Type. Byte.

This byte is bit mapped and contains flags indicating many parameters of the drive. For Floppy Drives, this byte contains a copy of the first byte in the Physical Drive Table (See 5.2.1, CONFIG+35) with the two reserved bytes set during the drive selection process. The byte is then defined as:

```
Drive Type Byte
Bit 7 6 5 4 3 2 1 0
    | | | | | |_|_|___ Disk Size  000=Fixed Disk, 001=8", 010=5.25", 011=3.5"
    | | | | |_________ 0 = Single Sided              1 = Double Sided
    | | | |___________ 0 = Single Step Drive         1 = Double Step Drive
    | | |_____________ 0 = Motor Always On           1 = Drive Motor Control Needed
    | |_______________ 0 = Max Speed 5.25" (300 rpm) 1 = 8" & HD Max Speed (360 rpm)
    |_________________ 0 = Double Density            1 = Single Density
```

For Hard Disk Partitions and the RAM Drive, this byte is not used and is set to all Zeros indicating a Fixed Drive type.

DPH-2 - Driver ID Number. Byte.

Three Driver Types are used in the basic B/P Bios configuration. A Zero value indicates a Non-existent driver, with other values used to direct disk accesses to the respective code appropriate to the device. Basic defined driver types exist for Floppy Disk (1), Hard Disk via the SCSI interface (2), and RAM Disk (3). If you wish to extend this table to include tailored drivers, please consult with the authors to preclude possible conflicts with planned extensions.

DPH-1 - Physical Drive/Unit Number. Byte.

This byte contains the Physical Drive or Unit Number hosting the logical drive. For Floppy Drives, this will usually be in the range of 0 to 3 for four drives. Hard drives may have several DPHs sharing the same physical drive number, while this field is ignored in the single RAM drive supported in the distribution B/P Bios version.

NOTE: The Physical Drive Number byte for Hard Drives is comprised of two fields to ease handling of SCSI devices. Up to seven devices (device 111B is reserved for the Host Computer) each having up to 8 Logical Units may be defined. The Byte is configured as:

Physical Drive Number

```
Bit 7 6 5 4 3 2 1 0
    | | | | | └─┴─┴── Physical Device (000-110B, 111B reserved for Host)
    | | | | └──────── <reserved>
    | | | └────────── 0 = Unit Not Available, 1 = Unit Active
    └─┴─┴──────────── Logical Unit Number (000-111B)
```

DPH+0/1 - Skew Table Pointer. Word.

This word contains a pointer to the Skew table indicator. It rarely is used for Hard and RAM drives, but is required in Floppy Disk drives. If the Bios was assembled using the Calculated Skew option, the address is of a Byte whose absolute value indicates the numerical value of skew (normally in the range of 1 to 6) used for disk accesses. This term is often replaced with Interleave, and is synonymous for this purpose. If the value of the byte is negative, it means that the sectors are recorded in a skewed form on the disk and that Reads and Writes should be sequential. If the value is positive, then an algorithm is called to compute a physical sector number based on the desired logical sector and the skew factor. For systems assembled without Calculated skew, this word points to a table of up to 26 bytes which must be indexed with the desired Physical Sector number (0..Maximum Sector Number) to obtain the Corresponding Disk Sector number.

DPH+2 - Dos Scratch Words. 3 Words.

These three words are available for the Dos to use as it requires. No fixed values are assigned, nor are meanings for the data stored there of any value.

DPH+8/9 - Directory Buffer Pointer. Word.

This word points to a 128-byte Data area that is used for Directory searches. It is usually a common area to all DPH's in a system and is frequently updated by the Dos in normal use.

DPH+10/11 - DPB Pointer. Word.

This word points to another data structure which details many of the logical parameters of the selected drive or partition. Its structure is detailed in Section 5.2.3 below. Drives of the same type and logical configuration may share DPB definitions, so it is not uncommon to find the DPB pointers in different DPH structures pointing to the same area.

DPH+12/13 - Disk Checksum Buffer. Word.

This word points to a scratch RAM buffer area for removable-media drives used to detect disk changes. Normally this feature is used only for Floppy Disk Drives, and is disabled by containing a Zero word (0000H) for Hard and RAM drives. For Floppy Drives, a RAM area with one byte for every four directory entries (128-byte sector) is needed (See 5.2.3, DPH+11/12). This scratch area cannot be shared among drives.

It should be noted that in a fully Banked B/P Bios system with ZSDOS2, the Checksum Buffer is placed in the System Bank and not directly accessible by applications programs.

DPH+14/15 - Allocation Vector (ALV) Buffer. **Word.**
This word points to a bit-mapped buffer containing one bit for each allocation block on the subject drive (See 5.2.3, DPB+5/6). A "1" bit in this buffer means that the corresponding block of data on the device is already allocated to a file, while a "0" means that the block is free. This buffer is unique to each logical drive and cannot be shared among drives.

