EXECUTIVE™

OSBORNE

Technical Manual

Volume B: Programmer's Guide

OSBORNE
COMPUTER CORPORATION

OSBORNE EXECUTIVE TECHNICAL MANUAL
VOLUME B: PROGRAMMER'S GUIDE

ABSTRACT

This manual contains a thorough description of Digital Research's operating system, CP/M Plus. Both BDOS and BIOS functions are discussed in sufficient detail to allow a programmer to fully utilize the extensive features of this operating system.

26538 Danti Court, Hayward, CA 94545
(415) 887-8080

OSBORNE EXECUTIVE TECHNICAL MANUAL
Volume B: Programmer's Guide
Issue date: May 1984

VOLUME B: PROGRAMMER'S GUIDE

TABLE OF CONTENTS

FIGURE LIST

TABLE LIST

TABLE LIST (Cont.)

SECTION LIST ALPHABETIZED

INTRODUCTION TO CP/M PLUS

This section introduces the general features of the Osborne Executive CP/M Plus Operating System, with an emphasis on how it organizes your computer's memory. CP/M Plus is available in two versions: a version that supports bank-switched memory, and a version that runs on nonbanked systems. The Osborne Executive CP/M Plus is a banked system; CP/M Plus uses the larger memory of the banked system to provide additional functions.

CP/M Plus provides a software environment for program development and execution for the Osborne Executive computer system. It allows rapid access to data and programs through a file structure that supports dynamic allocation of space for both sequential- and random-access files.

CP/M Plus supports a maximum of 16 logical floppy or hard disks with a storage capacity of up to 512 megabytes each. The maximum file size supported is 32 megabytes.

CP/M Plus supports the bank-switched memory capabilities of the Osborne Executive, and supplies additional facilities including extended command-line editing, password protection of files, and extended error messages.

The system requires a minimum of two memory banks with 11 kilobytes of memory in bank 0 and 1.5 kilobytes in common memory, plus space for the Osborne Executive BIOS. The bank-switched system provides more user memory for application programs.

CP/M Plus resides in the file CPM3.SYS, which is loaded into memory by the system loader during system initialization. The system loader resides on the first two tracks of the Executive program diskettes. CPM3.SYS contains the distributed BDOS and the Osborne Executive BIOS.

Banked Memory Organization

CP/M Plus supports bank-switched memory to expand memory capacity beyond the usual 64K-byte address space of an eight-bit microprocessor. The figure below shows memory organization under the Osborne Executive CP/M Plus system.

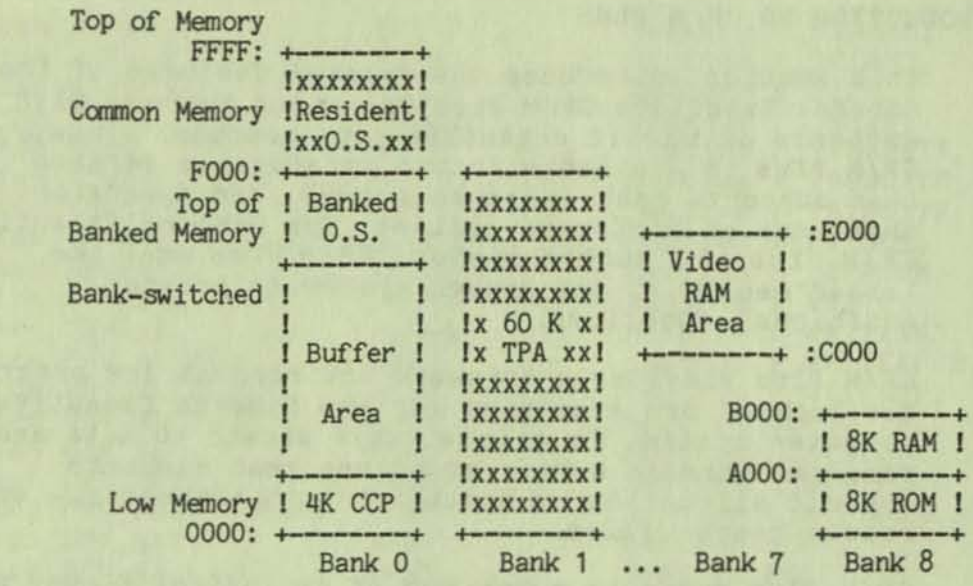


FIGURE B-1. BANKED SYSTEM MEMORY ORGANIZATION
(All addresses are expressed in hexadecimal notations.)

In the figure above, bank 0 is switched in, or in context. That is, any reference to a memory location will automatically address bank 0. The top region of memory, the common region, is always in context; that is, it can always be referenced no matter what bank is switched in. The figure below shows memory organization when bank 1 is in context.

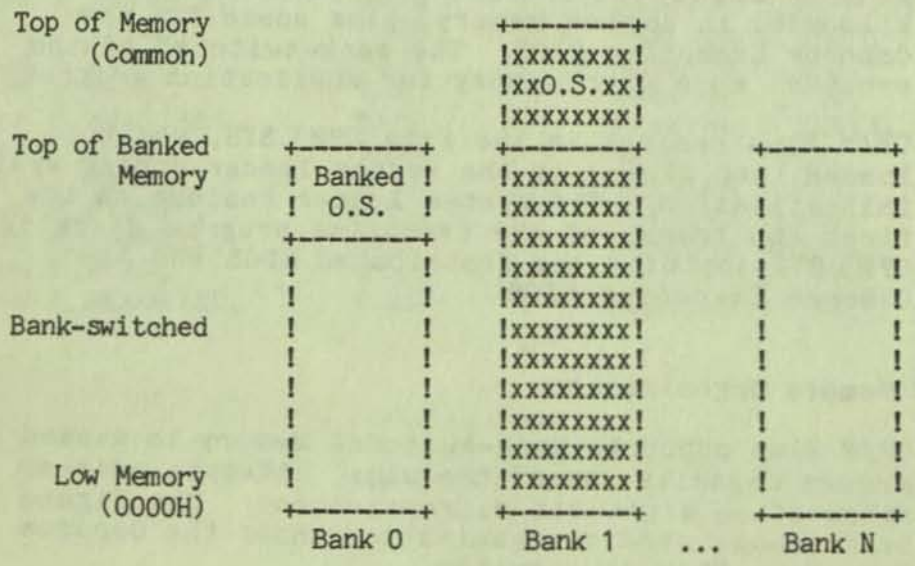


FIGURE B-2. BANKED MEMORY WITH BANK 1 IN CONTEXT

From a transient program's perspective, bank 1 is always in context. The operating system can switch to bank 0 or other banks when performing operating-system system calls without affecting the execution of the transient program. When exiting from a BDOS system-call routine, the operating system automatically switches back to the bank containing the calling program.

The operating system uses bank 0 for disk record buffers and directory hash tables. Space in the common region is allocated for transfer buffers.

In the Osborne Executive there can be a minimum of two to a maximum of seven memory banks, plus the video and ROM banks. Bank 0 is the system bank and is in context when CP/M Plus is started. Banks 1 through 6 are the transient program banks, and must be contiguous from location zero to the top of banked memory. (Banks 2 - 6 are not implemented in the current system.) This requirement does not apply to the other banks. Bank 7 is used for the video screen contents and attributes, and bank 8 contains the Boot ROM, Scratch Pad RAM, and Font RAM.

CP/M Plus always organizes memory logically so that, to a transient program in any CP/M Plus system, memory appears as shown in the figure below. The top-of-memory address is 64K-1 (0FFFFH).

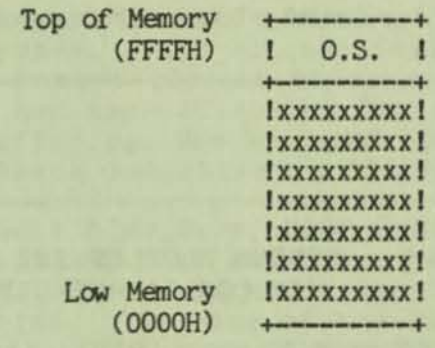


FIGURE B-3. CP/M PLUS LOGICAL MEMORY ORGANIZATION

System Components

The CP/M Plus Operating System is composed of several modules. The figure below shows where these modules reside in logical memory. Note that the figure below is just a more detailed version of the figure above.

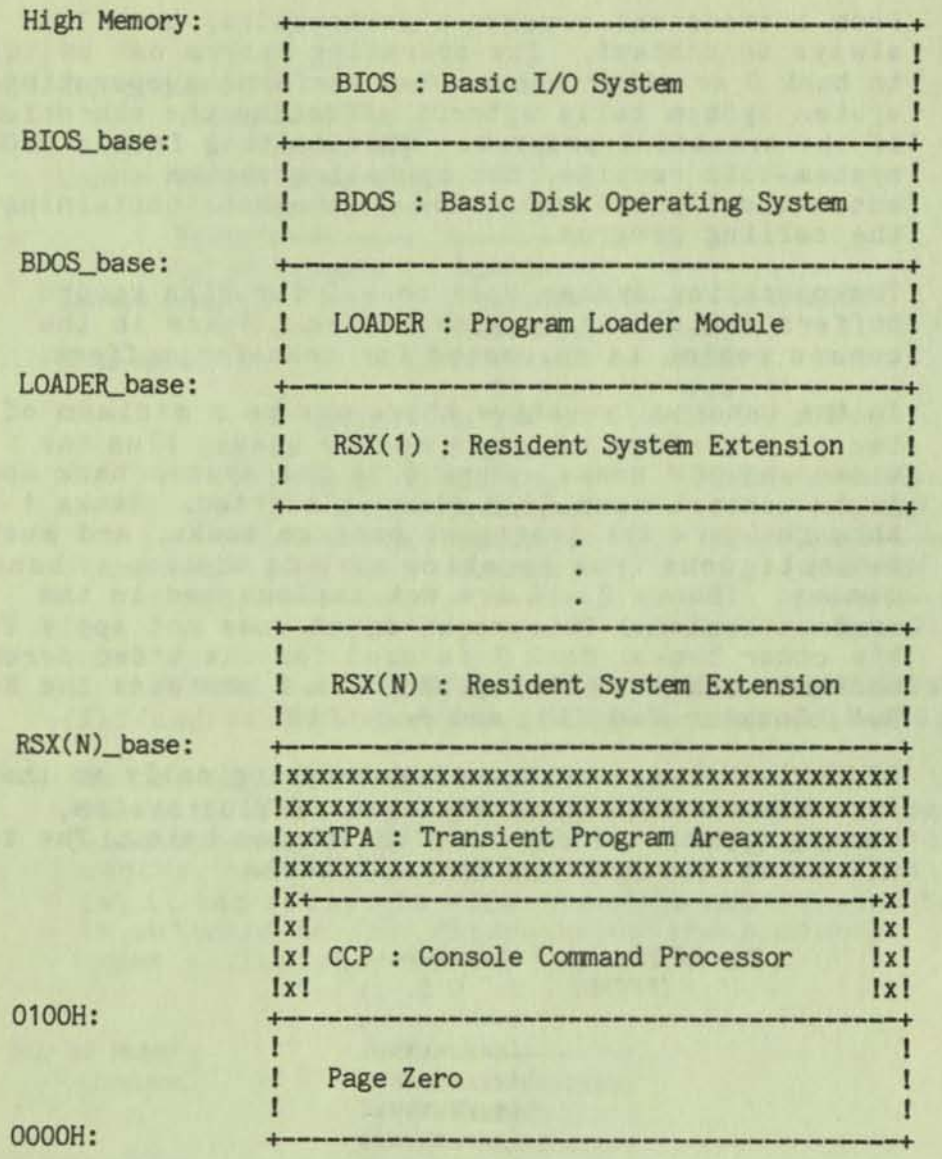


FIGURE B-4. SYSTEM MODULES AND REGIONS IN LOGICAL MEMORY

The Basic Input/Output System (BIOS) is a hardware-dependent module that defines the low-level interface to the Osborne Executive hardware. It contains the device-driving routines necessary for peripheral device I/O.

The Basic Disk Operating System, BDOS, is the hardware-independent logical nucleus of CP/M Plus. It provides a standard environment for transient programs through system calls.

The LOADER module handles program loading for the Console Command Processor and transient programs.

Usually, this module is not resident when transient programs execute. However, when it is resident, transient programs can access the module by making P_LOAD BDOS system calls.

Resident System Extensions (RSXs) are temporary, additional operating-system modules that extend or modify normal operating-system system calls. The LOADER module is always resident when RSXs are active.

The Transient Program Area (TPA) is the region of memory where transient programs execute. The CCP also executes in this region.

The Console Command Processor (CCP) is a transient system program that provides the user interface tc CP/M Plus.

The Page Zero region is an interfacing data structure to the BDOS module from the CCP and transient programs. It contains critical system parameters.

Memory Region Boundaries

Note: All memory regions in CP/M Plus begin on a page boundary. A page is defined as 256 (100H) bytes, so a page boundary always begins at an address with a low-order byte of zero.

"High memory" denotes the high address of the CP/M Plus system. This address falls below the actual top-of-memory address since space above the operating system has been allocated for directory hashing or data buffering. The maximum "high memory" address in the Osborne Executive is OEFFFH.

The labels BIOS_base, BDOS_base, and LOADER_base represent the base addresses of the operating system regions. These addresses always fall on page boundaries. The size of the BIOS region is about 1.5K bytes. The size of the BDOS region is six pages, 1.5K.

RSXs are page-aligned modules that are stacked below LOADER_base in memory. The memory ceiling of the TPA region is reduced when RSXs are active.

Under CP/M Plus the CCP is a transient program that the BIOS loads into the TPA region of memory at system cold and warm start. The BIOS also loads the LOADER module at this time. The LOADER module is contained in the CCP.

When the CCP gains control, it relocates the LOADER module just below BDOS_base. The LOADER module handles program loading for the CCP, and is three pages or .75K in size.

The maximum size of a transient program that can be loaded into the TPA is limited by LOADER_base because the LOADER cannot load a program over itself. Transient programs may extend beyond this point, however, by using memory above LOADER_base for uninitialized data areas such as I/O buffers. Programs that use memory above BDOS_base cannot make BDOS system calls.

The BDOS and BIOS

CP/M Plus achieves hardware independence through the interface between the BDOS and the BIOS sections of the operating system. This interface consists of a series of entry points in the BIOS that the BDOS calls to perform hardware-dependent primitive functions. For example, the BDOS calls the CONIN: entry point of the BIOS to read the next console input character. For a detailed description of the Osborne Executive CP/M Plus BIOS, see the **CP/M Plus BIOS** section.

Applications and the BDOS

Transient programs and the CCP access CP/M Plus facilities by making BDOS system calls. They are described in the **BDOS System Calls** section.

To make a BDOS system call, a transient program loads the CPU registers with specific entry parameters and calls location 0005H in Page Zero. If RSXs are not active in memory, location 0005H contains a jump instruction to location BDOS_base + 6. If RSXs are active, location 0005H contains a jump instruction to the RSX entry point at an address below BDOS_base.

Thus, the Page Zero interface allows programs to run without regard to where the operating system modules are located in memory. In addition, transient programs can use the address at location 0006H as a memory ceiling.

The Console Command Processor is a special system program that executes in the TPA and makes BDOS calls just like an application program. However, the CCP has a unique role: it gives the user access to operating system facilities while transient programs are not executing. It includes several built-in

commands, such as TYPE and DIR, that can be executed directly without having to be loaded from disk.

When the CCP receives control, it reads the user's command lines; distinguishes between built-in and transient commands; and when necessary, calls upon the LOADER module to load transient programs from disk into the TPA for execution.

Applications and RSXs

A Resident System eXtension (RSX) module is a temporary addition to the operating system. An RSX can extend or modify one or more operating-system system call(s).

At any one time there might be zero, one, or several RSXs active in memory. When a transient program makes a BDOS system call while RSXs are active, each RSX examines the number of the call. If the system call number matches the system call the RSX is designed to extend or modify, the RSX performs the requested function. Otherwise the RSX passes the system call request to the next RSX, if any. Non-intercepted system calls are passed to the BDOS for standard execution.

The CP/M Plus utility, GENCOM, can attach RSXs to program files. When attaching RSXs, GENCOM places a special one-page header at the beginning of the program file. The CCP reads this header, learns that a program has attached RSXs, and loads the RSXs accordingly.

The LOADER module is a special type of RSX that supports the P_LOAD system call. It is always resident when RSXs are active. To indicate that RSX support is required, a program that calls P_LOAD must have an RSX header attached by the CP/M Plus GENCOM utility, even if the program does not require other RSXs.

Disk and Drive Organization and Requirements

CP/M Plus can support up to 16 logical drives, identified by the letters A through P, with up to 512 megabytes of storage each. If the drive has adequate storage, a CP/M Plus file can be as large as 32 megabytes. A logical drive usually corresponds to a physical drive on the system, particularly for floppy disk drives. High-capacity hard disks, however, are commonly divided into multiple logical drives. The figure below illustrates the standard organization of

an Osborne Executive CP/M Plus disk.

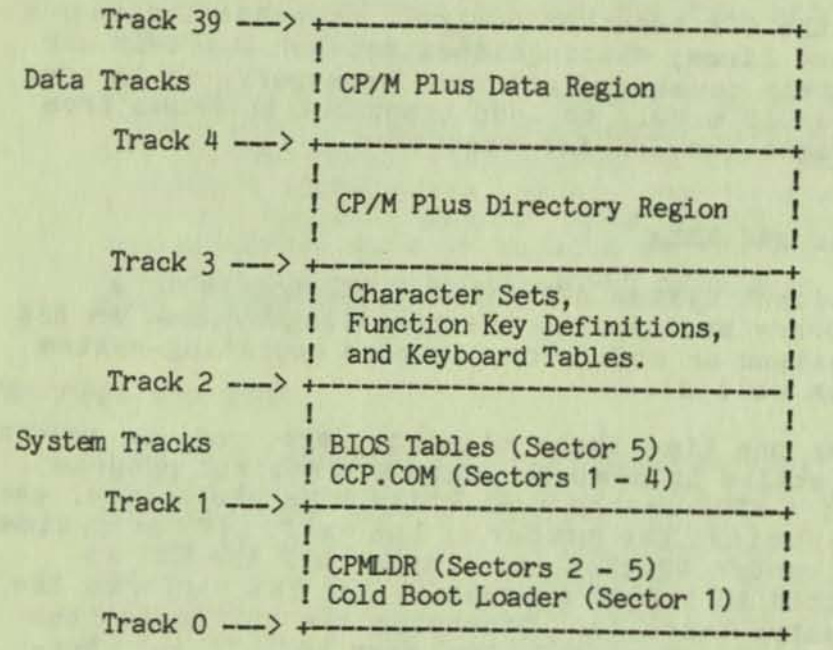


FIGURE 5. SYSTEM TRACK ORGANIZATION

The first three tracks are the system tracks. They are required only on the disk used by CP/M Plus for cold start or warm start. All subsequent CP/M Plus disk access is directed to the data tracks which CP/M Plus uses for file storage.

The data tracks are divided into a directory area and a data area. The directory area defines the files that exist on the drive and identifies the data space that belongs to each file. The data area contains the file data defined by the directory.

Cold Start Operation

The cold start procedure is executed immediately after the computer is turned on. The cold start brings CP/M Plus into memory and gives it control of the computer's resources. Cold start is a four-stage procedure.

First, ROM-based firmware loads a small program, called the Cold Boot Loader, into memory from the system tracks of drive A. It loads CPMLDR into memory from the system tracks of the system disk and passes control to it. CPMLDR reads the CPM3.SYS from the data area of the disk. The CPM3.SYS file contains the BDOS and BIOS system components and information

indicating where these modules are to reside in memory. Then CPMLDR sends a sign-on message to the console and passes control to the BIOS. These first stages of the cold boot procedure are performed with Bank 0 in context. The BIOS Cold-Start function switches Bank 1 into context before proceeding.

The final stage in the cold start procedure is performed by the P_TERMCPM system call. The entry point to this system call is located at BIOS_base as described above. It begins by performing any remaining hardware initialization, and initializing Page Zero.

It then completes the boot process by loading the autostart program EXECST.COM into the TPA region of memory and passing control to it. If there is no EXECST.COM on the disk, it will load the CCP.

When the CCP gains control, it displays the system prompt (A>). If a PROFILE.SUB file is present on drive A, the CCP executes it before prompting the user for a command.

CCP Operation

The Console Command Processor reads the user's command lines, differentiates between built-in commands and transient commands, and executes them accordingly.

When the CCP gains control following a cold start procedure, it displays the system prompt at the console. This signifies that the CCP is ready to execute a command.

It then scans the directory of the default drive for the file PROFILE.SUB. If the file exists, the CCP creates the command line SUBMIT PROFILE<cr>. Otherwise the CCP waits for the user to type a command.

The command form the CCP accepts is the standard CP/M command line, consisting of a command keyword followed by an optional command tail. The CCP converts all letters in the command line to uppercase. The following syntax defines the standard CP/M Plus command line:

<command> <command tail> <cr>

where

<command> => <filespec>

<command tail> => (no command tail) or <filespec> or <filespec><delimiter><filespec>

<filespec> => {d:}filename{.typ}{;password}

<delimiter> => one or more blanks or a tab or one of the following: "=,[]<>|"

d: => CP/M Plus drive specification, "A:" through "P:"

filename => 1-to-8-character filename

typ => 1-to-3-character filetype

password => 1-to-8-character password value

Fields enclosed in braces **{}** are optional. If there is no drive **{d:}** present in a file specification, the default drive is assumed. If the type field **{.typ}** is omitted, a type field of all blanks is assumed. Omitting the password field **{;password}** implies a password of all blanks. When a command line is entered at the console, it is terminated by a return or line-feed keystroke **<cr>**.

Transient programs that run under CP/M Plus are not restricted to the above command-tail definition. However, the CCP only parses command tails for transient programs in the standard format. Transient programs that define their command tails differently must perform their own command-tail parsing. When a transient program begins execution, the command tail is present in the default-drive DMA buffer at 0080H in the Page Zero area.

The command field must identify either a built-in command, a transient program, or a SUBMIT file. The following table summarizes the built-in commands.

TABLE B-1. BUILT-IN COMMANDS

Command	Meaning
DIR	Displays a list of all filenames from a disk directory except those marked with the SYS attribute.
DIRSYS	Displays a filename list of those files marked with the SYS attribute in the directory. May be abbreviated DIRS.

TABLE B-1. BUILT-IN COMMANDS (Cont.)

Command	Meaning
ERASE	Erases a filename from a disk directory and releases the storage occupied by the file. May be abbreviated ERA.
RENAME	Renames a file. May be abbreviated REN.
TYPE	Displays the contents of an ASCII character file at your console output device.
USER	Changes from one user number to another. May be abbreviated to the desired user number followed by a colon (4:).

Built-in commands are commands which are stored in memory so they can be executed without referencing a disk. A built-in command may have an associated command file on the disk that expands upon its options. If the CCP reads a command line and discovers the built-in command does not support the options requested, the CCP loads the built-in function's command file to perform the command. The DIR command is an example of this type of command. Simple DIR commands are supported by the DIR built-in command. More complex requests are handled by the DIR.COM utility.

The CCP assumes that all command keywords that do not identify built-in commands identify either a transient program file or a SUBMIT file. If the CCP identifies a command keyword as a transient program, the transient program file is loaded into the TPA from disk and executed. If it recognizes a SUBMIT file, the CCP reconstructs the command line into the following form:

SUBMIT <filename.SUB> <command tail> <cr>

and attempts to load and execute the SUBMIT utility. Thus, the original command field becomes the first command-tail field of the SUBMIT command. The procedure the CCP follows to parse a standard command line and execute built-in and transient commands is described as follows.

The CCP parses the command line to pick up the command field. If the command field is not preceded by a drive specification, or followed by a filetype or password field, the CCP checks to see if the command is a CCP built-in function. If the command is a built-in command, and the CCP can support the options specified in the command tail, the CCP

executes the command. Otherwise the CCP goes on to the steps described below.

At this point the CCP assumes the command field references a command file or SUBMIT file on disk. If the optional filetype field is omitted from the command, the CCP usually assumes the command field references a file of type .COM. For example, if the command field is PIP, the CCP attempts to open the file PIP.COM.

Optionally, the CP/M Plus utility SETDEF can be used to specify that a filetype of .SUB also be considered when the command filetype field is omitted. When this automatic SUBMIT option is in effect, the CCP attempts to open the command with a filetype of .COM. If the .COM file cannot be found, the CCP attempts the open operation with a filetype of .SUB. As an alternative, the order of file-open operations can be reversed, so that the CCP attempts to open with a filetype of .SUB first.

If the filetype field is present in the command, it must equal .COM, .SUB, or .PRL. A .PRL file is a Page-Relocatable file used in Digital Research's multiuser operating system, MP/M. Under CP/M Plus the CCP handles .PRL files exactly like .COM files.

When the current user number is nonzero, and a file-open request fails because the file cannot be found, the CCP attempts to locate the file under user zero. If the file exists under user zero with the SYS (system) attribute set, the file is opened from user zero. This search for a file under user zero is made by the F_OPEN BDOS system call.

If the password specified in the command field does not match the password of a file on a disk protected in Read mode, the CCP file-open operation is terminated with a password error.

If the CCP does not find the .COM or .SUB file, it echoes the command line followed by a question mark to the console. If it finds a command file with a filetype of .COM or .PRL, the CCP proceeds as described below. If it finds a SUBMIT file, it reconstructs the command line as described above and tries to load SUBMIT.COM.

When the CCP successfully opens the command file, it initializes the following Page Zero fields for access by the loaded transient program:

0050H : Drive for which the command file was loaded
0051H : Password address of first file in command tail
0053H : Password length of first file in command tail
0054H : Password address of second file in command tail
0056H : Password length of second file in command tail
005CH : Parsed FCB for first file in command tail
006CH : Parsed FCB for second file in command tail
0080H : Command tail preceded by command-tail length

Page Zero initialization is covered in more detail below.