It should be noted that in a fully Banked B/P Bios system with ZSDOS2, the ALV Buffer is placed in the System Bank and not directly accessible by applications programs. Since access to the ALV buffer is frequently needed to compute free space on drives, ZSDOS2 contains an added function to return disk free space. Using this call allows applications access to the information without directly accessing the data structure.

5.2.3 Disk Parameter Block.

The Disk Parameter Block (DPB) is a data structure defined by Digital Research for CP/M which defines the logical configuration of storage on mass storage. It has been expanded in B/P Bios to include additional information to provide enhanced flexibility and capability. The expansion is referred to as the Extended DPB or XDPB, and prepends the actual DPB structure. The address of the DPB may be obtained from the DPH pointer returned by the Bios or Dos after a disk selection (See 5.2.2 above). All DPBs reside in the Common Memory area and are available to applications programs whether in a Banked or Unbanked system. For the sake of a convention, the DPBs are labeled in the same manner as DPHs with DPB00-DPB49 used for Floppy Drives, DPB50-DPB89 for Hard Drive Partitions, and DPB90-99 for RAM Drives.

The layout of the Disk Parameter Block as indexed from the available DPB pointer is:

DPB-16 - Ascii ID String. **10 Bytes.**
This string serves as an identification which may be printed by applications programs such as our BPFORMAT. This string may be a mixed alphanumeric Ascii set of up to ten characters, but the last valid character must have the Most Significant Bit (Bit 7) Set to a "1".

DPB-6 - Format Type Byte 0. **Byte.**
This byte contains some of the information about the format of the drive, and the logical sequencing of information on the physical medium. The bits in the byte have the following significance:

```
Bit  7 6 5 4 3 2 1 0
     | | | | | └─┴─┴── Disk Size:  000 = Fixed Disk, 001 = 8", 010 = 5.25", 011 = 3.5"
     | | └─┴─┴─────── Track Type
     | |                 000 = Single Side          001 = Reserved
     | |                 010 = Sel by Sec, Cont     011 = Sel by Sec, Sec # Same
     | |                 100 = S0 All, S1 All       101 = S0 All,S1 All Reverse
     | |                 110 = Sel by Trk LSB       111 = Reserved
     | └──────────────── 0 = Track 0 Side 0 is Double Density, 1 = Single Density
     └────────────────── 0 = Data Tracks are Double Density, 1 = Single Density
```

For Hard Drives and RAM Drives, this byte contains all Zero bits to signify Fixed Media and format.

DPB-5 - Format Type Byte 1. Byte.
This byte contains additional information about the format of information. The bits have the following meanings:

```
Bit  7 6 5 4 3 2 1 0
     | | | | | |_|_|____ Sector Size:  000 = 128, 001 = 256, 010 = 512, 011 = 1024
     | | |_|_|__________ Allocation Size:  000=1K, 001=2K, 010=4K, 011=8K, 100=16K
     | |                    (NOTE: This should match the definition in DPH)
     | |________________ <Reserved>
     |__________________ 0 = Normal Speed (300 rpm)
                         1 = 8" & HD Floppy (360 rpm) or Hard Drive
```

For Hard Drives. The distribution version of B/P Bios and the support utilities assume that the Sector size is always 512 bytes. The remaining bits should be set as indicated.

DPB-4 - Skew Factor. Byte.
This byte is a signed binary value indicating the skew factor to be used during Format. Read and Write. It is normally used only with Floppy Drives and usually set to -1 (0FFH) for Hard and RAM drives to indicate that Reads and Writes should be done with No skew. If the option to calculate skew is in effect during Bios assembly, the Skew pointer in the DPH (DPH+0/1) points to this byte. If a skew table is used, this byte has no effect and should be set to 80H.

DPB-3 - Starting Sector Number. Byte.
This byte contains the number of the first Physical Sector on each track. Since most Disk Operating Systems use a Zero-based sequential scheme to reference sectors, this value provides the initial offset to correct logical to physical sector numbers.

DPB-2 - Physical Sectors per Track. Byte.
This byte contains the number of Physical (as opposed to logical) Sectors on each track. For example, CP/M computes sectors based on 128-byte allocations which are used on single-density 8" Floppy Disks. One of the popular five-inch formats uses five 1K physical sectors which equates to 40 logical CP/M sectors. This byte contains 5 in this instance for the number of 1K Physical Sectors.

DPB-1 - Physical Tracks per Side. Byte.
This byte contains the number of Physical Tracks per Side, also called the Number of Cylinders. It reflects the Disk, as opposed to the Drive capabilities and is used to establish the requirements for double-stepping of Floppy Drives. In the case of a 40-track disk placed in an 80-track drive, this byte would contain 40, while the Drive parameter in the Configuration Section contains 80 as the number of tracks on the drive. This byte has no meaning for Hard Drive partitions or RAM drives and should be set to Zero, although any arbitrary value is acceptable.