At this point the CCP calls the LOADER module to load the command file into the TPA. The LOADER module terminates the load operation if a read error occurs, or if the available TPA space is not large enough to contain the file. If no RSXs are resident in memory, the TPA ceiling is determined by the address LOADER_base because the LOADER cannot load over itself. Otherwise the maximum TPA address is determined by the base address of the lowest RSX in memory.

Once the program is loaded, the LOADER module checks for an RSX header on the program. Programs with RSX headers are identified by a RET instruction at location 100H.

If an RSX header is present, the LOADER relocates all RSXs attached to the end of the program to the top of the TPA region of memory under the LOADER module or any other RSXs that are already resident. It also updates the address in location 0006H of Page Zero to point to the lowest RSX in memory. Finally, the LOADER discards the RSX header and relocates the program file one page lower in memory, so that the first executable instruction resides at 100H.

After initializing Page Zero, the LOADER module sets up a 32-byte stack with the return address set to location 0000H of Page Zero and jumps to location 100H. At this point the loaded transient program begins execution.

When a transient program terminates execution, the BIOS warm-start routine reloads the CCP into memory. When the CCP receives control, it tests to see if RSXs are resident in memory. If not, it relocates the LOADER module below the BDOS module at the top of the TPA region of memory. Otherwise it skips this step because the LOADER module is already resident.

The CCP execution cycle then repeats.

Note: Unlike earlier versions of CP/M, the CCP does not reset the disk system at warm start. However, the CCP does reset the disk system if a CTRL-C is typed at the prompt.

Page Zero Initialization

Page Zero is the region of memory located from 0000H to 00FFH. This region contains several instructions and data structures that are used by transient programs while running under CP/M Plus.

TABLE B-2. PAGE ZERO AREAS

Locations From	To	Contents
0000H	0002H	Contains a jump instruction to the BIOS warm-start entry point at BIOS_base + 3. The address at location 0001H can also be used to make direct BIOS calls to the BIOS console status, console input, console output, and list output primitive functions.
0003H	0004H	(Reserved)
0005H	0007H	Contains a jump instruction to the BDOS, the LOADER, or to the most recently added RSX; and serves two purposes: executing the instruction JMP 0005H provides the primary entry point to the BDOS, and LHLD 0006H places the address field of the jump instruction in the HL register pair. This value, minus one, is the highest address of memory available to the transient program.
0008H	003AH	Reserved interrupt locations for Restarts 1 - 7.
003BH	003FH	Reserved.
0040H	0041H	BIOS version number.
0042H		BIOS product code.
0043H		BIOS feature code.
0044H	0045H	ROM version number.
0046H		ROM product code.

TABLE B-2. PAGE ZERO AREAS (Cont.)

Locations From	To	Contents
0047H		ROM feature code.
0048H		Reserved. (Executive product code = 2).
0050H		Identifies the drive from which the transient program was loaded. A value of one to 16 identifies drives A through P.
0051H	0052H	Contains the address of the password field of the first command-tail operand in the default DMA buffer beginning at 0080H. The CCP sets this field to zero if no password for the first command-tail operand is specified.
0053H		Contains the length of the password field for the first command-tail operand. The CCP also sets this field to zero if no password for the first command tail is specified.
0054H	0055H	Contains the address of the password field of the second command-tail operand in the default DMA buffer beginning at 0080H. The CCP sets this field to zero if no password for the second command-tail operand is specified.
0056H		Contains the length of the password field for the second command-tail operand. The CCP also sets this field to zero if no password for the second command tail is specified.
0057H	005BH	(Not currently used - reserved).
005CH	007BH	Default File Control Block (FCB) area 1 initialized by the CCP from the first command-tail operand of the command line, if it exists.

TABLE B-2. PAGE ZERO AREAS (Cont.)

Locations From	To	Contents
006CH	007BH	Default File Control Block (FCB) area 2 initialized by the CCP from the second command-tail operand of the command line, if it exists. **Note:** This area overlays the last 16 bytes of default FCB area 1. To use the information in this area, a transient program must copy it to another location before using FCB area 1.
007CH		Current record position of default FCB area 1. This field is used with default FCB area 1 in sequential record processing.
007DH	007FH	Optional default random-record position. This field is an extension of default FCB area 1 used in random-record processing.
0080H	00FFH	Default 128-byte disk buffer. This buffer contains the command tail when the CCP loads a transient program.

The following example illustrates the initialization of the command-line fields of Page Zero. Assuming the following command line is typed at the console:

A:PROGRAM B:FILE.TYP;PASS C:FILE.TYP;PASSWORD

A hexadecimal dump of 0050H to 00A5H would show the Page Zero initialization performed by the CCP:

```
0050H: 01 8D 00 04 9D 00 08 00 00 00 00 00 02 46 49 4C .............FIL
0060H: 45 20 20 20 20 54 59 50 00 00 00 00 03 46 49 4C E....TYP.....FIL
0070H: 45 20 20 20 20 54 59 50 00 00 00 00 00 00 00 00 E....TYP........
0080H: 24 20 42 3A 46 49 4C 45 2E 54 59 50 3B 50 41 53 . B:FILE.TYP;PAS
0090H: 53 20 43 3A 46 49 4C 45 2E 54 59 50 3B 50 41 53 S C:FILE.TYP;PAS
00A0H: 53 57 4F 52 44 00
```

Transient Program Operation

As the name transient implies, transient programs are not system-resident. The CCP must load a transient program into memory every time the program is executed. Generally, an executing transient program communicates with the operating system only through BDOS system calls. Transient programs make BDOS system calls by loading the CPU registers with the

appropriate entry parameters and calling location 0005H in Page Zero.

Transient programs can use the S_BIOS BDOS system call to access BIOS entry points. This is the preferred method for accessing the BIOS; however, for compatibility with earlier releases of CP/M, transient programs can also make direct BIOS calls for console and list I/O by using the jump instruction at location 0000H in Page Zero.

A transient program can terminate execution in one of three ways: by jumping to location 0000H, by making a BDOS P_TERMCPM call, or by making a P_CHAIN call. The first two methods are equivalent; they pass control to the BIOS warm-start entry point.

The P_CHAIN call allows a transient program to specify a transient program to be executed after it terminates. This call executes a standard warm-boot sequence, but passes the command specified by the terminating program to the CCP, which executes the specified command instead of prompting the console.

Resident System Extension Operation

A Resident System Extension (RSX) is a special type of program that can be attached to the operating system to modify or extend the functionality of the BDOS. RSX modules intercept BDOS system calls and either perform them, translate them into other BDOS system calls, or pass them to the operating system. The BDOS executes nonintercepted system calls in the standard manner.

A transient program can also use the S_RSX BDOS system call to call an RSX for special functions. S_RSX is a general-purpose system call that allows customized interfaces between programs and RSXs.

Two examples of RSX applications are the GET utility and the LOADER module. The GET.COM command file has an attached RSX, GET.RSX, that intercepts all console input calls and returns characters from the file specified in the GET command line. The LOADER module is another example of an RSX, but it is unique in that it supports the P_LOAD system call. It is always resident in memory when other RSXs are active.

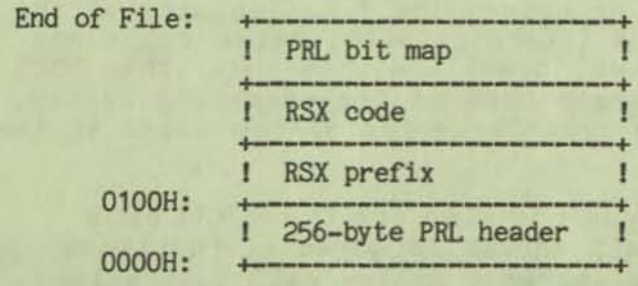
RSXs are loaded into memory at program load time. After the CCP locates a command file, it calls the LOADER module to load the program into the TPA. The LOADER loads the transient program into memory along with any attached RSXs. Subsequently, the loader

relocates each attached RSX to the top of the TPA and adjusts the TPA size by changing the jump at location 0005H in Page Zero to point to the RSX. When RSX modules reside in memory, the LOADER module resides directly below the BDOS, and the RSX modules stack downward from it.

The order in which the RSX modules are stacked affects the order in which they intercept BDOS calls. A more recently stacked RSX has precedence over an older RSX. Thus, if two RSXs in memory intercept the same BDOS system call, the more recently loaded RSX handles the system call.

The CP/M Plus utility GENCOM attaches RSX modules to program files. Program files with attached RSXs have a special one-page header that the LOADER recognizes when it loads the command file. GENCOM can also attach one or more RSXs to a null command file so that the CCP can load RSXs without having to execute a transient program. In this case the command file consists of the RSX header followed by the RSXs.

RSX modules are page-relocatable files with the filetype .RSX. RSX files must be page-relocatable because their execution address is determined dynamically by the LOADER module at load time. RSX files have the following format:



RSX files begin with a one-page PRL header that specifies the total size of the RSX prefix and code sections. The PRL bit map is a string of bits identifying those bytes in the RSX prefix and code sections that require relocation. The PRL format is described in detail in Appendix B.

Note: The PRL header and bit map are removed when an RSX is loaded into memory. They are only used by the LOADER module to load the RSX.

The RSX prefix is a standard data structure that the LOADER module uses to manage RSXs. Included in this data structure are jump instructions to the previous and next RSX in memory, and two flags. The LOADER

module initializes and updates these jump instructions to maintain the link from location six of Page Zero to the BDOS entry point. The RSX flags are the Remove flag and the Nonbanked flag. The Remove flag controls RSX removal from memory. The Nonbanked flag identifies RSXs that are loaded only in nonbanked CP/M Plus systems.

The RSX code section contains the main body of the RSX. This section always begins with code to intercept the BDOS system call that is intercepted by the RSX. This section can also include initialization and termination code that transient programs can call with the S_RSX BDOS system call.

When the CCP gains control after a system warm start, it removes any RSXs in memory that have the Remove flag set to OFFH. All other RSXs remain active in memory.

Note: If an RSX marked for removal is not the lowest active RSX in memory, it still occupies memory after removal. Although the removed RSX cannot be executed, its space is returned to the TPA only when all the lower RSXs are removed.

There is one special case where the CCP does not remove an RSX with the Remove flag set following warm start. This case occurs on warm starts following the load of an empty file with attached RSXs. This exception allows an RSX with the Remove flag set to be loaded into memory before a transient program. The transient program can then access the RSX during execution. After the transient program terminates, however, the CCP removes the RSX from the system environment. The CP/M Plus SAVE utility is an example of such an RSX.

As an example of RSX operation, here is a description of the operation of the GET utility. The GET.COM command file has an attached RSX. The LOADER moves this RSX to the top of the TPA when it loads the GET.COM command file. The GET utility performs necessary initializations which include opening the ASCII file specified in the GET command line. It also makes an S_RSX BDOS system call to initialize the GET.RSX. At this point the GET utility terminates. Subsequently, the GET.RSX intercepts all console input calls and returns characters from the file specified in the GET command line. It continues this action until it reads the end-of-file character (^Z). At this point it sets its Remove flag in the RSX prefix, and stops intercepting console input. On the following warm boot the CCP removes the RSX from memory.

SUBMIT Operation

When the SUBMIT utility begins execution, it opens and reads the file specified by <filespec> and creates a temporary SUBMIT file of the same name as type .$$$ on the system's temporary file drive. GENCPM sets the temporary file drive to the CCP's current default drive. If desired, the SETDEF utility can be used to set the temporary file drive to a different drive. As it creates the temporary file, SUBMIT performs the parameter substitutions requested by the <parameters> subfield of the SUBMIT command line.

After SUBMIT creates the temporary SUBMIT file, its operation is similar to that of the GET utility described above. The SUBMIT command file also has an attached RSX that performs console input redirection from a file. However, the SUBMIT RSX expands upon the simpler facilities provided by the GET RSX. Command lines in a SUBMIT file can be marked to indicate whether they are program or CCP input. Furthermore, if a program exhausts all its program input and the next SUBMIT command is a CCP command, the SUBMIT RSX temporarily reverts to console input. Redirected input from the SUBMIT file resumes when the program terminates.

Because CP/M Plus's SUBMIT facility is implemented with RSXs, SUBMIT files can be nested. That is, a SUBMIT file can contain additional SUBMIT or GET commands. Similarly, a GET command can specify a file that contains GET or SUBMIT commands. For example, when a SUBMIT command is encountered in a SUBMIT file, a new SUBMIT RSX is created below the current RSX. The new RSX handles console input until it reads end-of-file on its temporary SUBMIT file. At this point control reverts to the previous SUBMIT RSX.

CP/M PLUS SYSTEM CALLS

Under CP/M Plus a transient program can access standard operating-system functions and hardware resources through an invariant software interface. This interface consists of a number of system calls available to the transient program. Information applying to all the system calls is summarized in the sections below.

BDOS System-Call Conventions

CP/M Plus uses standard conventions for BDOS system calls. These calling conventions are summarized below:

Entry Parameters:
Register C: BDOS system-call Number
Registers DE: Byte or Word Value or Address

Return Parameters:
Register A: single-byte values = L
Registers HL: double-byte values
Register B: = H

Unsupported Calls Return:
system calls 0-127: HL = 00FFH
system calls 128-255: HL = 0000H

The BDOS does not restore registers before returning to the calling program. Therefore the responsibility for saving and restoring critical registers rests with the calling program.

The following example illustrates how a transient program calls a BDOS system call. This program reads characters continuously until it encounters an asterisk, then terminates execution by returning to the system.

```
bdos    equ   0005h       ;BDOS entry point in Page Zero
C_READ  equ   1           ;BDOS console-input system call
                          ;
        org   100h        ;Base of Transient Program Area
nextc:  mvi   c,C_READ    ;Use mnemonic for call number
        call  bdos        ;Return character in A
        cpi   '*'         ;End of processing?
        jnz   nextc       ;Loop if not
        ret               ;Terminate program
        end
```

System-Call Reference Tables

The following tables describe the CP/M Plus system calls. They are intended both as an introduction to the calls and as a reference for use during programming.

Note: A mnemonic has been assigned to each system call according to the functional group to which it belongs. The sections in this manual documenting the system calls are arranged alphabetically according to these mnemonics. All future releases of Digital Research operating systems will use this system of mnemonics, with extensions or modifications as needed. The reader is encouraged to become familiar with this system which makes the large number of system calls in sophisticated operating systems such as CP/M Plus easier to learn and use.

An equate file (SYSCALLS.ASM) containing the system call mnemonics and their numerical equivalents is provided for the convenience of the programmer (see Appendix E). This file can be included in the equate section of a program, making use of the standard mnemonics for the system call subroutines a simple matter.

Table B-3 enumerates the functional categories of system calls and their mnemonic prefixes. Table B-4 describes the categories of system calls and their general uses. Table B-5 presents the CP/M Plus system calls in numerical order for those who prefer that system. Use the tables as a quick reference to find the system call you need while programming.

TABLE B-3. SYSTEM CALL CATEGORIES

Prefix	Name	Definition
C_		Console System Calls - The Console System Calls handle console I/O operations on a character, string, or block basis.
DRV_		Disk-Drive System Calls - The Disk-Drive System Calls access and free drives, get and set specified disk parameters, and perform other disk- and drive-level functions.
F_		Disk-File System Calls - The Disk-File System Calls open and close, make and delete, and read from and write to disk files, as well as setting default passwords, error modes, attributes, and other file-level options.
L_		List-Device System Calls - The List-Device System Calls write characters or blocks to the default list device.
P_		Program System Calls - The Program System Calls terminate programs, chain to other programs, and perform other operations on programs.
S_		System System Calls - The System System Calls return various types of systems data, such as version numbers and addresses, and enable applications to call the BIOS directly.
T_		Time System Calls - The Time System Calls set the system calender and clock and return the time from them.

TABLE B-4. BDOS SYSTEM CALL SUMMARY

Mnemonic	Number	Name	Definition
C_AUXIN	3	Auxiliary Input	Return a character from the auxiliary input device.
C_AUXINST	7	Auxiliary Input Status	Return status of the auxiliary input device.
C_AUXOUT	4	Auxiliary Output	Send a character to the auxiliary output device.
C_AUXOUTST	8	Auxiliary Output Status	Return status of auxiliary output device.
C_DELIMIT	110	Get/Set Output Delimiter	Set or return current Output Delimiter.
C_MODE	109	Get/Set Console Mode	Set or return Console mode.
C_RAWIO	6	Raw Console I/O	Perform Raw mode I/O with the default virtual console.
C_READ	1	Read Console	Read a character from the default virtual console.
C_READSTR	10	Read Console Buffer	Read an edited line from the default virtual console.
C_STAT	11	Get Console Status	Obtain the status of the default virtual console.
C_WRITE	2	Write to Console	Write a character to the default virtual console.
C_WRITEBLK	111	Write Block	Write a string to the default virtual console.
C_WRITESTR	9	Write String	Write a string to the default virtual console until $.
DRV_ACCESS	38	Access Drive	For MP/M compatibility.
DRV_ALLOCVEC	27	Get Addr(Alloc)	Get the address of the disk allocation vector.
DRV_ALLRESET	13	Reset Disk System	Restore disk system to reset state.

TABLE B-4. BDOS SYSTEM CALL SUMMARY (Cont.)

Mnemonic	Number	Name	Definition
DRV_DPB	31	Get Addr(DPB)	Return the segment and offset address of the Disk Parameter Block for the default disk of the calling program.
DRV_FREE	39	Free Drive	For MP/M compatibility.
DRV_FREEBLOCKS	98	Free Blocks	Free temporarily allocated blocks in open files.
DRV_GET	25	Return Current Disk	Return the default disk of the calling program.
DRV_GETLABEL	101	Return Directory Label Data	Return the data byte of the directory label for the specified drive.
DRV_LOGINVEC	24	Return Login Vector	Return bit map of logged-in disk drives.
DRV_RESET	37	Reset Drive	Reset the specified drive(s).
DRV_ROVEC	29	Get R/O Vector	Return R/O bit vector.
DRV_SET	14	Select Disk	Set the default disk of calling program.
DRV_SETLABEL	100	Set Directory Label	Create or update a directory label.
DRV_SETRO	28	Write Protect Disk	Set the default disk to Read-Only.
DRV_SPACE	46	Get Disk Free Space	Return unallocated space on the specified disk drive.
F_ATTRIB	30	Set File Attributes	Set Disk File compatibility or interface attributes.
F_CLOSE	16	Close File	Close a disk file as specified in FCB.
F_DELETE	19	Delete File	Delete the disk file specified by the FCB.

TABLE B-4. BDOS SYSTEM CALL SUMMARY (Cont.)

Mnemonic	Number	Name	Definition
F_DMASET	26	Set DMA Address	Set the Direct Memory Address offset address.
F_ERRMODE	45	Set BDOS Error Mode	Set the Error mode to Default, Return, or Return and Display.
F_FLUSH	48	Flush Buffers	Write any data in the blocking/deblocking buffers to the disk.
F_LOCK	42	Lock Record	Assume exclusive ownership of one or more consecutive records in the FCB-specified disk file.
F_MAKE	22	Make File	Create the disk file as specified in the FCB.
F_MULTISEC	44	Set Multisector Count	Set the number of records for subsequent disk operations.
F_OPEN	15	Open File	Open a disk file as specified in FCB.
F_PARSE	152	Parse Filename	Parse an ASCII string and initialize an FCB.
F_PASSWD	106	Set Default Password	Establish a default password for file access.
F_RANDREC	36	Set Random Record	Return the Random-Record Number of the next sequential record of a Disk File in the specified FCB.
F_READ	20	Read Sequential	Read records sequentially from the FCB-specified disk file.
F_READRAND	33	Read Random	Read the FCB-specified record at random from a disk file.
F_RENAME	23	Rename File	Rename the FCB-specified disk file.

TABLE B-4. BDOS SYSTEM CALL SUMMARY (Cont.)

Mnemonic	Number	Name	Definition
F_SFIRST	17	Search for First	Find the first file that matches the specified FCB.
F_SIZE	35	Compute File Size	Return the size of a disk file in the specified FCB.
F_SNEXT	18	Search for Next	Find the next file matching the FCB of the previous SEARCH FOR FIRST call.
F_TIMEDATE	102	Read File Date Stamps and Password Mode	Return the XFCB of the FCB-specified disk file.
F_TRUNCATE	99	Truncate File	Set the last file record to the number in the referenced FCB.
F_TSTWRITE	41	Test and Write Record	Read record before writing to make sure it has not changed. For compatibility with MP/M.
F_UNLOCK	43	Unlock Record	Relinquish exclusive ownership of the FCB-specified records. For compatibility with MP/M.
F_USERNUM	32	Set/Get User Code	Set or return the default user number of the calling program.
F_WRITE	21	Write Sequential	Write records sequentially to the FCB-specified disk file.
F_WRITERAND	34	Write Random	Write the FCB-specified record at random to a disk file.
F_WRITEXFCB	103	Write File XFCB	Create or update the XFCB for the FCB-specified disk file.

TABLE B-4. BDOS SYSTEM CALL SUMMARY (Cont.)

Mnemonic	Number	Name	Definition
_WRITEZF	40	Write Random with Zero Fill	Write the FCB-specified record at random to a disk file, filling all previously unwritten lower-numbered records with zeros.
L_WRITE	5	Write to List	Write a character to the default list device.
L_WRITEBLK	112	List Block Write	Write a block of characters to the default list device.
P_CHAIN	47	Chain to Program	Load, initialize, and jump to the program specified in the DMA buffer.
P_LOAD	59	Load Overlay	Load the specified CMD file in memory; return its base-page segment address.
P_RETCODE	108	Get/Set Program Return Code	Set Program Return Code before terminating program.
P_TERMCPM	0	System Reset	Terminate calling program unconditionally, release all owned resources.
S_BDOSVER	12	Return Version Number	Return BDOS version number, CPU and operating system type.
S_BIOS	50	Direct Bios Calls	Call specified BIOS-character I/O routine.
S_RSX	60	Call Resident System Extension	Transfer control to the specified RSX.
S_SCB	49	Get/Set System Control Block	Return or set the value of a specified SCB field.
S_SERIAL	107	Return Serial Number	Return the system serial number.
T_GET	105	Get Date and Time	Obtain the system calendar and clock, hours and minutes only.

TABLE B-4. BDOS SYSTEM CALL SUMMARY (Cont.)

Mnemonic	Number	Name	Definition
T_SET	104	Set Date and Time	Set internal system calendar and clock to specified value.

The table below lists the CP/M Plus system calls in numerical order, showing the parameters a program must pass when making the system call, and the values it returns to the program.

TABLE B-5. BDOS SYSTEM CALL SUMMARY BY VALUE

DEC	HEX	Name	Page	Mnemonic	Input Parameters	Returned Values
0	00	System Reset		P_TERMCPM	none	none
1	01	Read Console		C_READ	none	A = char
2	02	Write to Console		C_WRITE	E = char	A = 00H
3	03	Auxiliary Input		C_AUXIN	none	A = char
4	04	Auxiliary Output		C_AUXOUT	E = char	A = 00H
5	05	Write to List		L_WRITE	E = char	A = 00H
6	06	Raw Console I/O		C_RAWIO	E = 0FFH/ 0FEH/0FDH/char	A = char/status/
7	07	Auxiliary Input Status		C_AUXINST	none	A = 00/0FFH
8	08	Auxiliary Output Status		C_AUXOUTST	none	A = 00/0FFH
9	09	Write String		C_WRITESTR	DE = .String	A = 00H
10	0A	Read Console Buffer		C_READSTR	DE = .Buffer/ 0FFFFH	Chars in buffer
11	0B	Get Console Status		C_STAT	none	A = 00/01
12	0C	Return Version Number		S_BDOSVER	none	HL= Version (0031H)
13	0D	Reset Disk System		DRV_ALLRESET	none	A = 00H
14	0E	Select Disk		DRV_GET	E = Disk Number	A = Err Flag
15	0F	Open File		F_OPEN	DE = .FCB	A = Dir Code
16	10	Close File		F_CLOSE	DE = .FCB	A = Dir Code
17	11	Search for First		F_SFIRST	DE = .FCB	A = Dir Code
18	12	Search for Next		F_SNEXT	none	A = Dir Code
19	13	Delete File		F_DELETE	DE = .FCB	A = Dir Code
20	14	Read Sequential		F_READ	DE = .FCB	A = Err Code
21	15	Write Sequential		F_WRITE	DE = .FCB	A = Err Code
22	16	Make File		F_MAKE	DE = .FCB	A = Dir Code
23	17	Rename File		F_RENAME	DE = .FCB	A = Dir Code
24	18	Return Login Vector		DRV_LOGINVEC	none	HL= Login Vector
25	19	Return Current Disk		DRV_GET	none	A = Cur Disk#

Note: The period (.) indicates the starting address of the specified module.

TABLE B-5. BDOS SYSTEM CALL SUMMARY BY VALUE (Cont.)

DEC	HEX	Name	Page	Mnemonic	Input Parameters	Returned Values
26	1A	Set DMA Address		F_DMASET	DE = .DMA	A = 00H
27	1B	Get Addr(Alloc)		DRV_ALLOCVEC	none	HL= .Alloc
28	1C	Write Protect Disk		DRV_SETRO	none	A = 00H
29	1D	Get R/O Vector		DRV_ROVEC	none	HL= R/O Vector
30	1E	Set File Attributes		F_ATTRIB	DE = .FCB	A = Dir Code
31	1F	Get Addr(DPB)		DRV_DPB	none	HL = .DPB
32	20	Set/Get User Code		F_USERNUM	E = 0FFH/ user number	A= Curr User/ 00H
33	21	Read Random		F_READRAND	DE = .FCB	A = Err Code
34	22	Write Random		F_WRITERAND	DE = .FCB	A = Err Code
35	23	Compute File Size		F_SIZE	DE = .FCB	r0, r1, r2 A = Err Flag
36	24	Set Random Record		F_RANDREC	DE = .FCB	r0, r1, r2
37	25	Reset Drive		DRV_RESET	DE = Drive Vector	A = 00H
38	26	Access Drive		DRV_ACCESS	none	A = 00H
39	27	Free Drive		DRV_FREE	none	A = 00H
40	28	Write Random with Zero Fill		F_WRITEZF	DE = .FCB	A = Err Code
41	29	Test and Write Record		F_TSTWRITE	DE = .FCB	A = 0FFH
42	2A	Lock Record		F_LOCK	DE = .FCB	A = 00H
43	2B	Unlock Record		F_UNLOCK	DE = .FCB	A = 00H
44	2C	Set Multisector Count		F_MULTISEC	E = # Sectors	A = Return Code
45	2D	Set BDOS Error Mode		F_ERRMODE	E = BDOS Err Mode	A = 00H
46	2E	Get Disk Free Space		DRV_SPACE	E = Drive number A = Err Flag	Number of Free Sectors
47	2F	Chain to Program		P_CHAIN	E = Chain Flag	A = 00H
48	30	Flush Buffers		F_FLUSH	E = Purge Flag	A = Err Flag
49	31	Get/Set System Control Block		S_SCB	DE = .SCB PB	A = Returned Byte HL= Returned Word
50	32	Direct BIOS Calls		S_BIOS	DE = .BIOS PB	BIOS Return
59	3B	Load Overlay		P_LOAD	DE = .FCB	A = Err Code
60	3C	Call Resident System Extension		S_RSX	DE = .RSX PB	A = Err Code
98	62	Free Blocks		DRV_FREEBLOCKS	none	A = 00H
99	63	Truncate File		F_TRUNCATE	DE = .FCB	A = Dir Code
100	64	Set Directory Label		DRV_SETLABEL	DE = .FCB	A = Dir Code
101	65	Return Directory Label Data		DRV_GETLABEL	E = Drive	A = Dir label data byte
102	66	Read File Date Stamps and Password Mode		F_TIMEDATE	DE = .FCB	A = Dir Code
103	67	Write File XFCB		F_WRITEXFCB	DE = .FCB	A = Dir Code
104	68	Set Date and Time		T_SET	DE = .DAT	A = 00H
105	69	Get Date and Time		T_GET	DE = .DAT	Date and Time A = seconds

Note: The period (.) indicates the starting address of the specified module.

TABLE B-5. BDOS SYSTEM CALL SUMMARY BY VALUE (Cont.)

DEC	HEX	Name	Page	Mnemonic	Input Parameters	Returned Values
106	6A	Set Default Password		F_PASSWD	DE = .Password	A = 00H
107	6B	Return Serial Number		S_SERIAL	DE = .Serial # field	Serial Number
108	6C	Get/Set Program Return Code		P_RETCODE	DE = 0FFFFH/Code	HL = Program Ret Code/ none
109	6D	Get/Set Console Mode		C_MODE	DE = 0FFFFH/Mode	HL = Console Mode/none
110	6E	Get/Set Output Delimiter		C_DELIMIT	DE = 0FFFFH/ E = Delimiter	A = Output Delimiter/ none
111	6F	Write Block		C_WRITEBLK	DE = .CCB	A = 00H
112	70	List Block		L_WRITEBLK	DE = .CCB	A = 00H
152	98	Parse Filename		F_PARSE	DE = .PFCB	See definition

Note: The period (.) indicates the starting address of the specified module.

Abbreviations used in the above table:

Abs	= Absolute	Dir	= Directory	Rqst	= Request
Addr	= Address	Err	= Error	Rtn	= Return
Char	= ASCII Character	Proc	= Process	Sp	= Space
Comm	= Command	#	= Number	Spec.	= Specified
Con	= Console	Pswd	= Password	Sys	= System
Cond.	= Conditional	Reloc	= Relocatable	Term.	= Termination
Ct	= Count	Rec	= Record	Vect	= Vector

DOS CONSOLE I/O

Console I/O system calls can be divided into four categories: basic console I/O, direct console I/O, buffered console input, and special console functions.

Using the basic console I/O system calls, programs can access the console device for simple input and output. The basic console-I/O system calls are:

1	**C_READ**	Inputs a single character
2	**C_WRITE**	Outputs a single character
9	**C_WRITESTR**	Outputs a string of characters
11	**C_STAT**	Signals if a character is ready for input
111	**C_WRITEBLK**	Outputs a block of characters

The input system call echoes the character to the console so that the user can identify the typed character. The output system calls expand tabs in columns of eight characters.

The basic I/O system calls also monitor the console to stop and start console output scroll at the user's request. If the user types a CTRL-S. these system calls suspend execution. Execution and console scrolling resume when the user types a CTRL-Q.

When the BDOS is waiting because of a CTRL-S, it scans input for three special characters: CTRL-Q, CTRL-C, and CTRL-P. If the user types any other character, the BDOS transmits a bell character (CTRL-G) to the console, discards the input character, and continues to wait. If the user types a CTRL-C, the BDOS executes a warm start which terminates the calling program. If the user types a CTRL-P, the BDOS toggles the printer echo switch. The printer echo switch controls whether console output is automatically echoed to the list device (LST:). The BDOS signals when it turns on printer echo by sending a bell character to the console.

All basic console I/O system calls discard any CTRL-Q or CTRL-P character that is not preceded by a CTRL-S character. Thus, the C_READ BDOS system call cannot read a CTRL-S, CTRL-Q, or CTRL-P character. Furthermore, these characters are invisible to the C_STAT system call.

Console I/O System Calls

```
+-----------------------------------------------+
!                                               !
!                   C_AUXIN                     !
!                                               !
!        SYSTEM CALL  3:  AUXILIARY INPUT       !
!                                               !
+-----------------------------------------------+
!                                               !
!      Entry Parameters:                        !
!          Register   C:  03H                   !
!                                               !
!      Returned   Value:                        !
!          Register   A:  ASCII Character       !
!                                               !
+-----------------------------------------------+
```

C_AUXIN reads the next character from the logical auxiliary input device (AUXIN:) into register A. Control does not return to the calling program until the character is read.

```
+-----------------------------------------------+
!                                               !
!                  C_AUXINST                    !
!                                               !
!    SYSTEM CALL  7:  AUXILIARY INPUT STATUS    !
!                                               !
+-----------------------------------------------+
!                                               !
!   Entry Parameters:                           !
!       Register   C:  07H                      !
!                                               !
!   Returned  Value :                           !
!       Register   A:  Auxiliary Input Status   !
!                                               !
+-----------------------------------------------+
```

C_AUXINST returns the value OFFH in register A if a character is ready for input from the logical auxiliary input device (AUXIN:). If no character is ready for input, the value OOH is returned.

```
+-----------------------------------------------+
!                                               !
!                  C_AUXOUT                     !
!                                               !
!       SYSTEM CALL  4:  AUXILIARY OUTPUT       !
!                                               !
+-----------------------------------------------+
!                                               !
!      Entry Parameters:                        !
!          Register   C:  04H                   !
!          Register   E:  ASCII Character       !
!                                               !
+-----------------------------------------------+
```

C_AUXOUT sends the ASCII character from register E to the logical auxiliary output device (AUXOUT:).

```
+-----------------------------------------------+
!                                               !
!                  C_AUXOUTST                   !
!                                               !
!    SYSTEM CALL  8:  AUXILIARY OUTPUT STATUS   !
!                                               !
+-----------------------------------------------+
!                                               !
!  Entry Parameters:                            !
!      Register   C:  08H                       !
!                                               !
!  Returned   Value:                            !
!      Register   A:  Auxiliary Output Status   !
!                                               !
+-----------------------------------------------+
```

C_AUXOUTST returns the value 0FFH in register A if the logical auxiliary output device (AUXOUT:) is ready to accept a character for output. If the device is not ready for output, the value 00H is returned.

```
+-----------------------------------------------+
!                                               !
!                  C_DELIMIT                    !
!                                               !
!  SYSTEM CALL  110:  GET/SET OUTPUT DELIMITER  !
!                                               !
+-----------------------------------------------+
!                                               !
!     Entry Parameters:                         !
!         Register   C:  6EH                    !
!         Register  DE:  OFFFFH (Get) or        !
!                    E:  Output Delimiter (Set) !
!                                               !
!     Returned   Value:                         !
!         Register   A:  Output Delimiter or    !
!                        (no value)             !
!                                               !
+-----------------------------------------------+
```

C_DELIMIT sets or interrogates the string delimiter for C_WRITESTR (Print String). The default delimiter value is a dollar sign ($). The CCP restores the Output Delimiter to the default value when a transient program is loaded.

If register pair DE = OFFFFH, then the current Output Delimiter is returned in register A. Otherwise C_DELIMIT sets the Output Delimiter to the value contained in register E.

```
+-----------------------------------------------+
!                                               !
!                   C_MODE                      !
!                                               !
!     SYSTEM CALL  109:  GET/SET CONSOLE MODE   !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  6DH                !
!             Register  DE:  OFFFFH (Get) or    !
!                            Console Mode (Set) !
!                                               !
!         Returned   Value:                     !
!             Register  HL:  Console Mode or    !
!                            (no value)         !
!                                               !
+-----------------------------------------------+
```

A program can set or interrogate the Console Mode by calling C_MODE. If register pair DE = OFFFFH, then the current Console Mode is returned in register HL. Otherwise C_MODE sets the Console Mode to the value contained in register pair DE.

The Console Mode is a 16-bit system parameter that determines the action of certain BDOS Console I/O system calls. The definition of the Console Mode is:

TABLE B-6. CONSOLE-MODE BIT DEFINITION

BIT NO	DEFINITION
bit 0	If this bit is set, the C_STAT system call returns true only if a CTRL-C is typed at the console. Programs that make repeated console status calls to test if execution should be interrupted can set this bit to interrupt on CTRL-C only. The CCP built-in commands DIR and TYPE run in this mode.
bit 1	Setting this bit disables stop and start scroll support for the basic console I/O system calls, which comprise the first category of system calls described in this section. When this bit is set, the C_READ system call reads CTRL-S, CTRL-Q, and CTRL-P; and the C_STAT system call returns true if the user types these characters. Use this mode in situations where raw console input and edited output are needed. While in this mode, you can use the C_RAWIO system call for input and input status; and the C_READ, C_WRITESTR. and C_WRITEBLK system calls for output without the possibility of the output system calls intercepting input CTRL-S, CTRL-Q, or CTRL-P characters.
bit 2	Setting this bit disables tab expansion and printer echo support for the C_WRITE, C_WRITESTR, and C_STAT system calls. Use this mode when nonedited output is required.
bit 3	This bit disables all CTRL-C intercept action in the BDOS. This mode is useful for programs that must control their own termination.
bits 8 and 9	The BDOS reserves these bits for the CP/M Plus GET RSX that performs console input redirection from a file. With one exception, these bits determine how the GET RSX responds to a program console-status request (the C_RAWIO, C_STAT, or S_BIOS system calls). bit 8 = 0, bit 9 = 0 conditional status bit 8 = 0, bit 9 = 1 false status bit 8 = 1, bit 9 = 0 true status bit 8 = 1, bit 9 = 1 do not perform redirection

Note: The Console Mode bits are numbered from right to left.

In conditional status mode, GET responds false to all status requests except for a status call preceded immediately by another status call. On the second call, GET responds with a true result. Thus, a program that spins on status to wait for a character is signaled that a character is ready on the second

call. In addition, a program that makes status calls periodically to see if the user wants to stop is not signaled.

When a transient program begins execution, the Console Mode bits are normally set to zero. However, the CP/M Plus utility GENCOM can attach an RSX header to a .COM file so that when it is loaded, the console mode bits are set differently. This feature allows you to modify a program's console I/O behavior without having to change the program.

```
+-----------------------------------------------+
!                                               !
!                   C_RAWIO                     !
!                                               !
!      SYSTEM CALL  6:  DIRECT CONSOLE I/O      !
!                                               !
+-----------------------------------------------+
!                                               !
!   Entry Parameters:                           !
!       Register   C:  06H                      !
!       Register   E:  0FFH (input/status) or   !
!                      0FEH (status) or         !
!                      0FDH (input)  or         !
!                      char (output)            !
!                                               !
!   Returned   Value:                           !
!       Register   A:  char or status           !
!                      (no value)               !
!                                               !
+-----------------------------------------------+
```

C_RAWIO can provide direct console I/O in situations where unadorned console I/O is required. C_RAWIO actually consists of several subsystem calls that support direct console input, output, and status checks. The BDOS does not filter out special characters during direct console I/O. The direct-output subsystem call does not expand tabs, and the direct-input subsystem call does not echo typed characters to the console.

Use direct console I/O carefully because it bypasses all the normal control-character functions. Programs that perform direct I/O through the BIOS under previous releases of CP/M should be changed to use direct I/O so that they can be fully supported under future releases of MP/M and CP/M.

A program calls C_RAWIO by passing one of four different values in register E. The values and their meanings are summarized in the following table.

TABLE B-7. C_RAWIO ENTRY PARAMETERS

Register E value	Meaning
OFFH	Console input/status command returns an input character; if no character is ready, a value of zero is returned.
OFEH	Console status command (On return, register A contains 00 if no character is ready; otherwise it contains FFH.)
OFDH	Console input command, returns an input character; this function will suspend the calling process until a character is ready.
ASCII	C_RAWIO assumes register E character contains a valid ASCII character and sends it to the console.

```
+-----------------------------------------------+
!                                               !
!                   C_READ                      !
!                                               !
!        SYSTEM CALL  1:  CONSOLE INPUT         !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  01H                  !
!                                               !
!       Returned   Value:                       !
!           Register   A:  ASCII Character      !
!                                               !
+-----------------------------------------------+
```

C_READ reads the next character from the logical console (CONIN:) to register A. Graphic characters, along with carriage return, line feed, and backspace (CTRL-H) are echoed to the console. Tab characters (CTRL-I) are expanded in columns of 8 characters. CTRL-S, CTRL-Q, and CTRL-P are normally intercepted as described below. All other nongraphic characters are returned in register A but are not echoed to the console.

When the Console Mode is in the default state (see C_MODE), C_READ intercepts CTRL-S, CTRL-Q, and CTRL-P characters. Any characters that are typed following a CTRL-S, and preceding a CTRL-Q are also intercepted. However, if start/stop scroll has been disabled by the Console Mode, the CTRL-S, CTRL-Q, and CTRL-P characters are not intercepted. Instead, they are returned in register A, but are not echoed to the console.

If printer echo has been invoked, all characters that are echoed to the console are also sent to the list device (LST:).

C_READ does not return control to the calling program until a nonintercepted character is typed, thus suspending execution if a character is not ready.

```
+-----------------------------------------------+
!                                               !
!                  C_READSTR                    !
!                                               !
!     SYSTEM CALL  10:  READ CONSOLE BUFFER     !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  0AH                 !
!            Registers DE:  Buffer Address      !
!                                               !
!          Returned Value:  Console Characters  !
!                           in Buffer           !
!                                               !
+-----------------------------------------------+
```

C_READSTR reads a line of edited console input from the logical console (CONIN:) to a buffer that register pair DE addresses. It terminates input and returns to the calling program when it encounters a return (CTRL-M) or a line feed (CTRL-J) character. C_READSTR also discards all input characters after the input buffer is filled. In addition, it outputs a bell character (CTRL-G) to the console when it discards a character to signal the user that the buffer is full. The input buffer addressed by DE has the following format:

```
DE: +0 +1 +2 +3 +4 +5 +6 +7 +8    . . .    +n
   ---------------------------------------------
   !mx!nc!c1!c2!c3!c4!c5!c6!c7!   . . .   !??!
   ---------------------------------------------
```

where mx is the maximum number of characters which the buffer holds, and nc is the number of characters placed in the buffer. The characters entered by the operator follow the nc value. The value mx must be set prior to making a C_READSTR call and may range in value from 1 to 255. Setting mx to zero is equivalent to setting mx to one. The value nc is returned to the calling program and may range from zero to mx. If nc < mx, then uninitialized positions follow the last character, denoted by ?? in the figure. Note that a terminating return or line-feed

character is not placed in the buffer and not included in the count nc.

If register pair DE is set to zero, C_READSTR assumes that an initialized input buffer is located at the current DMA address (see F_DMASET). This allows a program to put a string on the screen for the user to edit. To initialize the input buffer, set characters c1 through cn to the initial value followed by a binary zero terminator.

When a program calls C_READSTR with an initialized buffer, it operates as if the user had typed in the string. When C_READSTR encounters the binary zero terminator, it accepts input from the console. At this point the user can edit the initialized string or accept it as it is by pressing the RETURN key. However, if the initialized string contains a carriage return (CTRL-M) or a line-feed (CTRL-J) character, C_READSTR returns to the calling program without giving the user the opportunity to edit the string. The edit control characters are summarized in the table below.

C_READSTR also filters input for certain control characters. If the user types a CTRL-C as the first character in the line, C_READSTR terminates the calling program by branching to the BIOS warm-start entry point. A CTRL-C in any other position is simply echoed at the console. C_READSTR also watches for a CTRL-P keystroke, and if it finds one at any position in the command line, it toggles the printer echo switch. C_READSTR does not filter CTRL-S and CTRL-Q characters, but accepts them as normal input. In general, it accepts all control characters that it does not recognize as editing control characters as input characters. C_READSTR identifies a control character with a leading caret (^) when it echoes the control character to the console. Thus, CTRL-C appears as ^C in a C_READSTR command line on the screen.

TABLE B-8. C READSTR EDIT CONTROL CHARACTERS

Character	Edit Control Function
rub/del	Removes and echoes the last character if at the end of the line; otherwise deletes the character to the left of the current cursor position; GENCPM can change this function to CTRL-H.
CTRL-A	Moves cursor one character to the left.
CTRL-B	Moves cursor to the end of the line when at the beginning; otherwise moves cursor to the beginning of the line.
CTRL-C	Reboots when at the beginning of line; the Console Mode can disable this function.
CTRL-E	Causes physical end-of-line; if the cursor is positioned in the middle of a line, the characters at, and to the right of, the cursor are displayed on the next line.
CTRL-F	Moves cursor one character to the right.
CTRL-G	Deletes the character at the current cursor position when in the middle of the line; has no effect when the cursor is at the end of the line.
CTRL-H	Backspaces one character position when positioned at the end of the line; otherwise deletes the character to the left of the cursor; GENCPM can change this function to rub/del.
CTRL-J	(Line feed) terminates input; the cursor can be positioned anywhere in the line; the entire input line is accepted; sets the previous line buffer to the input line.
CTRL-K	Deletes all characters to the right of the cursor along with the character at the cursor.
CTRL-M	(Return) terminates input; the cursor can be positioned anywhere in the line; the entire input line is accepted; sets the previous line buffer to the input line.
CTRL-P	Echoes console output to the list device.
CTRL-R	Retypes the characters to the left of the cursor on the new line.

TABLE B-8. C_READSTR EDIT CONTROL CHARACTERS (Cont.)

Character	Edit Control Function
CTRL-U	Updates the previous line buffer to contain the characters to the left of the cursor; deletes current line, and advances to new line.
CTRL-W	Recalls previous line if current line is empty; otherwise moves cursor to end of line.
CTRL-X	Deletes all characters to the left of the cursor.

C_READSTR uses the console width field defined in the System Control Block. If the console width is exceeded when the cursor is positioned at the end of the line, C_READSTR automatically advances to the next line. The beginning of the line can be edited by entering a CTRL-R.

When a character is typed while the cursor is positioned in the middle of the line, the typed character is inserted into the line. Characters at, and to the right, of the cursor are shifted to the right. If the console width is exceeded, the characters disappear off the right of the screen. However, these characters are not lost. They reappear if characters are deleted out of the line, or if a CTRL-E is typed.

```
+-----------------------------------------------+
!                                               !
!                   C_STAT                      !
!                                               !
!     SYSTEM CALL  11:  GET CONSOLE STATUS      !
!                                               !
+-----------------------------------------------+
!                                               !
!      Entry Parameters:                        !
!          Register   C:  OBH                   !
!                                               !
!      Returned   Value:                        !
!          Register   A:  Console Status        !
!                                               !
+-----------------------------------------------+
```

C_STAT checks to see if a character has been typed at the logical console (CONIN:). If the Console Mode is in the default state (see C_MODE), C_STAT returns the value 01H in register A when a character is ready. If a character is not ready, it returns a value of 00H.

If the Console Mode is in CTRL-C Only Status mode, C_STAT returns the value 01H in register A only if a CTRL-C has been typed at the console.

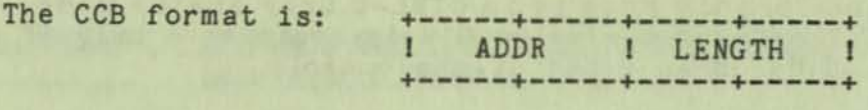
```
+-----------------------------------------------+
!                                               !
!                   C_WRITE                     !
!                                               !
!        SYSTEM CALL  2:  CONSOLE OUTPUT        !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  02H                  !
!           Register   E:  ASCII Character      !
!                                               !
+-----------------------------------------------+
```

C_WRITE sends the ASCII character from register E to the logical console device (CONOUT:). When the Console Mode is in the default state (see C_MODE), C_WRITE expands tab characters (CTRL-I) in columns of 8 characters; checks for CTRL-S, CTRL-Q, and echoes characters to the logical list device (LST:) if printer echo (CTRL-P) has been invoked.

```
+-----------------------------------------------+
!                                               !
!                  C_WRITEBLK                   !
!                                               !
!        SYSTEM CALL  111:  PRINT BLOCK         !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  6FH                 !
!            Register  DE:  CCB Address         !
!                                               !
!        Returned   Value:  None                !
!                                               !
+-----------------------------------------------+
```

C_WRITEBLK sends the character string located by the Character Control Block (CCB) addressed in register pair DE to the logical console (CONOUT):. If the Console Mode is in the default state (see C_MODE), C_WRITEBLK expands tab characters (CTRL-I) in columns of eight characters. It also checks for stop scroll (CTRL-S), start scroll (CTRL-Q), and echoes it to the logical list device (LST:) if printer echo (CTRL-P) has been invoked.

The CCB format is:



ADDR : Address of character string
LENGTH : Length of character string

C_WRITESTR

SYSTEM CALL 9: PRINT STRING

Entry Parameters:
Register C: 09H
Registers DE: String Address

C_WRITESTR sends the character string addressed by register pair DE to the logical console (CONOUT:) until it encounters a delimiter in the string. The default delimiter is a dollar sign ($) but it can be changed to any other value by C_DELIMIT. If the Console Mode is in the default state (see C_MODE), C_WRITESTR expands tab characters (CTRL-I) in columns of 8 characters. It also checks for CTRL-S, CTRL-Q, and echoes to the logical list device (LST:) if printer echo (CTRL-P) has been invoked.

BDOS FILE SYSTEM

The BDOS file system supports four categories of system calls: file-access system calls, directory system calls, drive-related system calls, and miscellaneous system calls. The file access category includes system calls to create a file (F_MAKE), open an existing file (F_OPEN), and close a file (F_CLOSE). Both the F_MAKE and F_OPEN system calls activate the file for subsequent access by BDOS file-access system calls. The BDOS read and write system calls are file-access system calls that operate either sequentially or randomly by record position. They transfer data in units of 128 bytes, which is the basic record size of the file system. The F_CLOSE system call makes any necessary updates to the directory to permanently record the status of an activated file.

File-Naming Conventions

Under CP/M Plus, a file specification consists of four parts: the drive specifier, the filename field, the filetype field, and the file password field. The general format for a command line file specification is shown below:

{d:}filename{.typ}{;password}

The drive-specifier field specifies the drive where the file is located. The filename and type fields identify the file. The password field specifies the password if a file is password-protected.

The drive, type, and password fields are optional, and the delimiters : . ; are required only when specifying their associated field. The drive specifier can be assigned a letter from A to P where the actual drive letters supported on a given system are determined by the BIOS implementation. When the drive letter is not specified, the current default drive is assumed.

The filename and password fields can contain one to eight nondelimiter characters. The filetype field can contain one to three nondelimiter characters. All three fields are padded with blanks, if necessary. Omitting the optional type or password fields implies a field specification of all blanks.

The CCP calls the F_PARSE system call to parse file specifications from a command line. See F_PARSE for

details of the operation of this system call.

It is not mandatory to follow the file-naming conventions of CP/M Plus when you create or rename a file with system calls. However, the conventions must be used if the file is to be accessed from a command line. For example, the CCP cannot locate a command file in the directory if its filename or filetype field contains a lowercase letter.

As a general rule, the filetype field names the generic category of a particular file, while the filename distinguishes individual files in each category. The following list of filetypes names some of the generic categories that have been established by usage convention.

ASM	Assembler Source	**PLI**	PL/I Source File
PRN	Printer Listing	**REL**	Relocatable Module
HEX	Hex Machine Code	**TEX**	TEX Formatter Source
BAS	Basic Source File	**BAK**	ED Source Backup
INT	Intermediate File	**SYM**	SID Symbol File
COM	Command File	**$$$**	Temporary File
PRL	Page Relocatable	**DAT**	Data File
SPR	Sys. Page Reloc.	**SYS**	System File

Disk and File Organization

The BDOS file system can support from 1 to 16 logical drives. The maximum file size supported on a drive is 32 megabytes. The maximum capacity of a drive is determined by the data-block size specified for the drive in the BIOS. The data-block size is the basic unit in which the BDOS allocates disk space to files.

Logical drives are divided into two regions: a directory area and a data area. The directory area contains entries that define which files exist on the drive. The directory entries corresponding to a particular file define those data blocks in the drive's data area that belong to the file. These data blocks contain the file's records.