DPB+0/1 - Logical Sectors per Track. Word.
This value is the number of Logical 128-byte sectors on each data track of the disk. It is equivalent to the number of Physical Sectors times the Physical Sector Size MOD 128.

DPB+2 - Block Shift Factor. Byte.
DPB+3 - Block Mask. Byte.
DPB+4 - Extent Mask. Byte.
These three bytes contain values used by the Operating System to compute Tracks and Sectors for accessing logical drives. Their values are detailed in various references on CP/M and ZSDOS programming and should not be varied without knowledge of their effects.

DPB+5/6 - Disk Size (Capacity). Word.
This Word contains the number of the last allocation block on the drive. It is the same as the capacity in allocation blocks - 1. For example, if 4k allocation blocks are being used and a 10 Megabyte drive is being defined. this word would contain 10,000,000/4000 - 1 or 2499.

DPB+7/8 - Maximum Directory Entry. Word.
This Word contains the number of the last Directory Entry and is the same as the Number of Entries - 1. For example. if 1024 directories are desired. this word would be set to 1024 - 1 = 1023.

DPB+9/10 - Allocations 0 and 1. 2 Bytes.
These two Bytes hold the initial allocations stored in the first two bytes of the ALV Buffer (See 5.2.2. DPH+14/15) during initi. drive selection. Their primary use is to indicate that the Directory Sectors are already allocated and unavailable for data storage. They are bit-mapped values and are used in Hi-byte. Lo-byte form as opposed to the normally used Lo-byte, Hi-byte storage used in Z80 type CPUs for Word storage. The bits are allocated from the MSB of the first byte thru the LSB. then MSB thru LSB of the second byte based on one bit per allocation block or fraction thereof used by the Directory. The bits may be calculated by first computing the number of entries per allocation block, then dividing the desired number of entries by this number. Any remainder requires an additional allocation bit.

For example, if 4k allocation blocks are used, each block is capable of 4096/32 bytes per entry = 128 Directory Entries. If 512 entries are desired. then 512/128 = 4 allocation blocks are needed which dictates that Allocation byte 0 would be 11110000B (0F0H) and Allocation Byte 1 would be 00000000B.

DPB+11/12 - Check Size. Word.
This Word is only used in removable media (normally only Floppy Drives) and indicates the number of sectors on which to compute checksums to detect changed disks. It should be set to 0000H for Fixed and RAM Disks to avoid the time penalty of relogging after each warm boot.

DPB+13/14 - Track Offset. Word.
This Word indicates the number of Logical Tracks to skip before the present DPB is effective. It is normally used to reserve boot tracks (usually 1 to

3), or to partition larger drives into smaller logical units by skipping tracks used for other drive definitions.

5.3.1 Environment Descriptor.

The Environment Descriptor, referred to as simply the ENV, is the heart of what is now known as *The Z-System.* The most recent additions to the system by Joe Wright and Jay Sage replaced some relatively meaningless elements in the ENV with system dependent information such as the location of the Operating System components. Consequently, the ENV is not just a feature of the ZCPR 3.4 Command Processor Replacement, but is an Operating System Resource which allows other programs such as ZCPR 3.4 to access its information.

The B/P Bios requires an ENV to be present, and uses several items of information contained in it. The banked ZSDOS2 and Z40 Command Processor Replacement use even more ENV features. A few remaining bytes have been re-defined for support to B/P Bios-based systems. To denote the definition of B/P Bios data elements, a new Type, 90H, has been reserved. Using this "Type" byte, user programs can access and take advantage of the new definitions and features.

A template for the Environment Descriptor used in B/P Bios which takes its values from the Z3BASE.LIB file included in the distribution disk is:

```
************************** C A U T I O N **************************
* If transitioning from an older operating environment which      *
* uses user-loaded Environment files such as SYS.ENV, you must    *
* either delete such load instructions or modify the ENV files    *
* to reflect the EXACT loaded system definition.                  *
* Failure to observe this fact will probably cause many           *
* undesired side effects!                                         *
*******************************************************************
```

```
; Environment Descriptor for ZCPR34

ENV:    JP      0               ; Leading jump (address is CBIOS when NZCOM)
        DEFB    'Z3ENV'         ; Environment ID
        DEFB    90H             ; Env type (>=90H means B/P Extended ENV w/User Area)
        DEFW    EXPATH          ; External path (PATH)
        DEFB    EXPATHS         ;
        DEFW    RCP             ; Resident command package (RCP)
        DEFB    RCPS            ;
        DEFW    IOP             ; Input/output package (IOP)
        DEFB    IOPS            ;
        DEFW    FCP             ; Flow command package (FCP)
        DEFB    FCPS            ;
        DEFW    Z3NDIR          ; Named directories (NDR)
        DEFB    Z3NDIRS         ;
        DEFW    Z3CL            ; Command line (CL)
        DEFB    Z3CLS           ;
        DEFW    Z3ENV           ; Environment (ENV) - Actual Starting Address
        DEFB    Z3ENVS          ;
```