Each disk file consists of a set of up to 262,144 128-byte records. Each record in a file is identified by its position in the file. This position is called the record's random record number. If a file is created sequentially, the first record has a position of zero, while the last record has a position one less than the number of records in the file. Such a file can be read sequentially in record position order beginning at record zero, or randomly by record position. Conversely, if a file is created randomly, records are added to the file by specified position.

A file created in this way is called sparse if positions exist within the file where a record has not been written.

The BDOS automatically allocates data blocks to a file to contain its records on the basis of the record positions consumed. Thus, a sparse file that contains two records, one at position zero, the other at position 262,143, consumes only two data blocks in the data area. Sparse files can only be created and accessed randomly, not sequentially. Note that any data block allocated to a file is permanently allocated to the file until the file is deleted or truncated. These are the only mechanisms supported by the BDOS for releasing data blocks belonging to a file.

Source files under CP/M Plus are treated as a sequence of ASCII characters, where each line of the source file is followed by a carriage-return line-feed sequence, 0DH followed by 0AH. Thus, a single 128-byte record could contain several lines of source text. The end of an ASCII file is denoted by a CTRL-Z character (1AH) or a real end-of-file, returned by the BDOS read operation. CTRL-Z characters embedded within machine code files such as .COM files are ignored. The actual end-of-file condition returned by the BDOS is used to terminate read operations.

File Control Block Definition

The File Control Block (FCB) is a data structure that is set up and initialized by a transient program, and then used by file access and directory system calls called by the transient program. Thus, the FCB is an important communication channel between the BDOS and a transient program. For example, when a program opens a file, and subsequently accesses it with BDOS read and write system calls, the BDOS file system maintains the current file state and position within the program's FCB. Some BDOS system calls use certain fields in the FCB for invoking special options. Other BDOS system calls use the FCB to return data to the calling program. In addition, all BDOS random-I/O system calls specify the random record number with a three-byte field at the end of the FCB.

When a transient program makes a file access or directory BDOS system call, register pair DE must address an FCB. The length of the FCB data area depends on the system call. For most system calls, the required length is 33 (21H) bytes. For the F_READRAND and F_WRITERAND system calls, the

F_TRUNCATE system call, and the F_SIZE system call, the FCB length must be 36 (24H) bytes. The FCB format is shown below:

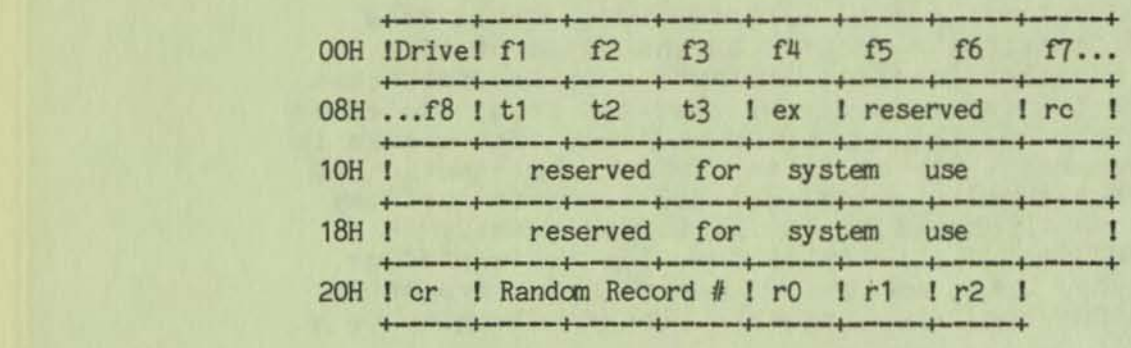


TABLE B-9. FCB FIELD DEFINITIONS

FCB Field	Explanation
dr	Drive code (0 - 16) 0 => use default drive for file 1 => auto disk select drive A, 2 => auto disk select drive B, ... 16=> auto disk select drive P.
f1...f8	Contain the filename in ASCII uppercase, with high bit = zero. f1', ..., f8' denote the high-order bit of these positions, and are file attribute bits.
t1,t2,t3	Contain the filetype in ASCII uppercase, with high bit = zero'. t1', t2', and t3' denote the high-order bit of these positions, and are file attribute bits. t1' = 1 => Read/Only file t2' = 1 => System file t3' = 1 => File has been archived
ex	Contains the current extent number, usually set to zero by the calling program, but can range zero - 31 during file I/O.
reserved	Reserved for internal system use.
rc	Record count for extent "ex" takes on values from 0 - 255 (values greater than 128 imply record count equals 128).
cr	Current record to read or write in a sequential file operation, normally set to zero by the calling program when a file is opened or created.

TABLE B-9. FCB FIELD DEFINITIONS (Cont.)

FCB Field	Explanation
Random Record #	Optional random record number from 0 to 262,143 (0 - 3FFFFH). Random Record # constitutes an 18-bit value with low byte r0, middle byte r1, and high byte r2.

For BDOS directory system calls, the calling program must initialize bytes 0 through 11 of the FCB before issuing the system call. The DRV_SETLABEL and F_WRITEXFCB system calls also require the calling program to initialize byte 12. The F_RENAME system call requires the calling program to place the new filename and type in bytes 17 through 27.

BDOS F_OPEN and F_MAKE system calls require the calling program to initialize bytes 0 through 12 of the FCB before making the call. Usually byte 12 is set to zero. In addition, if the file is to be processed from the beginning using sequential read or write system calls, byte 32 (cr) must be zeroed.

After an FCB is activated by an F_OPEN or F_MAKE system call, a program does not have to modify the FCB to perform sequential read or write operations. In fact, bytes 0 through 31 of an activated FCB must not be modified. However, the F_READRAND and F_WRITERAND system calls require that a program set bytes 33 through 35 to the requested random record number prior to making the system call.

File directory entries maintained in the directory area of each disk have the same format as FCBs, excluding bytes 32 through 35, except for byte 0 which contains the file's user number. Both the F_OPEN and F_MAKE system calls bring these entries, excluding byte 0, into memory in the FCB specified by the calling program. All read and write operations on a file must specify an FCB activated in this manner.

The BDOS updates the memory copy of the FCB during file processing to maintain the current position within the file. During file write operations, the BDOS updates the memory copy of the FCB to record the allocation of data to the file, and at the termination of file processing, the F_CLOSE system call permanently records this information on-disk.

Note: Data allocated to a file during file write operations is not completely recorded in the directory until the calling program issues an F_CLOSE system call. Therefore, a program that creates or

modifies files must close the files at the end of any write processing; otherwise data might be lost.

File Attributes

The high-order bits of the FCB filename (f1',...,f8') and filetype (t1',t2',t3') fields are called the Attribute Bits. Attribute Bits are one-bit Boolean fields (1 = on, 0 = off), that indicate two kinds of attributes within the file system: File Attributes and Interface Attributes.

The File Attribute Bits (f1',...,f4' and t1',t2',t3') can indicate that a file has a defined file attribute. These bits are recorded in a file's directory FCBs. File attributes can be set or reset only by the F_ATTRIB system call. When the F_MAKE system call creates a file, it initializes all file attributes to zero. A program can interrogate file attributes in an FCB activated by the F_OPEN system call, or in directory FCBs returned by the F_SFIRST and F_SNEXT system calls.

Note: The BDOS file system ignores File Attribute Bits when it attempts to locate a file in the directory.

TABLE B-10. FILE ATTRIBUTE BITS

Bit	Name	Definition
t1':		Read-Only attribute - The file system prevents write operations to a file with the Read-Only attribute set.
t2':		System attribute - This attribute, if set, identifies the file as a CP/M Plus system file. System files are not displayed by the CP/M Plus DIR command. In addition, user-zero system files can be accessed on a Read-Only basis from other user numbers.
t3':		Archive attribute - This attribute is designed for user-written archive programs. When an archive program copies a file to backup storage, it sets the archive attribute of the copied files. The file system automatically resets the archive attribute of a directory FCB that has been issued a write command. The archive program can test this attribute in each of the file's directory FCBs via the F_SFIRST and F_SNEXT system calls. If all directory FCBs have the archive attribute set, it indicates that the file has not been modified since the previous archive. Note that the CP/M Plus PIP utility supports file archival.

TABLE B-10. FILE ATTRIBUTE BITS (Cont.)

Bit	Name	Definition

f1',...,f4': User-definable attributes.
f5',...,f8': Interface attributes.
These attributes cannot be used as file attributes. Interface attributes f5' and f6' can request options for the F_MAKE, F_CLOSE, F_DELETE, and F_ATTRIB system calls. The table below defines options indicated by the f5' and f6' Interface Attribute Bits for these system calls.

TABLE B-11. BDOS INTERFACE ATTRIBUTES

BDOS System Call		Interface Attribute Definition
16	F_CLOSE	f5' = 1 : Partial Close
19	F_DELETE	f5' = 1 : Delete file XFCBs only
22	F_MAKE	f6' = 1 : Assign password to file
30	F_ATTRIB	f6' = 1 : Set file byte count

Each interface attribute is discussed in detail in the definitions of the above system calls. Attributes f5' and f6' are always reset when control is returned to the calling program. Interface attributes f7' and f8' are reserved for internal use by the BDOS file system.

User Number Conventions

The CP/M Plus User facility divides each drive directory into 16 logically independent directories, designated as user 0 through user 15. Physically, all user directories share the directory area of a drive. In most other aspects, however, they are independent. For example, files with the same name can exist on different user numbers of the same drive with no conflict. However, a single file cannot reside under more than one user number.

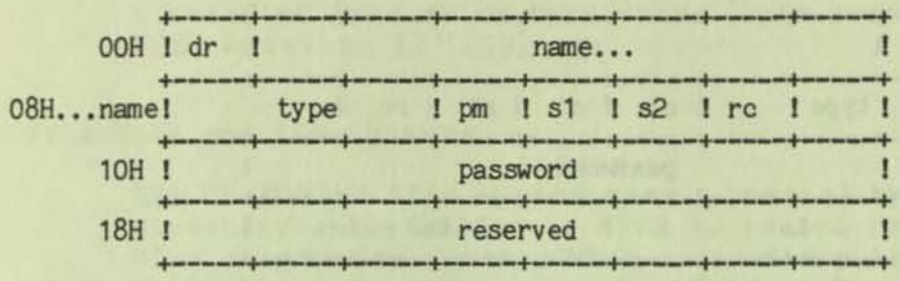
Only one user number is active for a program at one time, and the current user number applies to all drives on the system. Furthermore, the FCB format does not contain any field that can be used to override the current user number. As a result, all file and directory operations reference directories associated with the current user number. However, it

is possible for a program to access files on different user numbers; this can be accomplished by changing the user number with the F_USERNUM system call before accessing the desired file. Changing the user number in this way does not affect the CCP's user number displayed in the system prompt. When the transient program terminates, the original user number is restored. However, an option of the P_CHAIN system call allows a program to pass its current user number and default drive to the chained program. Note that this technique must be used carefully. An error occurs if a program attempts to read or write to a file under a user number different from the user number that was active when the file was opened.

User zero has special properties under CP/M Plus. When the current user number is not equal to zero, and if a requested file is not present under the current user number, the file system automatically attempts to open the file under user zero. If the file exists under user zero, and if it has the system (SYS) attribute bit (t2') set, the file is opened from user zero. Note, however, that files opened in this way are available only for read access. As a result, commonly needed utilities need not be copied to all user numbers on a directory, and you can control which user zero files are directly accessible from other user numbers.

Directory Labels and XFCBs

The BDOS file system includes two special types of FCBs: the XFCB and the Directory Label. The XFCB is an extended FCB that optionally can be associated with a file in the directory. If present, it contains the file's password. The format of the XFCB follows.



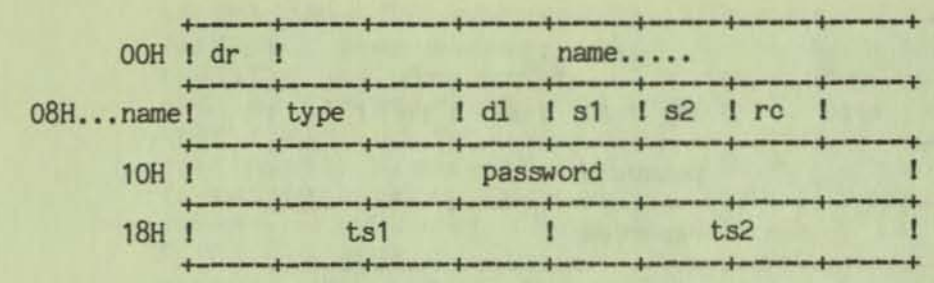
XFCB FORMAT

dr------drive code (0 - 16)
name----filename field
type----filetype field
pm------password mode
bit 7--Read mode
bit 6--Write mode
bit 5--Delete mode
**-----bit references are right to left, relative to zero
s1,s2,rc--reserved for system use
password--eight-byte password field (encrypted)
reserved--eight-byte reserved area

An XFCB can be created only on a drive that has a directory label, and only if the directory label has password protection activated. For drives in this state, an XFCB can be created for a file in three ways: by the F_MAKE system call, by the F_WRITEXFCB system call, or by typing a SET command at the console. The F_MAKE system call creates an XFCB if the calling program requests that a password be assigned to the created file. The F_WRITEXFCB system call can be used to assign a password to an existing file.

Note: In the directory, an XFCB is identified by a drive byte value (byte 0 in the FCB) equal to 16 (10H) + N, where N equals the user number.

For each drive the directory label specifies if file password support is to be activated, and if date and time stamping for files is to be performed. The format of the Directory Label follows.



DIRECTORY LABEL FORMAT

dr----------drive code (0 - 16)
name--------Directory Label name
type--------Directory Label type
dl----------Directory Label data byte
bit 7--require passwords for password-protected files
bit 6--perform access time stamping
bit 5--perform update time stamping
bit 4--perform create time stamping
bit 0--Directory Label exists
**-----bit references are right to left, relative to zero
s1,s2.rc----n/a
password----eight-byte password field (encrypted)
ts1---------four-byte creation or access time-stamp field
ts2---------four-byte update time-stamp field

Only one Directory Label can exist in a drive's directory. The Directory Label name and type fields are not used to search for a Directory Label; they can be used to identify a disk. A Directory Label can be created, or its fields can be updated by the DRV_SETLABEL system call. This system call can also assign a Directory Label a password. The Directory Label password, if assigned, cannot be circumvented, whereas file password protection is an option controlled by the Directory Label. Thus, access to the Directory Label password provides a kind of super-user status on that drive. That is, any password-protected file can be read simply by setting the Directory-Label Data Byte bit seven to zero, disabling password protection for the whole drive. However, this can be prevented by assigning a password to the Directory Label itself with the SET command.

The BDOS file system has no system call to read the Directory Label FCB directly. However, the Directory-Label data byte can be read directly with DRV_GETLABEL. In addition, the F_SFIRST and F_SNEXT system calls with a ? in the FCB drive byte can be used to find the Directory Label on the default drive. In the directory, the Directory Label is

identified by a drive byte value (byte zero in the FCB) equal to 32 (20H).

File Date and Time Stamps

The CP/M Plus File System uses a special type of directory entry called an SFCB to record date and time stamps for files. When a directory has been initialized for date and time stamping by invoking the INITDIR utility, SFCBs reside in every fourth position of the directory. Each SFCB maintains the date and time stamps for the previous three directory entries as shown in the figure below.

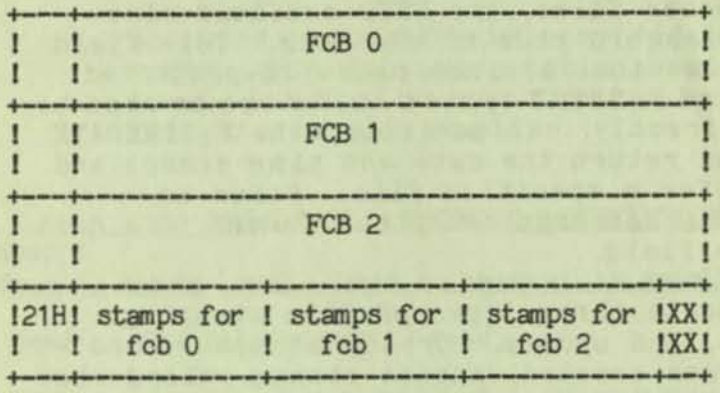
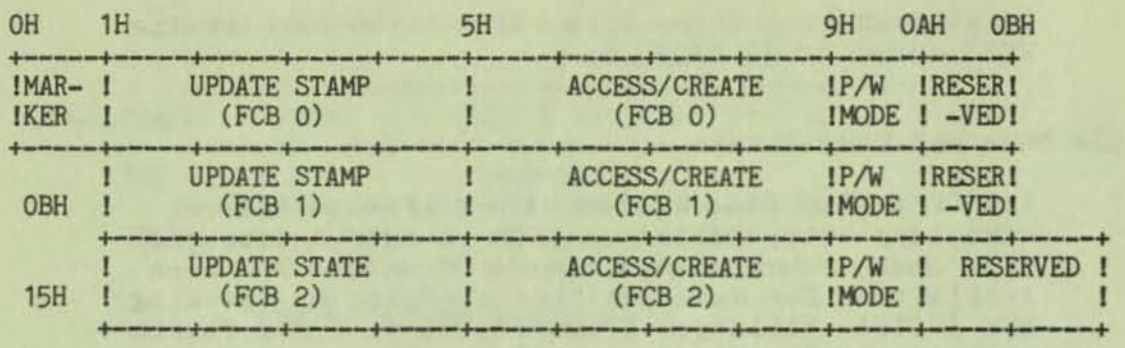


FIGURE B-6. DIRECTORY RECORD WITH SFCB

This figure shows a directory record that contains an SFCB. Directory records consist of four directory entries, each 32 bytes long. SFCBs always occupy the last position of a directory record.

The SFCB directory item contains five fields. The first field is one byte long and contains the value 21H. This value identifies the SFCB in the directory. The next three fields, the SFCB subfields, contain the date and time stamps for their corresponding FCB entries in the directory record. These fields are ten bytes long. The last byte of the SFCB is reserved for system use. The format of the SFCB subfields is shown below.



An SFCB subfield contains valid information only if its corresponding FCB in the directory record is an extent-zero FCB. In other words, the FCB referenced by the SFCB is a file's first directory entry. For password-protected files, the SFCB subfield also contains the password mode of the file. This field is zero for files that are not password-protected. The F_SFIRST and F_SNEXT system calls can be used to access SFCBs directly. In addition, the F_TIMEDATE system call can return the date and time stamps and password mode for a specified file. Refer to F_TIMEDATE for a description of the format of a date and time stamp field.

CP/M Plus supports three types of file stamping: create, access, and update. Create stamps record when the file was created, access stamps record when the file was last opened, and update stamps record the last time the file was modified. Create and access stamps share the same field. As a result, file-access stamping and file-create stamping are mutually exclusive. Turning on file-access stamping with the SET command, for example, automatically turns off file-create stamping if it is enabled.

The CP/M Plus utility, INITDIR, initializes a directory for date and time stamping by placing SFCBs in every fourth directory entry. Date and time stamping will not work on disks that have not been initialized in this manner. For initialized disks the Directory Label determines the type of date and time stamping supported for files on the drive. If a disk does not have a Directory Label, or if it is Read-Only, or if the disk's Directory Label does not specify date and time stamping, then date and time stamping for files is not performed.

Note: The Directory Label is also time-stamped, but these stamps are not made in an SFCB. The time-stamp fields in the last eight bytes of the Directory Label record when it was created and last updated. Access stamping for Directory Labels is not supported.

The BDOS file system uses the CP/M Plus system date and time when it records a date and time stamp. This value is maintained in a field in the System Control Block (SCB). The BIOS module directly updates the SCB system date and time field once per second. The CP/M Plus DATE utility can be used to set the system date and time.

File Passwords

Files can be assigned passwords in three ways: by the F_MAKE system call, by the F_WRITEXFCB system call, or by the SET command. A file's password can also be changed by the F_WRITEXFCB system call if the original password is supplied.

Password protection is provided in one of three modes. The following table shows the difference in access level allowed to system calls when the password is not supplied.

TABLE B-12. PASSWORD PROTECTION MODES

Password Mode	Access Level (when no password is supplied.)
1. Read	The file cannot be read.
2. Write	The file can be read, but not modified.
3. Delete	The file can be modified, but not deleted.

If a file is password-protected in Read mode, the password must be supplied to open the file. A file protected in Write mode cannot be written to without the password. A file protected in Delete mode allows read and write access, but the user must specify the password to delete the file, rename the file, or to modify the file's attributes. Thus, password protection in Read mode implies Write and Delete mode protection, and Write mode protection implies Delete mode protection. All three modes require the user to specify the password to delete the file, rename the file, or to modify the file's attributes.

If the correct password is supplied, or if password protection is disabled by the Directory Label, then access to the system calls is the same as for a file that is not password-protected. In addition, the F_SFIRST and F_SNEXT system calls are not affected by file passwords. The system calls that test for password are listed below:

15	F_OPEN
19	F_DELETE
23	F_RENAME
30	F_ATTRIB
99	F_TRUNCATE
100	DRV_SETLABEL
103	F_WRITEXFCB

File passwords are eight bytes in length. They are maintained in the XFCB Directory Label in encrypted form. To make a system call for a file that requires a password, a program must place the password in the first eight bytes of the current DMA, or specify it with the F_PASSWD system call prior to making the function call.

Note: The BDOS keeps an assigned default password value until it is replaced with a new assigned value, even if password protection is temporarily set to NONE using the SET command.

File Byte Counts

Although the logical record size of CP/M Plus is restricted to 128 bytes, CP/M Plus does provide a mechanism to store and retrieve a byte count for a file. This facility can identify the last byte of the last record of a file. The F_SIZE system call returns the random record number, plus one, of the last record of a file.

The F_ATTRIB system call can set a file's byte count. Conversely, the F_OPEN system call can return a file's byte count to the cr field of the FCB. The F_SFIRST and F_SNEXT system calls also return a file's byte count. These system calls return the byte count in the s1 field of the FCB in the current DMA buffer (see F_SFIRST and F_DMASET).

Note: The file system does not access or update the byte count value in file read or write operations. However, the F_MAKE system call does set the byte count of a file to zero when it creates a file in the directory.

BDOS Error Handling

The BDOS file system responds to error situations in one of three ways:

Method 1. It returns to the calling program with return codes in register A, H, and L identifying the error.

Method 2. It displays an error message on the console, and branches to the BIOS warm-start entry point, thereby terminating execution of the calling program.

Method 3. It displays an error message on the console, and returns to the calling program as in method 1.

The file system handles the majority of errors it detects by method 1. Two examples of this kind of error are the file-not-found error for the F_OPEN system call and the reading-unwritten-data error for a read system call. More serious errors, such as disk I/O errors, are usually handled by method 2. Errors in this category, called physical and extended errors, can also be reported by methods 1 and 3 under program control.

The BDOS Error Mode, which can exist in three states, determines how the file system handles physical and extended errors. In the default state, the BDOS displays the error message, and terminates the calling program (method 2). In return error mode, the BDOS returns control to the calling program with the error identified in registers A, H. and L (method 1). In return and display mode, the BDOS returns control to the calling program with the error identified in registers A, H, and L, and also displays the error message at the console (method 3). While both return modes protect a program from termination because of a physical or extended error, the return and display mode also allows the calling program to take advantage of the built-in error reporting of the BDOS file system. Physical and extended errors are displayed on the console in the following format:

CP/M Error on d: error message
system call = nn File = filename.typ

where d identifies the drive selected when the error condition is detected; error message identifies the error; nn is the system call number; and filename.typ identifies the file specified by the system call. If the system call did not involve an FCB, the file information is omitted.

The BDOS physical errors are identified by the following error messages:

- o Disk I/O
- o Invalid Drive
- o Read-Only File
- o Read-Only Disk

The Disk I/O error results from an error condition returned to the BDOS from the BIOS module.

If the BIOS does not support the selected disk, the BDOS returns an error code resulting in the Invalid-Drive error message.

The Read-Only File error is returned when a program attempts to write to a file that is marked with the Read-Only attribute.

The Read-Only Disk error is returned when a program writes to a disk that is in Read-Only status.

The BDOS extended errors are identified by the following error messages:

- o Password Error
- o File Exists
- o ? in Filename

The file Password Error is returned when the file password is not supplied, or when it is incorrect. The File Exists error is returned by the F_MAKE and F_RENAME system calls when the BDOS detects a duplicate filespec conflict.

The ? in Filename error is returned when the BDOS detects a ? in the filename or type field of the passed FCB for the F_RENAME, F_ATTRIB, F_OPEN, F_MAKE, and F_TRUNCATE BDOS system calls.

The following paragraphs describe the error return-code conventions of the BDOS file system calls. BDOS file system calls fall into three categories: they return an Error Code, a Directory Code, or an Error Flag.

The following system calls return an Error Code in register A:

20	F_READ	Read Sequential
21	F-WRITE	Write Sequential
33	F_READRAND	Read Random
34	F_WRITERAND	Write Random
40	F_WRITEZF	Write Random w/Zero Fill

TABLE B-13. REGISTER A ERROR-CODE DEFINITIONS

Code	Meaning
00	System call successful
255	Physical error : refer to register H
01	Reading unwritten data or No available directory space (Write Sequential)
02	No available data block
03	Cannot close current extent
04	Seek to unwritten extent
05	No available directory space
06	Random record number out of range
09	Invalid FCB (previous BDOS close call returned an error code and invalidated the FCB)
10	Media Changed (A media change was detected on the FCB's drive after the FCB was opened.)

For BDOS read or write system calls, the file system also sets register H to the number of 128-byte records successfully read or written before the error was encountered. On successful system calls, Error Code = zero, register H is set to zero. If the Error Code equals 255, register H contains a physical error code (see the table.)

The following system calls return a Directory Code in register A:

15	F_OPEN	Open File
16	F_CLOSE	Close File
17	F_SFIRST	Search for First
18	F_SNEXT	Search for Next
19	F_DELETE	Delete File
22	F_MAKE	Make File
23	F_RENAME	Rename File
30	F_ATTRIB	Set File Attributes
35	F_SIZE	Compute File Size
99	F_TRUNCATE	Truncate File
100	DRV_SETLABEL	Set Directory Label
102	F_TIMEDATE	Read File Date Stamps and Password Mode
103	F_WRITEXFCB	Write File XFCB

TABLE B-14. REGISTER A DIRECTORY-CODE DEFINITIONS

Code	Meaning
00 - 03	Successful system call
255	Unsuccessful system call

With the exception of the F_SFIRST and F_SNEXT system calls, all system calls in this category return with the directory code set to zero on successful returns.

However, for the F_SFIRST and F_SNEXT system calls, a successful Directory Code also identifies the relative starting position of the directory entry in the calling program's current DMA buffer.

If the F_ERRMODE system call is used to place the BDOS in return error mode, the following system calls return an Error Flag on physical errors:

14	DRV_SET	Select Disk
46	DRV_SPACE	Get Disk Free Space
48	F_FLUSH	Flush Buffers
98	DRV_FREEBLOCKS	Free Blocks
101	DRV_GETLABEL	Return Directory Label Data

TABLE B-15. REGISTER A ERROR-FLAG DEFINITIONS

Code	Meaning
00	Successful system call
255	Physical error : refer to register H

The BDOS returns non-zero values in register H to identify a physical or extended error if the BDOS Error Mode is in one of the return modes. Except for system calls that return a Directory Code, register A equal to 255 indicates that register H identifies the physical or extended error. For system calls that return a Directory Code, if register A equals 255 and register H is not equal to zero, register H identifies the physical or extended error. The following table shows the physical and extended error codes returned in register H.

TABLE B-16. REGISTER A PHYSICAL AND EXTENDED ERROR-CODE DEFINITIONS

Code	Meaning
00	No error, or not a register H error
01	Disk I/O error
02	Read-Only Disk
03	Read-Only File or File Opened under user zero from another user number or file password-protected in write mode and correct password not specified.
04	Invalid Drive : drive select error
07	Password Error
08	File Exists
09	? in Filename

The following two system calls represent a special case because they return an address in registers H and L.

27	DRV_ALLOCVEC	Get Addr(Alloc)
31	DRV_DPB	Get Addr(Disk Parms)

When the BDOS is in return error mode and it detects a physical error for these system calls, it returns to the calling program with registers A, H. and L all set to 255; otherwise they return no error code.

BDOS-Drive System Calls

```
+-----------------------------------------------+
!                                               !
!                  DRV_ACCESS                   !
!                                               !
!         SYSTEM CALL  38:  ACCESS DRIVE        !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  26H                !
!                                               !
+-----------------------------------------------+
```

This is an MP/M system call that is not supported under CP/M Plus. If called, the file system returns a zero in register A indicating that the access request is successful.

```
+-----------------------------------------------+
!                                               !
!                 DRV_ALLOCVEC                  !
!                                               !
!        SYSTEM CALL  27:  GET ADDR(ALLOC)      !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  1BH                !
!                                               !
!         Returned   Value:                     !
!             Registers HL:  ALLOC Address      !
!                                               !
+-----------------------------------------------+
```

CP/M Plus maintains an allocation vector in main memory for each active disk drive. Some programs use the information provided by the allocation vector to determine the amount of free data space on a drive.

Note: The allocation information might be inaccurate if the drive has been marked Read-Only.

DRV_ALLOCVEC returns the base address of the allocation vector for the currently selected drive in register pair HL. If a physical error is encountered when the BDOS error mode is one of the return modes (see the F_ERRMODE system call), DRV_ALLOCVEC returns the value OFFFFH in the register pair HL.

In banked CP/M Plus systems, the allocation vector can be placed in bank zero. In this case a transient program cannot access the allocation vector. However, the DRV_SPACE system call can be used to directly return the number of free 128-byte records on a drive. The CP/M Plus utilities that display a drive's free space, DIR and SHOW, use DRV_SPACE for that purpose.

```
+-----------------------------------------------+
!                                               !
!                  DRV_ALLRESET                 !
!                                               !
!      SYSTEM CALL  13:  RESET DISK SYSTEM      !
!                                               !
+-----------------------------------------------+
!                                               !
!           Entry Parameters:                   !
!               Register   C:  ODH              !
!                                               !
+-----------------------------------------------+
```

DRV_ALLRESET restores the file system to a reset state where all the disk drives are set to Read-Write (see DRV_SETRO and DRV_ROVEC), the default disk is set to drive A, and the default DMA address is reset to 0080H. This system call can be used, for example, by an application program that requires disk changes during operation. DRV_RESET can also be used for this purpose.

```
+-----------------------------------------------+
!                                               !
!                   DRV_DPB                     !
!                                               !
!      SYSTEM CALL  31:  GET ADDR (DPB PARMS)   !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  1FH                !
!                                               !
!          Returned  Value:                     !
!             Registers HL:  DPB Address        !
!                                               !
+-----------------------------------------------+
```

DRV_DPB returns the base address of the BIOS-resident Disk Parameter Block (DPB) for the currently selected drive in register pair HL. Refer to the section on Disk Parameter Headers for the format of the DPB. The calling program can use this address to extract the disk parameter values.

If a physical error is encountered when the BDOS error mode is one of the return modes (see the F_ERRMODE system call), DRV_DPB returns the value OFFFFH in the register pair HL.

```
+-----------------------------------------------+
!                                               !
!                   DRV_FREE                    !
!                                               !
!          SYSTEM CALL  39:  FREE DRIVE         !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!              Register  C:  27H                !
!                                               !
+-----------------------------------------------+
```

This is an MP/M system call that is not supported under CP/M Plus. If called, the file system returns a zero in register A indicating that the free request is successful.

the calling program in register A. Register A equal to zero indicates that no directory label exists on the specified drive. If a physical error is encountered by DRV_GETLABEL when the BDOS Error mode is in one of the return modes (see the F_ERRMODE system call), this function returns with register A set to OFFH (255 decimal) and register H set to one of the following:

01 : Disk I/O error
04 : Invalid drive error

```
+-----------------------------------------------+
!                                               !
!                  DRV_LOGINVEC                 !
!                                               !
!      SYSTEM CALL  24:  RETURN LOGIN VECTOR    !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  18H                !
!                                               !
!         Returned   Value:                     !
!             Registers HL:  Login Vector       !
!                                               !
+-----------------------------------------------+
```

DRV_LOGINVEC returns the login vector in register pair HL. The login vector is a 16-bit value with the least-significant bit of L corresponding to drive A; and the high-order bit of H corresponding to the sixteenth drive, labeled P. A 0 bit indicates that the drive is not on-line, while a 1 bit indicates the drive is active. A drive is made active by either an explicit BDOS Select Disk call (DRV_SET) or an implicit selection when a BDOS file operation specifies a non-zero dr byte in the FCB. DRV_LOGINVEC maintains compatibilty with earlier releases since registers A and L contain the same values upon return.

```
+-----------------------------------------------+
!                                               !
!                  DRV_RESET                    !
!                                               !
!          SYSTEM CALL  37:  RESET DRIVE        !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  25H                !
!             Register  DE:  Drive Vector       !
!                                               !
!         Returned   Value:                     !
!             Register   A:  00H                !
!                                               !
+-----------------------------------------------+
```

DRV_RESET restores the specified drive(s) to the reset state. A reset drive is not logged-in and is in Read-Write status. The passed parameter in register pair DE is a 16-bit vector of drives to be reset, where the least-significant bit corresponds to the first drive A; and the high-order bit corresponds to the sixteenth drive, labeled P. Bit values of 1 indicate that the specified drive is to be reset.

DRV_ALLRESET and DRV_RESET allow a program to control when a disk's directory is to be reinitialized (logged-in) for file operations. When CP/M Plus is cold-started, all drives are in the reset state. Subsequently, as drives are referenced, they are automatically logged-in by the file system until reset by a system call or a system reset. Note that DRV_ALLRESET and DRV_RESET have similar effects except that whereas DRV_ALLRESET automatically resets all drives on the system, any combination of drives can be reset with DRV_RESET.

The primary use of DRV_RESET under CP/M Plus is to prepare for a media change on a drive. Subsequently, when the drive is accessed by a system call, the drive is automatically logged-in. Be sure to close your files, particularly files that have been written to, prior to resetting a drive.

Although CP/M Plus automatically relogs-in removable media when media changes are detected, you should still explicitly reset a drive before prompting the user to change disks.

```
+-----------------------------------------------+
!                                               !
!                  DRV_ROVEC                    !
!                                               !
!     SYSTEM CALL  29:  GET READ-ONLY VECTOR    !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  1DH                !
!                                               !
!         Returned   Value:                     !
!             Registers HL:  R/O Vector Value   !
!                                               !
+-----------------------------------------------+
```

DRV_ROVEC returns a bit vector in register pair HL that indicates which drives have the temporary Read-Only bit set. The Read-Only bit can be set only by a BDOS Write-Protect Disk call.

The format of the bit vector is analogous to that of the login vector returned by DRV_LOGINVEC. The least-significant bit corresponds to drive A, while the most-significant bit corresponds to drive P.

```
+-----------------------------------------------+
!                                               !
!                  DRV_SET                      !
!                                               !
!         SYSTEM CALL  14:  SELECT DISK         !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  0EH                  !
!           Register   E:  Selected Disk        !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Error Flag           !
!           Register   H:  Physical Error       !
!                                               !
+-----------------------------------------------+
```

DRV_SET designates the disk drive named in register E as the default disk for subsequent BDOS file operations. Register E is set to 0 for drive A, 1 for drive B, and so on through 15 for drive P in a full 16-drive system. In addition, DRV_SET logs in the designated drive if it is currently in the reset state. Logging-in a drive activates the drive's directory until the next disk system reset or drive reset operation.

FCBs that specify drive code zero (dr = 00H) automatically reference the currently selected default drive. FCBs with drive code values between 1 and 16, however, ignore the selected default drive and directly reference drives A through P.

Upon return, register A contains a zero if the select operation was successful. If a physical error was encountered, DRV_SET performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is in the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise DRV_SET returns to the calling program with register A set to 0FFH and register H set to one of the following physical error codes:

```
01 : Disk I/O Error
04 : Invalid drive
```

```
+-----------------------------------------------+
!                                               !
!                  DRV_SETLABEL                 !
!                                               !
!    SYSTEM CALL  100:  SET DIRECTORY LABEL     !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  64H                 !
!            Register  DE:  FCB Address         !
!                                               !
!        Returned   Value:                      !
!            Register  A :  Directory Code      !
!            Register  H :  Physical or         !
!                           Extended Error      !
!                                               !
+-----------------------------------------------+
```

DRV_SETLABEL creates a directory label or updates the existing directory label for the specified drive. The calling program passes in register pair DE the address of an FCB containing the name, type, and extent fields to be assigned to the directory label. The name and type fields of the referenced FCB are not used to locate the directory label in the directory; they are simply copied into the updated or created directory label. The extent field of the FCB (byte 12) contains the user's specification of the directory-label data byte. The definition of the directory-label data byte is:

bit 7--Require passwords for password-protected files
6--Perform access date and time stamping
5--Perform update date and time stamping
4--Perform create date and time stamping
0--Assign a new password to the directory label

If the current directory label is password-protected, the correct password must be placed in the first eight bytes of the current DMA, or have been previously established as the default password (see F_PASSWD). If bit 0 (the low-order bit) of byte 12 of the FCB is set to 1, it indicates that a new password for the directory label has been placed in the second eight bytes of the current DMA.

DRV_SETLABEL also requires that the referenced directory contain SFCBs to activate date and time stamping on the drive. If an attempt is made to activate date and time stamping when no SFCBs exist, DRV_SETLABEL returns an error code of 0FFH in register A and performs no action. The CP/M Plus INITDIR utility initializes a directory for date and time stamping by placing an SFCB record in every fourth entry of the directory.

DRV_SETLABEL returns a Directory Code in register A with the value 0 if the directory label create or update is successful; or 0FFH (255 decimal) if no space exists in the referenced directory to create a directory label; or if date and time stamping was requested and the referenced directory did not contain SFCBs. Register H is set to zero in both of these cases. If a physical error or extended error is encountered, DRV_SETLABEL performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise DRV_SETLABEL returns to the calling program with register A set to 0FFH and register H set to one of the following physical or extended error codes:

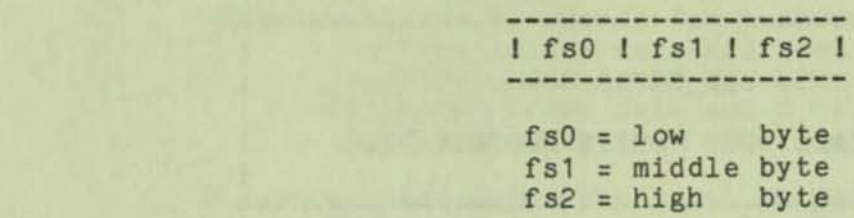
01 : Disk I/O error
02 : Read-Only disk
04 : Invalid drive error
07 : File password error

```
+-----------------------------------------------+
!                                               !
!                  DRV_SETRO                    !
!                                               !
!      SYSTEM CALL  28:  WRITE PROTECT DISK     !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  1CH                !
!                                               !
+-----------------------------------------------+
```

DRV_SETRO provides temporary write protection for the currently selected disk by marking the drive as Read-Only. No program can write to a disk that is in the Read-Only state. A drive reset operation must be performed for a Read-Only drive to restore it to the Read-Write state (see DRV_ALLRESET and DRV_RESET).

```
+-----------------------------------------------+
!                                               !
!                  DRV_SPACE                    !
!                                               !
!      SYSTEM CALL  46:  GET DISK FREE SPACE    !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  2EH                !
!                        E:  Drive              !
!                                               !
!         Returned   Value:  First 3 bytes      !
!                            of current DMA     !
!                            buffer             !
!             Register   A:  Error Flag         !
!             Register   H:  Physical Error     !
!                                               !
+-----------------------------------------------+
```

DRV_SPACE determines the number of free sectors, 128-byte records, on the specified drive. The calling program passes the drive number in register E, with 0 for drive A, 1 for B, and so on, through 15 for drive P in a full 16-drive system. DRV_SPACE returns a binary number in the first 3 bytes of the current DMA buffer. This number is returned in the following format:



Note: The returned free-space value might be inaccurate if the drive has been marked Read-Only.

Upon return, register A is set to zero if the system call is successful. However, if the BDOS Error Mode is one of the return modes (see F_ERRMODE) and a physical error is encountered, register A is set to OFFH (255 decimal) and register H is set to one of the following values:

01 : Disk I/O error
04 : Invalid drive error

BDOS File System Calls

F_ATTRIB

SYSTEM CALL 30: SET FILE ATTRIBUTES

Entry Parameters:		
Register	C:	1EH
Registers	DE:	FCB Address
Returned Value:		
Register	A:	Directory Code
Register	H:	Physical or Extended Error

By calling F_ATTRIB, a program can modify a file's attributes and set its last-record byte count. Other system calls can be called to interrogate these file parameters, but only F_ATTRIB can change them. The file attributes that can be set or reset by F_ATTRIB are f1' through f4', Read-Only (t1'), System (t2'), and Archive (t3').

The register pair DE addresses an FCB containing a filename with the appropriate attributes set or reset. The calling program must ensure that it does not specify an ambiguous filename. In addition, if the specified file is password-protected, the correct password must be placed in the first eight bytes of the current DMA buffer or have been previously established as the default password (see F_PASSWD).

Interface attribute f6' specifies whether the last-record byte count of the specified file is to be set:

f6' = 0 : Do not set byte count (default mode)
f6' = 1 : Set byte count

If interface attribute f6' is set, the calling program must set the cr field of the referenced FCB to the byte-count value. A program can access a file's byte-count value with the F_OPEN, F_SFIRST, or F_SNEXT system calls.

F_ATTRIB searches the referenced directory for entries belonging to the current user number that match the FCB-specified name and type fields. It then updates the directory to contain the selected indicators, and if interface attribute f6' is set, the specified byte-count value. Note that the last-record byte count is maintained in byte 13 of a file's directory FCBs.

File attributes t1', t2', and t3' are defined by CP/M Plus. Attributes f1' through f4' are not presently used, but can be useful for application programs because they are not involved in the matching program used by the BDOS during Open File and Close File operations. Indicators f5' through f8' are reserved for use as interface attributes.

Upon return, F_ATTRIB returns a Directory Code in register A with the value 0 if the system call is successful, or 0FFH (255 Decimal) if the file specified by the referenced FCB is not found. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_ATTRIB performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the program is terminated. Otherwise F_ATTRIB returns to the calling program with register A set to 0FFH, and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
02 : Read-Only disk
04 : Select error
07 : File password error
09 : ? in filename or filetype field
```

```
+-----------------------------------------------+
!                                               !
!                   F_CLOSE                     !
!                                               !
!         SYSTEM CALL  16:  CLOSE FILE          !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  10H                  !
!           Registers DE:  FCB Address          !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Directory Code       !
!           Register   H:  Physical or          !
!                          Extended Error       !
!                                               !
+-----------------------------------------------+
```

F_CLOSE performs the inverse of the F_OPEN system call. The calling program passes the address of an FCB in register pair DE. The referenced FCB must have been previously activated by a successful F_OPEN or F_MAKE system call. Interface attribute f5' specifies how the file is to be closed as shown below:

```
f5' = 0 : Permanent close (default mode)
f5' = 1 : Partial close
```

A permanent close operation indicates that the program has completed file operations on the file. A partial close operation updates the directory, but indicates that the file is to be maintained in the open state.

If the referenced FCB contains new information because of write operations to the FCB, F_CLOSE permanently records the new information in the referenced disk directory. Note that the FCB does not contain new information, and the directory update step is bypassed if only read or update operations have been made to the referenced FCB.

Upon return, F_CLOSE returns a directory code in register A with the value 00H if the close was successful, or FFH (255 decimal) if the file was not found. Register H is set to zero in both of these

cases. If a physical or extended error is encountered, F_CLOSE performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is in the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_CLOSE returns to the calling program with register A set to 0FFH and register H set to one of the following physical error codes:

```
01 : Disk I/O error
02 : Read-Only disk
04 : Invalid drive error
```

```
+-----------------------------------------------+
!                                               !
!                  F_DELETE                     !
!                                               !
!         SYSTEM CALL  19:  DELETE FILE         !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  13H                  !
!           Registers DE:  FCB Address          !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Directory Code       !
!           Register   H:  Extended or          !
!                          Physical Error       !
!                                               !
+-----------------------------------------------+
```

F_DELETE removes files or XFCBs that match the FCB addressed in register pair DE. The filename and filetype can contain ambiguous references (question marks in bytes f1 through t3), but the dr byte cannot be ambiguous as it can in the F_SFIRST and F_SNEXT system calls. Interface attribute f5' specifies the type of delete operation that is performed.

```
f5' = 0 : Standard Delete (default mode)
f5' = 1 : Delete only XFCBs
```

If any of the files that the referenced FCB specify are password-protected, the correct password must be placed in the first eight bytes of the current DMA buffer, or have been previously established as the default password (see F_PASSWD).

For standard delete operations, F_DELETE removes all directory entries belonging to files that match the referenced FCB. All disk directory and data space owned by the deleted files is returned to free space

and becomes available for allocation to other files. Directory XFCBs that were owned by the deleted files are also removed from the directory. If interface attribute f5' of the FCB is set to 1, F_DELETE deletes only the directory XFCBs that match the referenced FCB.

Note: If any of the files that match the input FCB specification fail the password check, or are Read-Only, then F_DELETE does not delete any files or XFCBs. This applies to both types of delete operations.

Upon return, F_DELETE returns a Directory Code in register A with the value 0 if the delete is successful, or 255 (0FFH) if no file that matches the referenced FCB is found. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_DELETE performs different actions depending on the BDOS error mode (see SETERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_DELETE returns to the calling program with register A set to 0FFH and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
02 : Read-Only disk
03 : Read-Only file
04 : Invalid drive error
07 : File password error
```

```
+-----------------------------------------------+
!                                               !
!                  F_DMASET                     !
!                                               !
!       SYSTEM CALL  26:  SET DMA ADDRESS       !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  1AH                !
!             Registers DE:  DMA Address        !
!                                               !
+-----------------------------------------------+
```

DMA is an acronym for Direct Memory Address, which is often used in connection with disk controllers that directly access the memory of the computer to transfer data to and from the disk subsystem. Under CP/M Plus, the current DMA is usually defined as the buffer in memory where a record resides before a disk write and after a disk read operation. If the BDOS

Multisector Count is equal to one (see F_MULTISEC), the size of the buffer is 128 bytes. However, if the BDOS Multisector Count is greater than one, the size of the buffer must equal N * 128, where N equals the Multisector Count.

Some system calls also use the current DMA to pass parameters and to return values. For example, system calls that check and assign file passwords require that the password be placed in the current DMA. As another example, DRV_SPACE returns its results in the first 3 bytes of the current DMA. When the current DMA is used in this context, the size of the buffer in memory is determined by the specific requirements of the system call.

When a transient program is initiated by the CCP, its DMA address is set to 0080H. DRV_ALLRESET also sets the DMA address to 0080H. F_DMASET can change this default value to another memory address. The DMA address is set to the value passed in the register pair DE. The DMA address remains at this value until it is changed by another F_DMASET or DRV_ALLRESET call.

```
+-----------------------------------------------+
!                                               !
!                  F_ERRMODE                    !
!                                               !
!      SYSTEM CALL  45:  SET BDOS ERROR MODE    !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  2DH                !
!                        E:  BDOS Error Mode    !
!                                               !
!         Returned   Value:  None               !
!                                               !
+-----------------------------------------------+
```

F_ERRMODE sets the BDOS error mode for the calling program to the mode specified in register E. If register E is set to 0FFH (255 decimal), the error mode is set to Return Error mode. If register E is set to 0FEH (254 decimal), the error mode is set to Return and Display mode. If register E is set to any other value, the error mode is set to the default mode.

F_ERRMODE determines how physical and extended errors are handled for a program. The Error Mode can exist in three modes: the default mode, Return Error mode, and Return and Display Error mode. In the default

mode, the BDOS displays a system message at the console that identifies the error and terminates the calling program. In the return modes, the BDOS sets register A to 0FFH (255 decimal), places an error code that identifies the physical or extended error in register H, and returns to the calling program. In Return and Display mode, the BDOS displays the system message before returning to the calling program. No system messages are displayed, however, when the BDOS is in Return Error mode.

```
+-----------------------------------------------+
!                                               !
!                   F_FLUSH                     !
!                                               !
!        SYSTEM CALL  48:  FLUSH BUFFERS        !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  30H                 !
!            Register   E:  Purge Flag          !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Error Flag          !
!            Register   H:  Physical Error      !
!                                               !
+-----------------------------------------------+
```

F_FLUSH forces the write of any write-pending records contained in internal blocking/deblocking buffers. If register E is set to 0FFH, this system call also purges all active data buffers. Programs that provide write with read-verify support need to purge internal buffers to ensure that verifying reads actually access the disk instead of returning data that is resident in internal data buffers. The CP/M Plus PIP utility is an example of such a program.

Upon return, register A is set to zero if the flush operation is successful. If a physical error is encountered, F_FLUSH performs different actions depending on the BDOS error mode (see SETERRMODE). If the BDOS error mode is in the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_FLUSH returns to the calling program with register A set to 0FFH and register H set to the following physical error code:

01 : Disk I/O error
02 : Read-Only disk
04 : Invalid drive error

```
+-----------------------------------------------+
!                                               !
!                   F_LOCK                      !
!                                               !
!         SYSTEM CALL  42:  LOCK RECORD         !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!              Register C:  2AH                 !
!                       DE:  FCB Address        !
!                                               !
!         Returned  Value:                      !
!              Register A:  00H                 !
!                                               !
+-----------------------------------------------+
```

F_LOCK is an MP/M II system call that is supported under CP/M Plus only to provide compatibility between CP/M Plus and MP/M. It is intended for use in situations where more than one running program has Read-Write access to a common file. Because CP/M Plus is a single-user operating system in which only one program can run at a time, this situation cannot occur. Thus, under CP/M Plus F_LOCK performs no action except to return the value 00H in register A indicating that the record lock operation is successful.

```
+-----------------------------------------------+
!                                               !
!                   F_MAKE                      !
!                                               !
!         SYSTEM CALL  22:  MAKE FILE           !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  16H                 !
!            Registers DE:  FCB Address         !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Directory Code      !
!            Register   H:  Physical or         !
!                           Extended Error      !
!                                               !
+-----------------------------------------------+
```

F_MAKE creates a new directory entry for a file under the current user number. It also creates an XFCB for the file if the referenced drive has a directory label that enables password protection on the drive, and the calling program assigns a password to the

file.

The calling program passes the address of the FCB in register pair DE, with byte 0 of the FCB specifying the drive; bytes 1 through 11 specifying the filename and filetype; and byte 12 set to the extent number. Usually byte 12 is set to zero. Byte 32 of the FCB (the cr field) must be initialized to zero, before or after the Make call, if the intent is to write sequentially from the beginning of the file.

Interface attribute f6' specifies whether a password is to be assigned to the created file.

f6' = 0 : Do not assign password (default)
f6' = 1 : Assign password to created file

When attribute f6' is set to 1, the calling program must place the password in the first 8 bytes of the current DMA buffer, and set byte 9 of the DMA buffer to the password mode (see the F_TIMEDATE system call).

Note: F_MAKE only interrogates interface attribute f6' if passwords are activated on the referenced drive.

F_MAKE returns with an error if the referenced FCB names a file that currently exists in the directory under the current user number.

If the F_MAKE system call is successful, it activates the referenced FCB for file operations by opening the FCB, and initializes both the directory entry and the referenced FCB to an empty file. It also initializes all file attributes to zero. In addition, F_MAKE makes a Creation date and time stamp for the file if the following conditions are satisfied: the referenced drive has a directory label that requests Creation date and time stamping, and the FCB extent number field is equal to zero. F_MAKE also makes an Update stamp if the directory label requests update stamping and the FCB extent field is equal to zero.

If the referenced drive contains a directory label that enables password protection, and if interface attribute f6' has been set to 1, F_MAKE creates an XFCB for the file. In addition, F_MAKE also assigns the password and the password mode placed in the first nine bytes of the DMA to the XFCB.

Upon return, F_MAKE returns a directory code in register A with the value zero if the make operation is successful, or 255 (0FFH) if no directory space is available. Register H is set to 0 in both of these

cases. If a physical or extended error is encountered, F_MAKE performs different actions depending on the BDOS error mode (see SETERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_MAKE returns to the calling program with register A set to OFFH, and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
02 : Read-Only disk
04 : Invalid drive error
08 : File already exists
09 : ? in filename or filetype field
```

```
+-----------------------------------------------+
!                                               !
!                  F_MULTISEC                   !
!                                               !
!  SYSTEM CALL  44:  SET MULTISECTOR COUNT      !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  2CH                 !
!                       E:  Number of Sectors   !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Return Code         !
!                                               !
+-----------------------------------------------+
```

F_MULTISEC provides logical record blocking under CP/M Plus. It enables a program to read and write from 1 to 128 physical records of 128 bytes at a time during subsequent BDOS Read and Write system calls.

F_MULTISEC sets the Multisector Count value for the calling program to the value passed in register E. Once set, the specified Multisector Count remains in effect until the calling program makes another F_MULTISEC call and changes the value.

Note: The CCP sets the Multisector Count to 1 when it initiates a transient program.

CP/M Plus can read or write multiple 128-byte disk records in a single BDOS system call. In a multisector I/O operation, data is transferred directly between the TPA and the drive. In addition, the BIOS can use multisector I/O to optimize the I/O operation, resulting in better performance. Thus, the primary objective of multisector I/O is to

improve sequential I/O performance. The number of records read or written with multisector I/O ranges from 1 to 128. The Multisector Count is set to one when a transient program begins execution. However, transient programs can set the CP/M Plus Multisector Count to 128 when sufficient buffer space is available.

Note: The greatest potential performance increases are obtained when the Multisector Count is set to 128. Of course, this requires a 16K buffer.

The Multisector Count determines the number of operations to be performed by the following system calls:

F_READ
F_READRAND
F_WRITE
F_WRITERAND
F_WRITEZF

The Multisector Count affects BDOS error reporting for the BDOS Read and Write system calls. If an error interrupts these system calls when the Multisector Count is greater than one, they return the number of records successfully read or written in register H for all errors except for physical errors (A = 255).

Upon return, register A is set to zero if the specified value is in the range of 1 to 128. Otherwise register A is set to 0FFH.

```
+-----------------------------------------------+
!                                               !
!                   F_OPEN                      !
!                                               !
!          SYSTEM CALL  15:  OPEN FILE          !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  OFH                  !
!           Registers DE:  FCB Address          !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Directory Code       !
!           Register   H:  Physical or          !
!                          Extended Error       !
!                                               !
+-----------------------------------------------+
```

F_OPEN activates the FCB for a file that exists in

the disk directory under the currently active user number or user zero. The calling program passes the address of the FCB in register pair DE, with byte 0 of the FCB specifying the drive; bytes 1 through 11 specifying the filename and filetype; and byte 12 specifying the extent. Usually byte 12 of the FCB is initialized to zero.

If the file is password-protected in Read mode, the correct password must be placed in the first eight bytes of the current DMA, or have been previously established as the default password (see F_PASSWD). If the current record field of the FCB (cr) is set to 0FFH, F_OPEN returns the byte count of the last record of the file in the cr field.

You can set the last-record byte count for a file with F_ATTRIB. Note that the current record field of the FCB (cr) must be zeroed by the calling program before beginning read or write operations if the file is to be accessed sequentially from the first record.

If the current user is non-zero, and the file to be opened does not exist under the current user number, F_OPEN searches user zero for the file. If the file exists under user zero, and has the system (SYS) attribute (t2') set, the file is opened under user zero. Write operations are not supported for a file that is opened under user zero in this manner.

If the F_OPEN operation is successful, the user's FCB is activated for read and write operations. The relevant directory information is copied from the matching directory FCB into bytes d0 through dn of the FCB. If the file is opened under user zero when the current user number is not zero, interface attribute f8' is set to one in the user's FCB. In addition, if the referenced file is password-protected in Write mode, and the correct password was not passed in the DMA, or did not match the default password, interface attribute f7' is set to one. Write operations are not supported for an activated FCB if interface attribute f7' or f8' is true.

When F_OPEN is successful, it makes an Access date and time stamp for the opened file when the following conditions are satisfied: the referenced drive has a directory label that requests Access date and time stamping, and the FCB extent number field is zero.

Upon return, F_OPEN returns a directory code in register A with the value 00H if the open was successful, or FFH (255 decimal) if the file was not found. Register H is set to zero in both of these cases. If a physical or extended error was

encountered, F_OPEN performs different actions depending on the BDOS error mode (see SETERRMODE). If the BDOS error mode is in the default mode, a message identifying the error is displayed at the console and the program is terminated. Otherwise F_OPEN returns to the calling program with register A set to OFFH, and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
04 : Invalid drive error
07 : File password error
09 : ? in the FCB filename or filetype field
```

```
+-----------------------------------------------+
!                                               !
!                   F_PARSE                     !
!                                               !
!       SYSTEM CALL  152:  PARSE FILENAME       !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  98H                !
!                        DE: PFCB Address       !
!                                               !
!         Returned   Value:                     !
!             Registers HL:  Return code        !
!                            Parsed file        !
!                            control block      !
+-----------------------------------------------+
```

F_PARSE parses an ASCII file specification and prepares a File Control Block (FCB). The calling program passes the address of a data structure called the Parse Filename Control Block (PFCB) in register pair DE. The PFCB contains the address of the input ASCII filename string followed by the address of the target FCB as shown below:

```
PFCB:DW  INPUT  ; Address of input ASCII string
     DW  FCB    ; Address of target FCB
```

The maximum length of the input ASCII string to be parsed is 128 bytes. The target FCB must be 36 bytes in length.

F_PARSE assumes the input string contains file specifications in the following form:

{d:}filename{.typ}{;password}

where items enclosed in curly brackets are optional. F_PARSE also accepts isolated drive specifications

(d:) in the input string. When it encounters one, it sets the filename, filetype, and password fields in the FCB to blank.

F_PARSE parses the first file specification it finds in the input string, eliminating leading blanks and tabs. The system call then assumes that the file specification ends on the first delimiter it encounters that is out of context with the specific field it is parsing. For instance, if F_PARSE finds a colon and it is not the second character of the file specification, the colon delimits the entire file specification.

F_PARSE recognizes the following characters as delimiters:

space
tab
return
null
; (semicolon)--except before password field
= (equal)
< (less than)
> (greater than)
. (period)--except after filename and before filetype
: (colon)--except before filename and after drive
, (comma)
| (vertical bar)
[(left square bracket)
] (right square bracket)

If F_PARSE encounters a nongraphic character not listed above, in the range 1 through 31, it treats the character as an error. F_PARSE initializes the specified FCB shown in the table that follows:

TABLE B-17. FCB FORMAT

Location	Contents
byte 0	The drive field is set to the specified drive. If the drive is not specified, the default drive code is used. 0 = default, 1 = A, 2 = B.
byte 1 - 8	The name is set to the specified filename. All letters are converted to uppercase. If the name is not eight characters long, the remaining bytes in the filename field are padded with blanks. If the filename has an asterisk (*), all remaining bytes in the filename field are filled in with question marks (?). An error occurs if the filename is more than eight bytes long.
byte 9 - 11	The type is set to the specified filetype. If no filetype is specified, the type field is initialized to blanks. All letters are converted to uppercase. If the type is not three characters long, the remaining bytes in the filetype field are padded with blanks. If an asterisk (*) occurs, all remaining bytes are filled in with question marks (?). An error occurs if the type field is more than three bytes long.
byte 12 - 15	Filled in with zeros.
byte 16 - 23	The password field is set to the specified password. If no password is specified, it is initialized to blanks. If the password is less than eight characters long, remaining bytes are padded with blanks. All letters are converted to uppercase. If the password field is more than eight bytes long, an error occurs. **Note:** A blank in the first position of the password field implies no password was specified.
byte 24 - 31	Reserved for system use.

If an error occurs, F_PARSE returns an 0FFFFH in register pair HL.

On a successful parse, F_PARSE checks the next item in the input string. It skips over trailing blanks and tabs and looks at the next character. If the character is a null or carriage return, it returns a 0 indicating the end of the input string. If the character is a delimiter, it returns the address of the delimiter. If the character is not a delimiter, it returns the address of the first trailing blank or tab.

If the first nonblank or nontab character in the input string is a null (0) or carriage return, F_PARSE returns a zero indicating the end of string.

If F_PARSE is to be used to parse a subsequent file specification in the input string, the returned address must be advanced over the delimiter before placing it in the PFCB.

F_PARSE also excludes all control characters from the file fields, and translates all lowercase letters to upper-case.

Avoid using parentheses and the backslash character (\) in the filename and filetype fields because they are commonly used delimiters. Use asterisk and question mark characters (* and ?) only to make an ambiguous file reference. When F_PARSE encounters an * in a filename or filetype field, it pads the remainder of the field with question marks. For example, a filename of X*.* is parsed to X???????.???. The BDOS F_SFIRST, F_SNEXT, and F_DELETE system calls treat a ? in the filename and type fields as follows: A ? in any position matches the corresponding field of any directory entry belonging to the current user number. Thus, a search operation for X???????.??? finds all the current user files on the directory beginning in X. Most other file-related BDOS system calls treat the presence of a ? in the filename or filetype field as an error.

```
+-----------------------------------------------+
!                                               !
!                  F_PASSWD                     !
!                                               !
!    SYSTEM CALL  106:  SET DEFAULT PASSWORD    !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  6AH                 !
!            Register  DE:  Password Address    !
!                                               !
!        Returned   Value:  None                !
!                                               !
+-----------------------------------------------+
```

F_PASSWD allows a program to specify a password value before a file protected by the password is accessed. When the file system accesses a password-protected file, it checks the current DMA and the default password for the correct value. If either value matches the file's password, full access to the file

is allowed.

Note: This system call performs no action in nonbanked CP/M Plus systems because file passwords are not supported.

To make an F_PASSWD call, the calling program sets register pair DE to the address of an 8-byte field containing the password.

```
+-----------------------------------------------+
!                                               !
!                  F_RANDREC                    !
!                                               !
!      SYSTEM CALL  36:  SET RANDOM RECORD      !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  24H                 !
!            Registers DE:  FCB Address         !
!                                               !
!        Returned   Value:                      !
!            Random-Record Field Set            !
!                                               !
+-----------------------------------------------+
```

F_RANDREC sets the random-record number of the next record to be accessed from a file that has been read or written sequentially to a particular point. This value is returned in the random-record field (bytes r0, r1, and r2) of the FCB addressed by the register pair DE. F_RANDREC can be useful in two ways.

First, it is often necessary to initially read and scan a sequential file to extract the positions of various key fields. As each key is encountered, F_RANDREC is called to compute the random-record position for the data corresponding to this key. If the data unit size is 128 bytes, the resulting record number, minus one, is placed into a table with the key for later retrieval. After scanning the entire file and tabularizing the keys and their record numbers, you can move directly to a particular record by performing a random read using the corresponding random-record number that you saved earlier. The scheme is easily generalized when variable record lengths are involved, because the program need only store the buffer-relative byte position along with the key and record number to find the exact starting position of the keyed data at a later time.

A second use of F_RANDREC occurs when switching from a sequential read or write over to random read

or write. A file is sequentially accessed to a particular point in the file, then F_RANDREC is called to set the record number, and subsequent random read and write operations continue from the next record in the file.

```
+-----------------------------------------------+
!                                               !
!                   F_READ                      !
!                                               !
!      SYSTEM CALL  20:  READ SEQUENTIAL        !
!                                               !
+-----------------------------------------------+
!                                               !
!     Entry Parameters:                         !
!         Register   C:  14H                    !
!         Registers DE:  FCB Address            !
!                                               !
!     Returned   Value:                         !
!         Register   A:  Error Code             !
!         Register   H:  Physical Error         !
!                                               !
+-----------------------------------------------+
```

F_READ reads the next 1 to 128 128-byte records from a file into memory beginning at the current DMA address. The BDOS Multisector Count (see the F_MULTISEC system call) determines the number of records to be read. The default is one record. The FCB addressed by register pair DE must have been previously activated by an F_OPEN or F_MAKE system call.

F_READ reads each record from byte cr of the extent, then automatically increments the cr field to the next record position. If the cr field overflows, then F_READ automatically opens the next logical extent and resets the cr field to 0 in preparation for the next read operation. The calling program must set the cr field to 0 following the F_OPEN call if the intent is to read sequentially from the beginning of the file.

Upon return, F_READ sets register A to zero if the read operation is successful. Otherwise register A contains an error code identifying the error as shown below:

```
  01 : Reading unwritten data (end of file)
  09 : Invalid FCB
  10 : Media change occurred
 255 : Physical error; refer to register H
```

Error Code 01 is returned if no data exists at the

next record position of the file. Usually the no-data situation is encountered at the end of a file. However, it can also occur if an attempt is made to read a data block that has not been previously written, or an extent which has not been created. These situations are usually restricted to files created or appended with F_WRITERAND and F_WRITEZF.

Error Code 09 is returned if the FCB is invalidated by a previous BDOS close call that returns an error.

Error Code 10 is returned if a media change occurs on the drive after the referenced FCB is activated by an F_OPEN or F_MAKE system call.

Error Code 255 is returned if a physical error is encountered and the BDOS error mode is Return Error mode or Return and Display Error mode (see SETERRMODE). If the error mode is the default mode, a message identifying the physical error is displayed at the console and the calling program is terminated. When a physical error is returned to the calling program, register H contains one of the following error codes:

01 : Disk I/O error
04 : Invalid drive error

On all error returns except for physical error returns, A = 255, F_READ sets register H to the number of records successfully read before the error is encountered. This value can range from 0 to 127 depending on the current BDOS Multisector Count. It is always set to zero when the Multisector Count is equal to one.

F_READRAND

SYSTEM CALL 33: READ RANDOM

Entry Parameters:
Register C: 21H
Registers DE: FCB Address

Returned Value:
Register A: Error Code
Register H: Physical Error

F_READRAND is similar to F_READ except that the read operation takes place at a particular random-record number, selected by the 24-bit value constructed from the three-byte r0, r1, r2 field beginning at position 33 (21H) of the FCB. Note that the sequence of 24 bits is stored with the least-significant byte first (r0), the middle byte next (r1), and the high byte last (r2). The random-record number can range from 0 to 262,143. This corresponds to a maximum value of 3 in byte r2.

To read a file with F_READRAND, the calling program must first open the base extent (extent 0). This ensures that the FCB is properly initialized for subsequent random-access operations. The base extent may or may not contain any allocated data. F_READRAND reads the record specified by the random-record field into the current DMA address. The system call automatically sets the logical extent and current record values, but unlike the F_READ system call, it does not advance the current record number. Thus, a subsequent F_READRAND call rereads the same record. After a random read operation, a file can be accessed sequentially, starting from the current randomly accessed position. However, the last randomly accessed record is reread or rewritten when switching from random to sequential mode.

If the BDOS Multisector Count is greater than one (see F_MULTISEC), F_READRAND reads multiple consecutive records into memory beginning at the current DMA. The r0, r1, and r2 field of the FCB is automatically incremented to read each record. However, the FCBs random-record number is restored to the first record's value upon return to the calling program.

Upon return, F_READRAND sets register A to zero if the read operation was successful. Otherwise register A contains one of the following error codes:

```
  01 : Reading unwritten data (end of file)
  03 : Cannot close current extent
  04 : Seek to unwritten extent
  06 : Random-record number out of range
  10 : Media change occurred
 255 : Physical error : refer to register H
```

Error Code 01 is returned if no data exists at the next record position of the file. Usually the no-data situation is encountered at the end of a file. However, it can also occur if an attempt is made to read a data block that has not been previously written.

Error Code 03 is returned when F_READRAND cannot close the current extent prior to moving to a new extent.

Error Code 04 is returned when an F_READRAND operation accesses an extent that has not been created.

Error Code 06 is returned when byte 35 (r2) of the referenced FCB is greater than 3.

Error Code 10 is returned if a media change occurs on the drive after the referenced FCB is activated by a BDOS open or make call.

Error Code 255 is returned if a physical error is encountered, and the BDOS error mode is one of the return modes (see F_ERRMODE). If the error mode is the default mode, a message identifying the physical error is displayed at the console and the calling program is terminated. When a physical error is returned to the calling program, register H contains one of the following error codes:

01 : Disk I/O error
04 : Invalid drive error

On all error returns except for physical errors (A = 255), F_READRAND sets register H to the number of records successfully read before the error is encountered. This value can range from 0 to 127 depending on the current BDOS Multisector Count. It is always set to zero when the Multisector Count is equal to 1.

F_RENAME

SYSTEM CALL 23: RENAME FILE

Entry Parameters:
Register C: 17H
Registers DE: FCB Address

Returned Value:
Register A: Directory Code
Register H: Physical or Extended Error

F_RENAME uses the FCB addressed by register pair DE to change all directory entries of the file specified by the file specification in the first 16 bytes of the FCB to the file specification in the second 16 bytes. If the file specified by the first filespec is password-protected, the correct password must be placed in the first eight bytes of the current DMA buffer, or have been previously established as the default password (see F_PASSWD).

The calling program must also ensure that the filenames specified in the FCB are valid and unambiguous, and that the new filename does not already exist on the drive. F_RENAME uses the dr code at byte zero of the FCB to select the drive. The drive code at byte 16 (10H) of the FCB is ignored.

Upon return, F_RENAME returns a Directory Code in register A with the value zero if the rename is successful, or OFFH (255 decimal) if the file named by the first filename in the FCB is not found. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_RENAME performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the program is terminated. Otherwise F_RENAME returns to the calling program with register A set to OFFH and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
02 : Read-Only disk
03 : Read-Only file
04 : Invalid drive error
07 : File password error
08 : File already exists
09 : ? in filename or filetype field
```

```
+-----------------------------------------------+
!                                               !
!                  F_SFIRST                     !
!                                               !
!       SYSTEM CALL  17:  SEARCH FOR FIRST      !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  11H                  !
!           Registers DE:  FCB Address          !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Directory Code       !
!           Register   H:  Physical Error       !
!                                               !
+-----------------------------------------------+
```

F_SFIRST scans the directory for a match with the FCB addressed by register pair DE. Two types of searches can be performed. For standard searches, the calling program initializes bytes 0 through 12 of the referenced FCB, with byte 0 specifying the drive directory to be searched; bytes 1 through 11 specifying the file or files to be searched for; and byte 12 specifying the extent. Usually byte 12 is set to zero. An ASCII question mark (63 decimal or 3FH) in any of the bytes 1 through 12 matches all entries on the directory in the corresponding position. This facility, called ambiguous reference, can be used to search for multiple files on the directory. When called in the standard mode, the F_SEARCHF system call scans for the first file entry in the specified directory that matches the FCB and belongs to the current user number.

If byte 0 of the referenced FCB is set to a question mark, the F_SFIRST system call ignores the remainder of the referenced FCB, and locates the first directory entry residing on the current default drive. All remaining directory entries can be located by making multiple F_SNEXT calls. This type of search operation is usually not made by application programs, but it does provide complete flexibility to scan all current directory values.

Note: This type of search operation must be performed to access a drive's directory label.

Upon return, F_SFIRST returns a Directory Code in register A with the value 0 to 3 if the search is successful, or 0FFH (255 Decimal) if a matching directory entry is not found. Register H is set to zero in both of these cases. For successful searches, the current DMA is also filled with the

directory record containing the matching entry, and the relative starting position is A * 32 (that is, rotate the A register left five bits, or ADD A five times). Although it is usually not required for application programs, the directory information can be extracted from the buffer at this position.

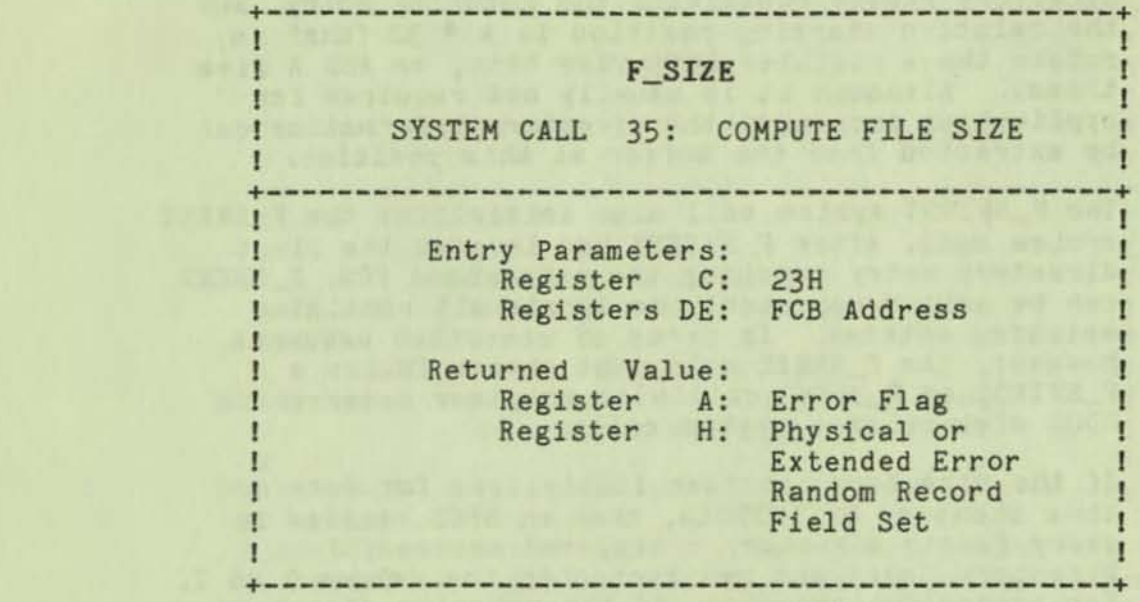
The F_SFIRST system call also initializes the F_SNEXT system call. After F_SFIRST has located the first directory entry matching the referenced FCB, F_SNEXT can be called repeatedly to locate all remaining matching entries. In terms of execution sequence, however, the F_SNEXT call must either follow a F_SFIRST or F_SNEXT call with no other intervening BDOS disk-related system calls.

If the directory has been initialized for date and time stamping by INITDIR, then an SFCB resides in every fourth directory entry, and successful Directory Codes are restricted to the values 0 to 2. For successful searches, if the matching directory record is an extent-zero entry, and if an SFCB resides at offset 96 within the current DMA (contents of DMA Address + 96 = 21H), the SFCB contains the date and time stamp information and password mode for the file. This information is located at the relative starting position of 97 + (A * 10) within the current DMA in the following format:

```
0 - 3 : Create or Access Date- and
        Time-Stamp Field
4 - 7 : Update Date- and Time-Stamp Field
8     : Password Mode Field
```

If a physical error is encountered, F_SFIRST performs different actions depending on the BDOS error mode (see F_ERRMODE system call). If the BDOS error mode is in the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_SFIRST returns to the calling program with register A set to 0FFH, and register H set to one of the following physical error codes:

```
01 : Disk I/O error
04 : Invalid drive error
```



F_SIZE determines the virtual file size which is, in effect, the address of the record immediately following the end of the file. The virtual size of a file corresponds to the physical size if the file is written sequentially. If the file is written in random mode, gaps might exist in the allocation, and the file might contain fewer records than the indicated size. For example, if a single record with record number 262,143 (the CP/M Plus maximum) is written to a file using F_WRITERAND, then the virtual size of the file is 262,144 records even though only one data block is actually allocated.

To compute file size, the calling program passes the address of an FCB in random mode format (bytes r0, r1, and r2 present) in register pair DE. Note that the FCB must contain an unambiguous filename and filetype. F_SIZE sets the random-record field of the FCB to the random-record number, plus one, of the last record in the file. If the r2 byte is set to 04H, then the file contains the maximum record count 262,144.

A program can append data to the end of an existing file by calling F_SIZE to set the random-record position to the end of file, and then performing a sequence of random writes starting at the preset record address.

Note: The BDOS does not require that the file be open to use F_SIZE. However, if the file has been written to, it must be closed before calling F_SIZE. Otherwise an incorrect file size might be returned.

Upon return, F_SIZE returns a zero in register A if the file specified by the referenced FCB is found, or an 0FFH in register A if the file is not found. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_SIZE performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the program is terminated. Otherwise F_SIZE returns to the calling program with register A set to 0FFH, and register H set to one of the following physical or extended errors:

01 : Disk I/O error
04 : Invalid drive error

F_SNEXT

SYSTEM CALL 18: SEARCH FOR NEXT

Entry Parameters:
Register C: 12H

Returned Value:
Register A: Directory Code
Register H: Physical Error

F_SNEXT is identical to the F_SFIRST system call, except that the directory scan continues from the last entry that was matched. F_SNEXT returns a Directory Code in register A, analogous to F_SFIRST.

Note: In execution sequence, a F_SNEXT call must follow either a F_SFIRST or another F_SNEXT call with no other intervening BDOS disk-related system calls.

```
+-----------------------------------------------+
!                                               !
!                  F_TIMEDATE                   !
!                                               !
!   SYSTEM CALL  102:  READ FILE DATE STAMPS    !
!                      AND PASSWORD MODE        !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  66H                 !
!            Register  DE:  FCB Address         !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Directory Code      !
!            Register   H:  Physical Error      !
!                                               !
+-----------------------------------------------+
```

F_TIMEDATE returns the date- and time-stamp information and password mode for the specified file in byte 12 and bytes 24 through 32 of the specified FCB. The calling program passes the address of an FCB in which the drive, filename, and filetype fields have been defined in register pair DE.

If F_TIMEDATE is successful, it sets the following fields in the referenced FCB:

byte 12 (0CH): Password mode field
bit 7--Read mode
bit 6--Write mode
bit 4--Delete mode

Byte 12 equal to zero indicates the file has not been assigned a password.

byte 24 - 27 (18H - 11BH): Create or Access time-stamp field
byte 28 - 31 (1CH - 1FH) : Update time-stamp field

The date-stamp fields are set to binary zeros if a stamp has not been made. The format of the time-stamp fields is the same as the format of the date and time structure described in T_SET.

Upon return, F_TIMEDATE returns a Directory Code in register A with the value zero if the system call is successful, or 0FFH (255 decimal) if the specified file is not found. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_TIMEDATE performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is in the default mode, a message

identifying the error is displayed at the console and the calling program is terminated. Otherwise F_TIMEDATE returns to the calling program with register A set to 0FFH and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
04 : Invalid drive error
09 : ? in filename or filetype field
```

```
+-----------------------------------------------+
!                                               !
!                  F_TRUNCATE                   !
!                                               !
!        SYSTEM CALL  99:  TRUNCATE FILE        !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  63H                 !
!                       DE:  FCB Address        !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Directory Code      !
!            Register   H:  Extended or         !
!                           Physical Error      !
!                                               !
+-----------------------------------------------+
```

F_TRUNCATE sets the last record of a file to the random-record number contained in the referenced FCB. The calling program passes the address of the FCB in register pair DE, with byte 0 of the FCB specifying the drive; bytes 1 through 11 specifying the filename and filetype; and bytes 33 through 35 (r0, r1, and r2) specifying the last record number of the file. The last record number is a 24-bit value stored with the least-significant byte first (r0), the middle byte next (r1), and the high byte last (r2). This value can range from 0 to 262,143, which corresponds to a maximum value of 3 in byte r2.

If the file specified by the referenced FCB is password-protected, the correct password must be placed in the first eight bytes of the current DMA buffer, or have been previously established as the default password (see F_PASSWD).

F_TRUNCATE requires that the file specified by the FCB not be open, particularly if the file has been written to. In addition, any activated FCBs naming the file are not valid after F_TRUNCATE is called. Close your file before calling F_TRUNCATE and then reopen it after the call to continue processing on

the file.

F_TRUNCATE also requires that the random-record number field of the referenced FCB specify a value less than the current file size. In addition, if the file is sparse, the random-record field must specify a record in a region of the file where data exists.

Upon return, F_TRUNCATE returns a Directory Code in register A with the value zero if F_TRUNCATE is successful, or OFFH (255 decimal) if the file is not found or the record number is invalid. Register H is set to zero in both of these cases. If a physical or extended error is encountered, F_TRUNCATE performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is in the 0 default mode, a message identifying the error is displayed at the console and the program is terminated. Otherwise F_TRUNCATE returns to the calling program with register A set to OFFH and register H set to one of the following physical or extended error codes:

```
01 : Disk I/O error
02 : Read-Only disk
03 : Read-Only file
04 : Invalid drive error
07 : File password error
09 : ? in filename or filetype field
```

F_TSTWRITE

SYSTEM CALL 41: TEST AND WRITE RECORD

Entry Parameters:		
	Register C:	29H
	Register DE:	FCB Address
Returned Value:		
	Register A:	Error Code
	Register H:	Physical Error

F_TSTWRITE is an MP/M II system call that is not supported under CP/M Plus. If called, F_TSTWRITE returns with register A set to OFFH and register H set to zero.

```
+-----------------------------------------------+
!                                               !
!                  F_UNLOCK                     !
!                                               !
!        SYSTEM CALL  43:  UNLOCK RECORD        !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  2BH                 !
!                       DE:  FCB Address        !
!                                               !
!        Returned   Value:                      !
!            Register   A:  00H                 !
!                                               !
+-----------------------------------------------+
```

F_UNLOCK is an MP/M II system call that is supported under CP/M Plus only to provide compatibility between CP/M Plus and MP/M. It is intended for use in situations where more than one running program has Read-Write access to a common file. Because CP/M Plus is a single-user operating system in which only one program can run at a time, this situation cannot occur. Thus, under CP/M Plus, F_UNLOCK performs no action except to return the value 00H in register A indicating that the record unlock operation is successful.

```
+-----------------------------------------------+
!                                               !
!                  F_USERNUM                    !
!                                               !
!       SYSTEM CALL  32:  SET/GET USER CODE     !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  20H                  !
!           Register   E:  0FFH (get) or        !
!                          User Code (set)      !
!                                               !
!       Returned   Value:                       !
!           Register   A:  Current Code or      !
!                          (no value)           !
!                                               !
+-----------------------------------------------+
```

A program can change or interrogate the currently active user number by calling F_USERNUM. If register E = 0FFH, then the value of the current user number is returned in register A where the value is in the range of 0 to 15. If register E is not 0FFH, then the current user number is changed to the value of E

(module 16). In other words, only the value of the four least-significant bits of register E, 0 through 15 (0 - 0FH), is used to set the user number.

```
+-----------------------------------------------+
!                                               !
!                   F_WRITE                     !
!                                               !
!     SYSTEM CALL  21:  WRITE SEQUENTIAL        !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  15H                !
!             Registers DE:  FCB Address        !
!                                               !
!         Returned   Value:                     !
!             Register   A:  Error Code         !
!             Register   H:  Physical Error     !
!                                               !
+-----------------------------------------------+
```

F_WRITE writes 1 to 128 128-byte data records, beginning at the current DMA address, into the file named by the FCB addressed in register pair DE. The BDOS Multisector Count (see F_MULTISEC) determines the number of 128-byte records that are written. The default is one record. The referenced FCB must have been previously activated by an F_OPEN or F_MAKE system call.

F_WRITE places the record into the file at the position indicated by the cr byte of the FCB, and then automatically increments the cr byte to the next record position. If the cr field overflows, the system call automatically opens, or creates the next logical extent, and resets the cr field to zero in preparation for the next write operation. If F_WRITE is used to write to an existing file, then the newly written records overlay those already existing in the file. The calling program must set the cr field to zero following an Open or Make call if the intent is to write sequentially from the beginning of the file.

F_WRITE makes an Update date and time for the file if the following conditions are satisfied: the referenced drive has a directory label that requests date and time stamping, and the file has not already been stamped for update by a previous F_MAKE or F_WRITE system call.

Upon return, F_WRITE sets register A to zero if the write operation is successful. Otherwise register A contains an error code identifying the error as shown

below:

```
 01 : No available directory space
 02 : No available data block
 09 : Invalid FCB
 10 : Media change occurred
255 : Physical error : refer to register H
```

Error Code 01 is returned when F_WRITE attempts to create a new extent that requires a new directory entry, and no available directory entries exist on the selected disk drive.

Error Code 02 is returned when F_WRITE attempts to allocate a new data block to the file, and no unallocated data blocks exist on the selected disk drive.

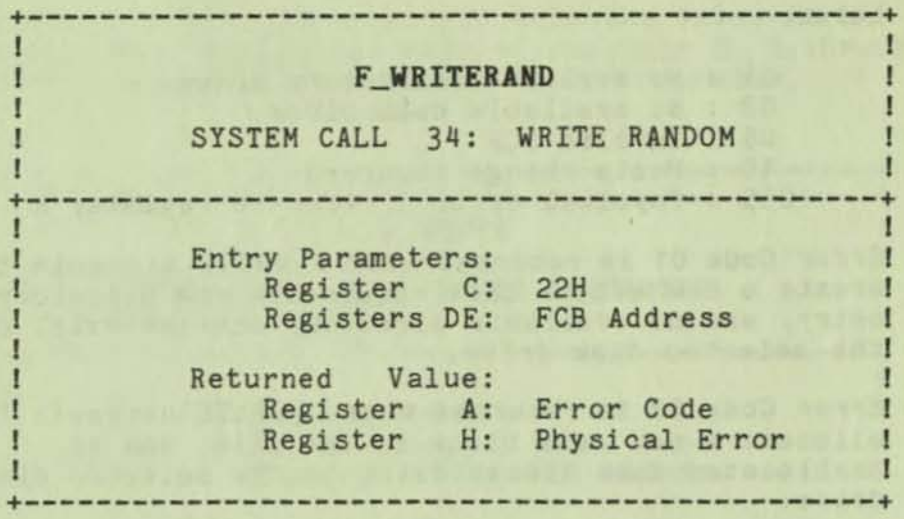
Error Code 09 is returned if the FCB is invalidated by a previous BDOS close call that returns an error.

Error Code 10 is returned if a media change occurs on the drive after the referenced FCB is activated by a BDOS open or make call.

Error Code 255 is returned if a physical error is encountered and the BDOS error mode is Return Error mode, or Return and Display Error mode (see F_ERRMODE). If the error mode is the default mode, a message identifying the physical error is displayed at the console and the calling program is terminated. When a physical error is returned to the calling program, register H contains one of the following error codes:

```
01 : Disk I/O error
02 : Read-Only disk
03 : Read-Only file or
     File open from user zero when
     the current user number is non-zero or
     File password-protected in Write mode
04 : Invalid drive error
```

On all error returns, except for physical error returns (A = 255), F_WRITE sets register H to the number of records successfully written before the error was encountered. This value can range from 0 to 127 depending on the current BDOS Multisector Count. It is always set to zero when the Multisector Count is set to one.



F_WRITERAND is analogous to F_READRAND, except that data is written to the disk from the current DMA address. If the disk extent or data block where the data is to be written is not already allocated, the BDOS automatically performs the allocation before the write operation continues.

To write to a file using F_WRITERAND, the calling program must first open the base extent (extent 0). This ensures that the FCB is properly initialized for subsequent random-access operations. If the file is empty, the calling program must create the base extent with F_MAKE before calling F_WRITERAND. The base extent might or might not contain any allocated data, but it does record the file in the directory so that the file can be displayed by the DIR utility.

F_WRITERAND sets the logical extent and current record positions to correspond with the random record being written, but does not change the random-record number. Thus, F_READ or F_WRITE operations can follow a F_WRITERAND, with the current record being reread or rewritten as the calling program switches from random to sequential mode.

F_WRITERAND makes an Update date and time stamp for the file if the following conditions are satisfied: the referenced drive has a directory label that requests Update date and time stamping if the file has not already been stamped for update by a previous F_MAKE or F_WRITE call.

If the BDOS Multisector Count is greater than one (see F_MULTISEC), F_WRITERAND reads multiple consecutive records into memory beginning at the current DMA. The r0, r1, and r2 field of the FCB is automatically incremented to write each record.

However, the FCB's random-record number is restored to the first record's value when it returns to the calling program. Upon return, F_WRITERAND sets register A to zero if the write operation is successful. Otherwise register A contains one of the following error codes:

```
  02 : No available data block
  03 : Cannot close current extent
  05 : No available directory space
  06 : Random-record number out of range
  10 : Media change occurred
 255 : Physical error : refer to register H
```

Error Code 02 is returned when F_WRITERAND attempts to allocate a new data block to the file and no unallocated data blocks exist on the selected disk drive.

Error Code 03 is returned when F_WRITERAND cannot close the current extent prior to moving to a new extent.

Error Code 05 is returned when F_WRITERAND attempts to create a new extent that requires a new directory entry and no available directory entries exist on the selected disk drive.

Error Code 06 is returned when byte 35 (r2) of the referenced FCB is greater than 3.

Error Code 10 is returned if a media change occurs on the drive after the referenced FCB is activated by a BDOS open or make call.

Error Code 255 is returned if a physical error is encountered and the BDOS error mode is one of the return modes (see F_ERRMODE). If the error mode is the default mode, a message identifying the physical error is displayed at the console and the calling program is terminated. When a physical error is returned to the calling program, it is identified by register H as shown below:

```
  01 : Disk I/O error
  02 : Read-Only disk
  03 : Read-Only file or
       File open from user zero when the current
       user number is non-zero or
       File password-protected in write mode
  04 : Invalid drive error
```

On all error returns, except for physical errors (A = 255), F_WRITERAND sets register H to the number of records successfully written before the error is

encountered. This value can range from 0 to 127 depending on the current BDOS Multisector Count. It is always set to zero when the Multisector Count is equal to one.

```
+-----------------------------------------------+
!                                               !
!                F_WRITEXFCB                    !
!                                               !
!      SYSTEM CALL  103:  WRITE FILE XFCB       !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  67H                 !
!            Register  DE:  FCB Address         !
!                                               !
!        Returned   Value:                      !
!            Register   A:  Directory Code      !
!            Register   H:  Physical Error      !
!                                               !
+-----------------------------------------------+
```

F_WRITEXFCB creates a new XFCB or updates the existing XFCB for the specified file. The calling program passes in register pair DE the address of an FCB in which the drive, name, type, and extent fields have been defined. The extent field specifies the password mode and whether a new password is to be assigned to the file. The format of the extent byte is shown below:

```
FCB byte 12 (0CH) (ex) : XFCB password mode
     bit  7--Read mode
     bit  6--Write mode
     bit  5--Delete mode
     bit  0--Assign new password to the file
```

If the specified file is currently password-protected, the correct password must reside in the first eight bytes of the current DMA, or have been previously established as the default password (see F_PASSWD). If bit 0 is set to 1, the new password must reside in the second eight bytes of the current DMA.

Upon return, F_WRITEXFCB returns a Directory Code in register A with the value zero if the XFCB create or update is successful; or 0FFH (255 decimal) if no directory label exists on the specified drive; or the file named in the FCB is not found; or no space exists in the directory to create an XFCB.

F_WRITEXFCB also returns with 0FFH in register A if

passwords are not enabled by the referenced directory's label. Register H is set to zero in all of these cases. If a physical or extended error is encountered, F_WRITEXFCB performs different actions depending on the BDOS error mode (see F_ERRMODE). If the BDOS error mode is the default mode, a message identifying the error is displayed at the console and the calling program is terminated. Otherwise F_WRITEXFCB returns to the calling program with register A set to 0FFH and register H set to one of the following physical or extended error codes:

01 : Disk I/O error
02 : Read-Only disk
04 : Invalid drive error
07 : File password error
09 : ? in filename or filetype field

```
+-----------------------------------------------+
!                                               !
!                  F_WRITEZF                    !
!                                               !
!       SYSTEM CALL  40:  WRITE RANDOM WITH     !
!                         ZERO FILL             !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  28H                !
!             Register  DE:  FCB Address        !
!                                               !
!         Returned   Value:                     !
!             Register   A:  Error Code         !
!             Register   H:  Physical Error     !
!                                               !
+-----------------------------------------------+
```

F_WRITEZF is identical to F_WRITERAND, with the exception that a previously unallocated data block is filled with zeros before the record is written. If this system call has been used to create a file, records accessed by a read random operation that contain all zeros identify unwritten random-record numbers. Unwritten random records in allocated data blocks of files created using F_WRITERAND contain uninitialized data.

BDOS List Device System Calls

```
+-----------------------------------------------+
!                                               !
!                  L_WRITE                      !
!                                               !
!          SYSTEM CALL  5:  LIST OUTPUT         !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  05H                  !
!           Register   E:  ASCII Character      !
!                                               !
+-----------------------------------------------+
```

L_WRITE sends the ASCII character in register E to the logical list device (LST:).

```
+-----------------------------------------------+
!                                               !
!                 L_WRITEBLK                    !
!                                               !
!          SYSTEM CALL  112:  LIST BLOCK        !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  70H                !
!             Register  DE:  CCB Address        !
!                                               !
!         Returned   Value:  none               !
!                                               !
+-----------------------------------------------+
```

L_WRITEBLK sends the character string located by the Character Control Block (CCB) addressed in register pair DE to the logical list device (LST:). The CCB format is:

byte 0 - 1 : Address of character string (word value)
byte 2 - 3 : Length of character string (word value)

BDOS Program System Calls

```
+-----------------------------------------------+
!                                               !
!                   P_CHAIN                     !
!                                               !
!       SYSTEM CALL  47:  CHAIN TO PROGRAM      !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  2FH                !
!                        E:  Chain Flag         !
!                                               !
+-----------------------------------------------+
```

P_CHAIN provides a means of chaining from one program to the next without operator intervention. The calling program must place a command line terminated by a null byte (00H) in the default DMA buffer. If register E is set to 0FFH, the CCP initializes the default drive and user number to the current program values when it passes control to the specified transient program. Otherwise these parameters are set to the default CCP values.

Note: P_RETCODE can be used to pass a two-byte value to the chained program.

P_CHAIN does not return any values to the calling program and any errors encountered are handled by the CCP.

```
+-----------------------------------------------+
!                                               !
!                   P_LOAD                      !
!                                               !
!         SYSTEM CALL  59:  LOAD OVERLAY        !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  3BH                !
!             Register  DE:  FCB Address        !
!                                               !
!         Returned   Value:                     !
!             Register   A:  Error Code         !
!             Register   H:  Physical Error     !
!                                               !
+-----------------------------------------------+
```

Only transient programs with an RSX header can use P_LOAD because it is supported by the LOADER module.

The calling program must have a header to force the LOADER to remain resident after the program is loaded.

P_LOAD loads either an absolute or relocatable module. Relocatable modules are identified by a filetype of PRL. P_LOAD does not call the loaded module.

The referenced FCB must be successfully opened before P_LOAD is called. The load address is specified in the first two random-record bytes of the FCB (r0 and r1). The LOADER returns an error if the load address is less than 100H; or if performing the requested load operation would overlay the LOADER or any other Resident System Extensions that have been previously loaded.

When loading relocatable files, the LOADER requires enough room at the load address for the complete PRL file including the header and bit map (see Appendix B). Otherwise an error is returned. P_LOAD also returns an error on PRL file-load requests if the specified load address is not on a page boundary.

Upon return, P_LOAD sets register A to zero if the load operation is successful. If the LOADER RSX is not resident in memory because the calling program did not have a RSX header, the BDOS returns with register A set to OFFH and register H set to zero. If the LOADER detects an invalid load address, or if insufficient memory is available to load the overlay, P_LOAD returns with register A set to OFEH. All other error returns are consistent with the error codes returned by BDOS F_READ.

P_RETCODE

SYSTEM CALL 108: GET/SET PROGRAM RETURN CODE

Entry Parameters:
- Register C: 6CH
- Register DE: OFFFFH (Get) or Program Return Code (Set)

Returned Value:
- Register HL: Program Return Code or (no value)

CP/M Plus allows programs to set a return code before terminating. This provides a mechanism for programs to pass an error code or value to a following job step in batch environments. For example, Program Return Codes are used by the CCP in CP/M Plus's conditional command-line batch facility. Conditional command lines are command lines that begin with a colon (:). The execution of a conditional command depends on the successful execution of the preceding command. The CCP tests the return code of a terminating program to determine whether it successfully completed or terminated in error. Program return codes can also be used by programs to pass an error code or value to a chained program (see P_CHAIN, Chain to Program).

The CCP has a conditional command facility that uses the Program Return Code. If a command line SUBMITted to the CCP by the SUBMIT utility begins with a colon, the CCP skips execution of the command if the previous command set an unsuccessful Program Return Code. In the following example, the SUBMIT utility sends a command sequence to the CCP.

```
A>SUBMIT SUBFILE

A>COMPUTE RESULTS.DAT

A>:REPORT RESULTS.DAT
```

The CCP does not execute the REPORT command if the COMPUTE command sets an unsuccessful Program Return Code.

A program can set or interrogate the Program Return Code by calling P_RETCODE. If register pair DE = OFFFFH, then the current Program Return Code is returned in register pair HL. Otherwise P_RETCODE sets the Program Return Code to the value contained in register pair DE. Program Return Codes are defined in the table below.

TABLE B-18. PROGRAM RETURN CODES

Code	Meaning
0000 - FEFF	Successful return.
FF00 - FFFE	Unsuccessful return.
0000	The CCP initializes the Program Return Code to zero unless the program is loaded as the result of program chain.
FF80 - FFFC	Reserved.
FFFD	The program is terminated because of a fatal BDOS error.
FFFE	The program is terminated by the BDOS because the user typed a CTRL-C.

```
+-----------------------------------------------+
!                                               !
!                  P_TERMCPM                    !
!                                               !
!        SYSTEM CALL  0:  SYSTEM RESET          !
!                                               !
+-----------------------------------------------+
!                                               !
!           Entry Parameters:                   !
!               Register   C:  00H              !
!                                               !
+-----------------------------------------------+
```

P_TERMCPM terminates the calling program and returns control to the CCP via a warm-start sequence. Calling this system call has the same effect as a jump to location 0000H of Page Zero.

Note: The disk subsystem is not reset by P_TERMCPM under CP/M Plus. The calling program can pass a return code to the CCP by calling P_RETCODE prior to making a P_TERMCPM call or jumping to location 0000H.

When the CCP loads a transient program, the LOADER module sets the stack pointer to a 16-level stack, and then pushes the address 0000H onto the stack. Thus, an immediate return to the system is equivalent to a jump to 0000H. However, most transient programs set up their own stack, and terminate execution by making a P_TERMCPM call or by jumping to location 0000H.

System Calls

```
+-----------------------------------------------+
!                                               !
!                  S_BDOSVER                    !
!                                               !
!    SYSTEM CALL  12:  RETURN VERSION NUMBER    !
!                                               !
+-----------------------------------------------+
!                                               !
!       Entry Parameters:                       !
!           Register   C:  OCH                  !
!                                               !
!       Returned   Value:                       !
!           Registers HL:  Version Number       !
!                                               !
+-----------------------------------------------+
```

S_BDOSVER provides information that allows version-independent programming. It returns a two-byte value in register pair HL: H contains 00H for CP/M and L contains 31H, the BDOS file-system version number. S_BDOSVER is useful for writing applications programs that must run on multiple versions of CP/M and MP/M.

```
+-----------------------------------------------+
!                                               !
!                   S_BIOS                      !
!                                               !
!       SYSTEM CALL  50:  DIRECT BIOS CALLS     !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  32H                !
!             Register  DE:  BIOS PB Address    !
!                                               !
!         Returned   Value:  BIOS RETURN        !
!                                               !
+-----------------------------------------------+
```

S_BIOS provides a direct BIOS call through the BDOS to the BIOS. The calling program passes the address of a data structure called the BIOS Parameter Block (BIOSPB) in register pair DE. The BIOSPB contains the BIOS function number and register contents as shown below:

```
BIOSPB: db FUNC  ; BIOS function number
        db AREG  ; A register contents
        dw BCREG ; BC register contents
        dw DEREG ; DE register contents
        dw HLREG ; HL register contents
```

P_TERMCPM is equivalent to S_BIOS with a BIOS function number of 1.

Note: The register-pair BIOSPB fields (BCREG, DEREG, HLREG) are defined in low-byte, high-byte order. For example, in the BCREG field, the first byte contains the C register value and the second byte contains the B register value.

Under CP/M Plus, direct BIOS calls via the BIOS jump vector are only supported for the BIOS Console I/O and List functions. You must use S_BIOS to call any other BIOS functions. In addition, S_BIOS intercepts BIOS DRV_ALLOCVEC (Select Memory) calls and returns with register A set to zero. Refer to the CP/M Plus BIOS section for the definition of the BIOS functions and their register-passing and return conventions.

See the section entitled BIOS Overview for a list of Osborne Executive CP/M Plus BIOS functions and function numbers.

Note: XMOVE and MOVE, when used in conjunction, should be called directly and not through BDOS. The reason for this is that the BDOS calls move when function 50 is called.

S_RSX

SYSTEM CALL 60: CALL RESIDENT SYSTEM EXTENSION

Entry Parameters:
- Register C: 3CH
- Register DE: RSX PB Address

Returned Value:
- Register A: Error Code
- Register H: Physical Error

S_RSX is a special BDOS function that you use when you call Resident System Extensions. The RSX subfunction is specified in a structure called the RSX Parameter Block, defined as follows:

```
RSXPB: db FUNC      ; RSX Function number
       db NUMPARMS  ; Number of word parameters
```

```
        dw PARMETER1 ; Parameter 1
        dw PARMETER2 ; Parameter 2
        . . .
        dw PARMETERn ; Parameter n
```

RSX modules filter all BDOS calls and capture RSX function calls that they can handle. If there is no RSX module present in memory that can handle a specific RSX function call, the call is not trapped, and the BDOS returns 0FFH in registers A and L. RSX function numbers from 0 to 127 are available for CP/M Plus-compatible software use. RSX function numbers 128 to 255 are reserved for system use.

```
+-----------------------------------------------+
!                                               !
!                    S_SCB                      !
!                                               !
!      SYSTEM CALL  49:  GET/SET SYSTEM         !
!                        CONTROL BLOCK          !
!                                               !
+-----------------------------------------------+
!                                               !
!         Entry Parameters:                     !
!             Register   C:  31H                !
!             Register  DE:  SCB PB Address     !
!                                               !
!         Returned   Value:                     !
!             Register   A:  Returned Byte      !
!             Register  HL:  Returned Word      !
!                                               !
+-----------------------------------------------+
```

The System Control Block (abbreviated SCB) is a 100-byte (64H-byte) CP/M Plus data structure that resides in the BDOS system component. The SCB contains internal BDOS flags and data, CCP flags and data, and other system information such as console characteristics and the current date and time. The BDOS, BIOS, and CCP system components, as well as CP/M Plus utilities and RSXs, reference SCB fields. The S_SCB system call provides access to the SCB fields for transient programs, RSXs, and the CCP.

However, use caution when you access the SCB and use S_SCB for two reasons. First, the SCB is a data structure applicable only to CP/M Plus. Digital Research's multiuser operating system, MP/M, does not support S_SCB. Therefore, programs that access the SCB can run only on CP/M Plus. Secondly, the SCB contains critical system parameters that reflect the current state of the operating system. If a program modifies these parameters illegally, the operating system might crash. However, for application writers

who are writing system-oriented applications, access to the SCB variables might prove valuable.

For example, the CCP default drive and current user number are maintained in the System Control Block. This information is displayed in the system prompt. If a transient program changes the current disk or user number by making an explicit BDOS call, the System Control Block values are not changed. They continue to reflect the state of the system when the transient program was loaded. For compatibility with CP/M version 2, the current disk and user number are also maintained in location 0004H of Page Zero. The high-order nibble contains the user number, and the low-order nibble contains the drive.

To use S_SCB, the calling program passes the address of the SCB parameter block in register pair DE. This data structure identifies the byte or word of the SCB to be updated or returned. The SCB parameter block is defined as:

```
SCBPB: DB OFFSET ; Offset within SCB
       DB SET    ; OFFH if setting a byte
                 ; OFEH if setting a word
                 ; 001H - OFDH are reserved
                 ; 000H if a get operation
       DW VALUE  ;Byte or word value tobe set
```

The OFFSET parameter identifies the offset of the field within the SCB to be updated or accessed. The SET parameter determines whether S_SCB is to set a byte or word value in the SCB or if it is to return a byte from the SCB. The VALUE parameter is used only in set calls. In addition, only the first byte of VALUE is referenced in set byte calls.

The System Control Block is summarized in the following table. Each of these fields is documented in detail in Appendix A.

TABLE B-19. SYSTEM CONTROL BLOCK

Hex Offset	Description
00 - 04	Reserved for System Use
05	BDOS version number
06 - 09	User Flags
0A - 0F	Reserved for System Use
10 - 11	Program Error return code
12 - 19	Reserved for System Use
1A	Console Width (columns)
1B	Console Column Position
1C	Console Page Length
1D - 21	Reserved for System Use
22 - 23	CONIN: Redirection flag, bit 7 = 0 > none
24 - 25	CONOUT: Redirection flag, bit 7 = 0 > none
26 - 27	AUXIN: Redirection flag, bit 7 = 0 > none
28 - 29	AUXOUT: Redirection flag, bit 7 = 0 > none
2A - 2B	LSTOUT: Redirection flag, bit 7 = 0 > none
2C	Page Mode
2D	Reserved for System Use
2E	CTRL-H Active
2F	Rubout Active
30 - 32	Reserved for System Use
33 - 34	Console Mode
35 - 36	Reserved for System Use
37	Output Delimiter
38	List Output Flag
39 - 3B	Reserved for System Use
3C - 3D	Current DMA Address
3E	Current Disk
3F - 43	Reserved for System Use
44	Current User Number
45 - 49	Reserved for System Use
4A	BDOS MultiSector Count
4B	BDOS Error Mode
4C - 4F	Drive Search Chain (DISKS A:,E:,F:)
50	Temporary File Drive
51	Error Disk
52 - 56	Reserved for System Use
57	BDOS Flags
58 - 5C	Date Stamp
5D - 5E	Common-Memory Base Address
5F - 63	Reserved for System Use

If S_SCB is called with the OFFSET parameter of the SCB parameter block greater than 63H, the system call performs no action but returns with registers A and HL set to zero.

```
+-----------------------------------------------+
!                                               !
!                   S_SERIAL                    !
!                                               !
!    SYSTEM CALL  107:  RETURN SERIAL NUMBER    !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  6BH                 !
!            Register  DE:  Serial Number       !
!                           Field               !
!                                               !
!        Returned Value:    Serial Number       !
!                           Field Set           !
!                                               !
+-----------------------------------------------+
```

S_SERIALreturns the CP/M Plus serial number to the6-byte field addressed by register pair DE.

BDOS-Time System Calls

```
+-----------------------------------------------+
!                                               !
!                   T_GET                       !
!                                               !
!     SYSTEM CALL  105:  GET DATE AND TIME      !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  69H                 !
!            Register  DE:  DAT Address         !
!                                               !
!        Return     Value:                      !
!            Register   A:  Seconds             !
!            DAT set                            !
!                                               !
+-----------------------------------------------+
```

T_GET obtains the system internal date and time. The calling program passes in register pair DE the address of a 4-byte data structure that receives the date and time values. The format of the date and time (DAT) data structure is the same as the format described in T_SET below. T_GET also returns the seconds field of the system date and time in register A as a two-digit BCD value.

```
+-----------------------------------------------+
!                                               !
!                    T_SET                      !
!                                               !
!     SYSTEM CALL  104:  SET DATE AND TIME      !
!                                               !
+-----------------------------------------------+
!                                               !
!        Entry Parameters:                      !
!            Register   C:  68H                 !
!            Register  DE:  DAT Address         !
!                                               !
!        Returned   Value:  None                !
!                                               !
+-----------------------------------------------+
```

T_SET sets the system internal date and time. The calling program passes the address of a 4-byte structure containing the date and time specification in the register pair DE. The format of the date and time (DAT) data structure is:

```
byte 0 - 1 : Date field
byte 2     : Hour field
byte 3     : Minute field
```

The date is represented as a 16-bit integer with day 1 corresponding to January 1, 1978. The time is represented as two bytes: hours and minutes are stored as two BCD digits.

This system call also sets the seconds field of the system date and time to zero.

CP/M PLUS BIOS DOCUMENTATION

The following sections provide information of interest to those desiring a deeper understanding of CP/M Plus, especially the system programmer who wants to customize the Osborne Executive CP/M Plus BIOS for special applications. CP/M Plus already provides this facility in the form of Resident System Extensions (RSXs); however, certain applications that cannot be implemented with RSXs may require direct modification of the BIOS.

CP/M System Components

The CP/M Plus Operating System consists of the following modules: the Console Command Processor (CCP), the Basic Disk Operating System (BDOS), and the Basic Input/Output System (BIOS).

The CCP provides the basic user interface to the operating system. It supplies six built-in commands: DIR, DIRS. ERASE, RENAME, TYPE, and USER. The CCP executes in the Transient Program Area (TPA), the region of memory for application programs. It contains the Program Loader Module which loads transient programs from disk into the TPA for execution.

The BDOS is the logical nucleus and file system of CP/M Plus. It provides the standard CP/M software interface between the application program and the physical input/output routines of the BIOS.

The BIOS interfaces the BDOS to the Osborne Executive hardware. The BIOS performs all physical I/O in the system.

The BDOS and the BIOS modules cooperate to provide the CCP and other transient programs with hardware-independent access to CP/M Plus facilities. Because the BIOS is configured for different hardware environments and the BDOS remains constant, you can transfer programs that run under CP/M Plus unchanged from systems with different hardware configurations to the Osborne Executive.

Communication Among CP/M Plus Modules

The BIOS loads the CCP into the TPA at system cold and warm start. The CCP moves the Program Loader Module to the top of the TPA and uses it to load

transient programs.

The BDOS contains a set of system calls that the CCP and applications programs call to perform disk and character I/O operations.

The BIOS contains a Jump Table with a set of 33 entry points that the BDOS calls to perform hardware-dependent primitive functions, such as peripheral device I/O. For example, CONIN: is an entry point of the BIOS called by the BDOS to read the next console input character.

Similarities exist between the BDOS system calls and the BIOS functions, particularly for simple device I/O. For example, when a transient program makes a C_WRITE system call to the BDOS, the BDOS makes a console-output function call to the BIOS. In the case of disk I/O, however, this relationship is more complex. The BDOS typically makes several BIOS function calls to perform a single BDOS file I/O system call. BDOS disk I/O is in terms of 128-byte logical records, and BIOS disk I/O is in terms of physical sectors and tracks. Therefore, sector translation must also take place.

The System Control Block (SCB) is a 100-byte (64H-byte) CP/M Plus data structure that resides in the BDOS. The BDOS and the BIOS communicate through fields in the SCB. It contains BDOS flags and data, CCP flags and data, and other system information, such as console characteristics and the current date and time. You can also access some of the System Control Block fields from an application program. However, note that the SCB contains critical system parameters which reflect the current state of the operating system. If a program modifies these parameters, the operating system can crash. See the System Control Block section of this manual, and the description of the S_SCB system call in the BDOS-System System Calls section for more information on the System Control Block.

Page Zero is a region of memory that acts as an interface between transient programs and the operating system. It contains critical system parameters, including the entry to the BDOS and the entry to the BIOS Warm-Boot routine. At system startup, the BIOS initializes these two entry points in Page Zero. All linkage between transient programs and the BDOS is restricted to the indirect linkage through Page Zero.

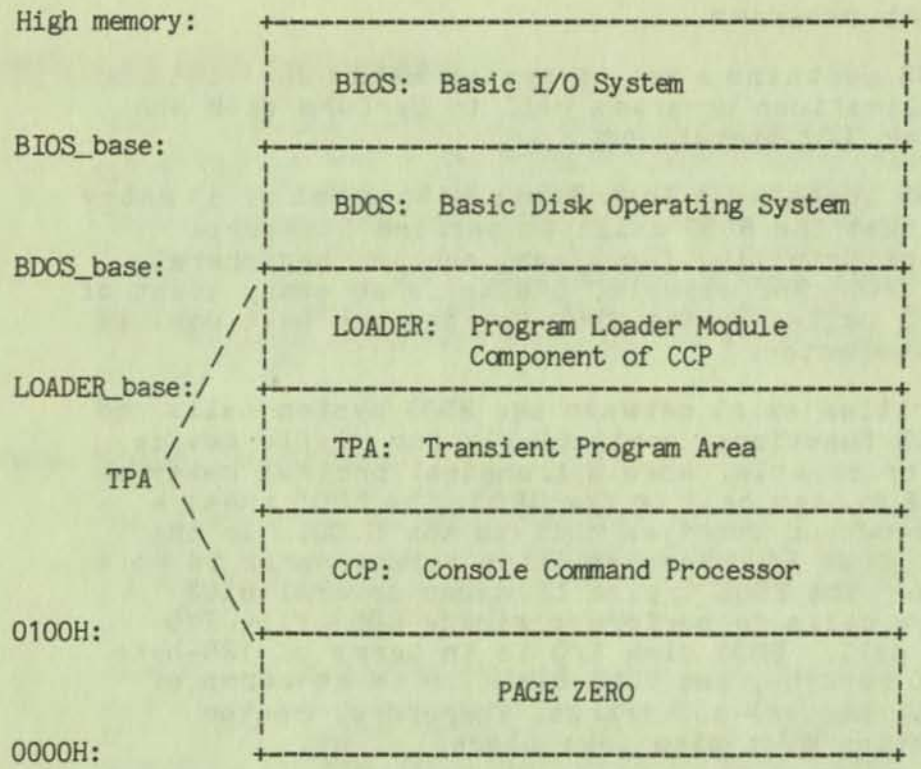


FIGURE B-7. GENERAL MEMORY ORGANIZATION OF CP/M PLUS

Note: All memory regions in CP/M Plus are page-aligned, which means that they must begin on a page boundary. Because a page is defined as 256 (100H) bytes, a page boundary always begins at a hexadecimal address that has a low-order byte of zero.

Banked and Nonbanked Systems

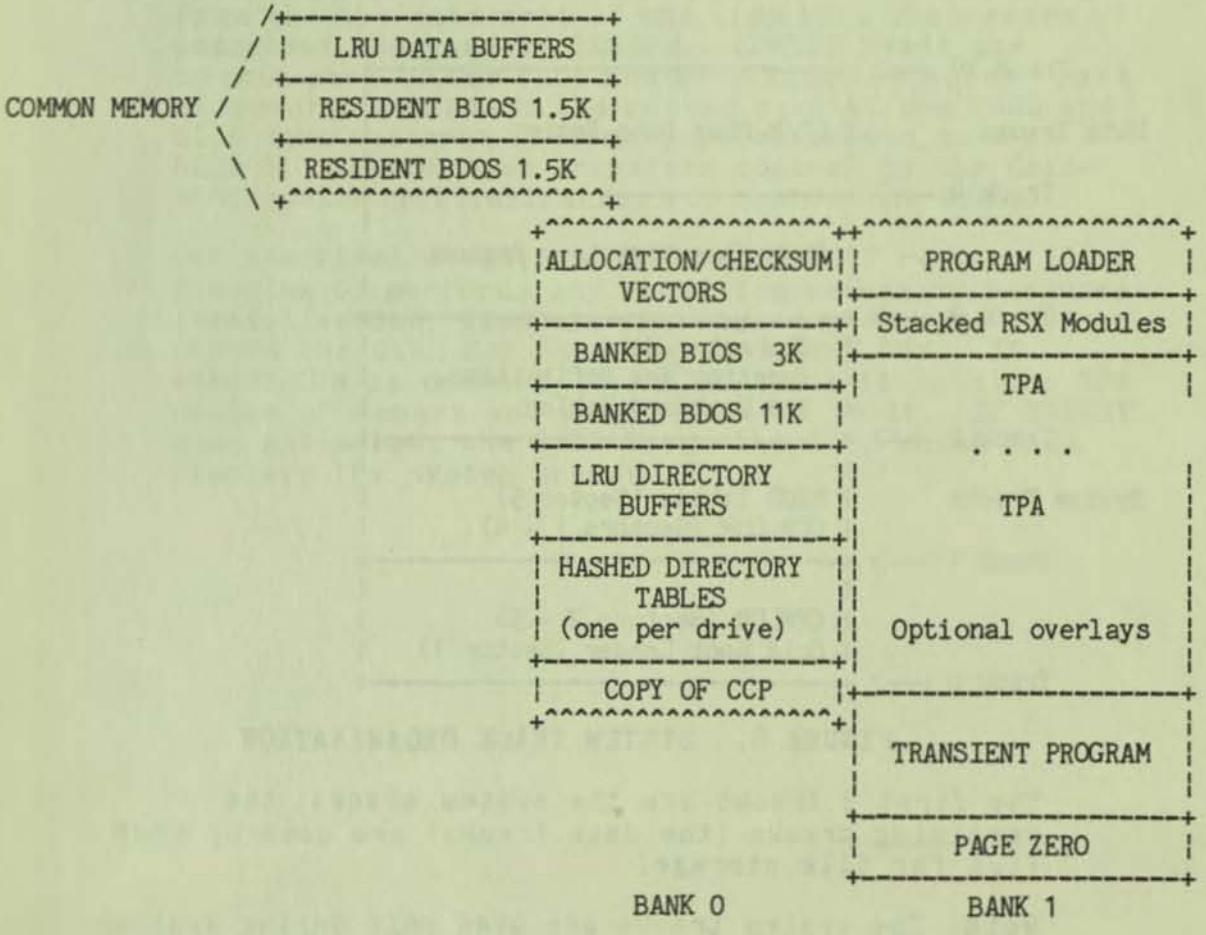
CP/M Plus supports banked memory hardware, with a minimum of 96 kilobytes of memory. For a detailed explanation of memory organization for the Osborne Executive CP/M Plus banked system, see the Introductory section of this document. Bank 0 and common memory are for the operating system. Bank 1 is the Transient Program Area, which contains the Page Zero region of memory. You can use additional banks to enhance operating system performance.

Common memory is always enabled and addressable. The operating system is divided into two modules: the resident portion, which resides in common memory; and the banked portion, which resides just below common memory in Bank 0. CPMLDR, the system loader, loads part of the BDOS into common memory and part of the BDOS into Bank 0. CPMLDR loads the BIOS in the same manner.

In the Osborne Executive the CP/M Plus banked system is large enough to contain the required buffers and the resident (common) portion of the operating system, which means a 1.5K BDOS and the common part of your customized BIOS.

CP/M Plus maintains a cache of deblocking buffers and directory records using a Least Recently Used (LRU) buffering scheme. The LRU buffer is the first to be reused when the system runs out of buffer space. The BDOS maintains separate buffer pools for directory and data-record caching.

This diagram shows the memory organization in the Osborne Executive bank-switched CP/M Plus system.



The banked system supports a TPA of 60K or more. The banked portion of the operating system in Bank 0 requires at least 16K of memory.

In the banked system the BDOS and the BIOS are separated into two parts: a resident portion and a banked portion. The resident BDOS and BIOS are

located in common memory. The banked BDOS and BIOS are located in the operating system bank, called Bank 0.

The RSX modules shown in the diagram above are Resident System Extensions (RSX) that are loaded directly below the operating system when included in an application or utility program. The Program Loader places the RSX in memory and chains BDOS calls through the RSX entry point in the RSX.

Disk Organization

The figure below illustrates the organization of an Osborne Executive CP/M Plus system disk.

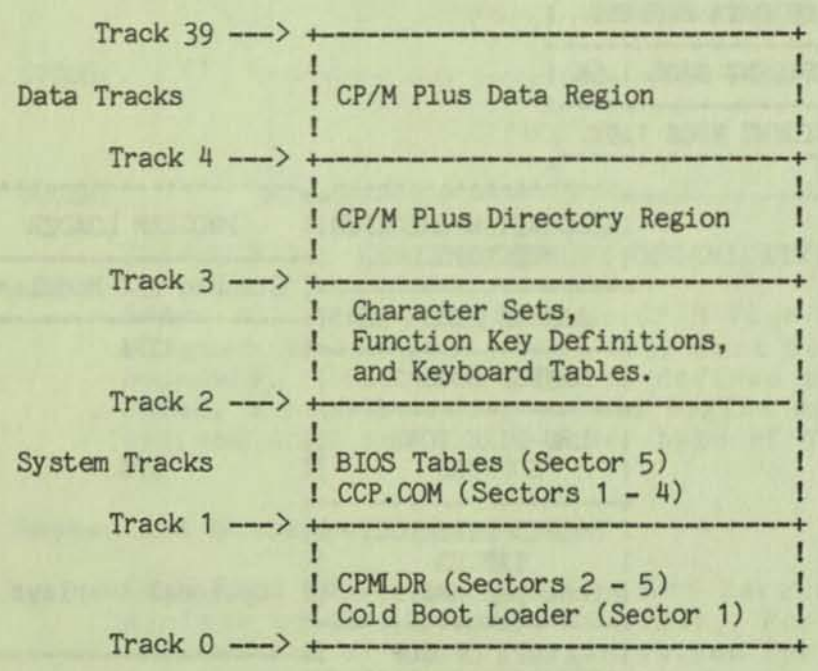


FIGURE 8. SYSTEM TRACK ORGANIZATION

The first 3 tracks are the system tracks; the remaining tracks (the data tracks) are used by CP/M Plus for file storage.

Note: The system tracks are used only during system cold start. All other CP/M Plus disk access is directed to the data tracks of the disk.

Initial Load (Cold Boot) of CP/M Plus

CP/M Plus is loaded into memory in a four-stage procedure. The first stage consists of loading into memory a small program, called the Cold Boot Loader, from the system tracks of the Boot disk. This load operation is handled by a small program in Read-Only Memory (ROM) that begins execution upon system reset.

In the second stage, the Cold Boot Loader loads the memory image of the CP/M Plus system loader program (CPMLDR) from the system tracks of a disk into memory and passes control to it. The Cold Boot Loader loads CPMLDR into Bank 0.

In the third stage, CPMLDR reads the CPM3.SYS file, which contains the BDOS and Osborne Executive BIOS, from the the data area of the disk into the memory addresses assigned by GENCPM. CPMLDR reads the common part of the BDOS and BIOS into the common part of memory, and reads the banked part of the BDOS and BIOS into the area of memory below common_base in Bank 0. CPMLDR then transfers control to the Cold-BOOT system initialization routine in the BIOS.

For the final stage, the BIOS Cold-BOOT routine (BIOS Function 0) performs any remaining necessary hardware initialization, displays the sign-on message, and checks the disk for the program EXECST.COM. If EXECST.COM is present, the BIOS loads it into the TPA region of memory and passes control to it. If EXECST does not exist, the BIOS loads the CCP, which then displays the system prompt.

THE SYSTEM CONTROL BLOCK

The System Control Block (SCB) is a data structure located in the BDOS. The SCB contains flags and data used by the CCP, the BDOS, the BIOS, and other system components. The BIOS, or any other program, can access specific data in the System Control Block through BDOS.

In the SCB.ASM file, the high-order byte of the various SCB addresses is defined as OFEH. The linker marks absolute external equates as page-relocatable when generating a System Page-Relocatable (SPR) format file. GENCPM recognizes page-relocatable addresses of OFExxH as references to the System Control Block in the BDOS. GENCPM changes these addresses to point to the actual SCB in the BDOS when it is relocating the system.

Do not perform assembly-time arithmetic on any references to the external labels of the SCB. The result of the arithmetic could alter the page value to something other than OFEH.

The example below shows the fields of the System Control Block. An "@" before a name indicates that it is a data item. A "?" preceding a name indicates that it is the label of an instruction. In the example, r/w means Read-Write, and r/o means Read-Only. The BIOS can modify a Read-Write variable, but must not modify a Read-Only variable.

THE SCB.ASM FILE

```
        title 'System Control Block Definition for CP/M3 BIOS'

        public @civec, @covec, @aivec, @aovec, @lovec, @bnkbf
        public @crdma, @crdsk, @vinfo, @resel, @fx, @usrcd
        public @mltio, @ermde, @erdsk, @media, @bflgs
        public @date, @hour, @min, @sec, ?erjmp, @mxtpa

scb$base equ    OFE00H          ; Base of the SCB

@CIVEC  equ     scb$base+22h    ; Console Input Redirection
                                ; Vector (word, r/w)
@COVEC  equ     scb$base+24h    ; Console Output Redirection
                                ; Vector (word, r/w)
@AIVEC  equ     scb$base+26h    ; Auxiliary Input Redirection
                                ; Vector (word, r/w)
@AOVECequ   scb$base+28h  ;Auxiliary Output Redirection
                                ; Vector (word, r/w)
```

```
@LOVEC  equ     scb$base+2Ah    ; List Output Redirection
                                ; Vector (word, r/w)
@BNKBF  equ     scb$base+35h    ; Address of 128-Byte Buffer
                                ; for Banked BIOS (word, r/o)
@CRDMA  equ     scb$base+3Ch    ; Current DMA Address
                                ; (word, r/o)
@CRDSK  equ     scb$base+3Eh    ; Current Disk (byte, r/o)
@VINFO  equ     scb$base+3Fh    ; BDOS Variable "INFO"
                                ; (word, r/o)
@RESEL  equ     scb$base+41h    ; FCB Flag (byte, r/o)
@FX     equ     scb$base+43h    ; system call for Error
                                ; Messages (byte, r/o)
@USRCD equ      scb$base+44h     ;Current User Code(byte, r/o)
@MLTIO equ      scb$base+4Ah    ;Current Multisector Count
                                ; (byte,r/w)
@ERMDE  equ     scb$base+4Bh    ; BDOS Error Mode (byte, r/o)
@ERDSK  equ     scb$base+51h    ; BDOS Error Disk (byte, r/o)
@MEDIA  equ     scb$base+54h    ; Set by BIOS to indicate
                                ; open door (byte,r/w)
@BFLGS  equ     scb$base+57h    ; BDOS Message Size Flag
                                ; (byte,r/o)
@DATE   equ     scb$base+58h    ; Date in Days Since 1 Jan 78
                                ; (word, r/w)
@HOUR   equ     scb$base+5Ah    ; Hour in BCD (byte, r/w)
@MIN    equ     scb$base+5Bh    ; Minute in BCD (byte, r/w)
@SEC    equ     scb$base+5Ch    ; Second in BCD (byte, r/w)
?ERJMP  equ     scb$base+5Fh    ; BDOS Error Message Jump
                                ; (three bytes, r/w)
@MXTPA  equ     scb$base+62h    ; Top of User TPA
                                ; (address at 6,7)(word, r/o)
        end
```

TABLE B-20. SCB FIELDS

Field	Meaning
@CIVEC, @COVEC, @AIVEC, @ADVEC, @LOVEC	**(Read-Write Variables)** These fields are the 16-bit I/O redirection vectors for the five logical devices: console input, console output, auxiliary input, auxiliary output, and the list device. (See the section on Character I/O Functions below.)
@BNKBF	**(Read-Only Variable)** @BNKBF contains the address of a 128-byte buffer in the resident portion of the BDOS. This buffer is available for use during BOOT and WBOOT only. The BIOS uses it to transfer a copy of the CCP from an image in an alternate bank.

TABLE B-20. SCB FIELDS (Cont.)

Field	Meaning
@CRDMA, @FX, USRCD, @ERDSK	**(Read-Only Variables)** These variables contain the current DMA address, the BDOS system call number, the current user code, and the disk code of the drive on which the last error occurred. They can be displayed when a BDOS error is intercepted by the BIOS. See ?ERJMP.
@CRDSK	**(Read-Only Variable)** @CRDSK is the current default drive, set by the DRV_SET BDOS system call.
@VINFO, @RESEL	**(Read-Only Variables)** If @RESEL is equal to OFFH, then @VINFO contains the address of a valid FCB. If @RESEL is not equal to OFFH, then @VINFO is undefined. The BIOS uses @VINFO to display the filespec when it intercepts a BDOS error.
@MLTIO	**(Read-Write Variable)** @MLTIO contains the current multisector count. The BIOS can change the multisector count directly, or through the F_MULTISEC BDOS system call. The value of the multisector count can range from 1 to 128.
@ERMDE	**(Read-Only Variable)** @ERMDE contains the current BDOS error mode. OFFH indicates the BDOS is returning error codes to the application program without displaying any error messages. OFEH indicates the BDOS is both displaying and returning errors. Any other value indicates the BDOS is displaying errors without notifying the application program.
@MEDIA	**(Read-Write Variable)** @MEDIA is a global system flag indicating that a drive door has been opened. The BIOS routine that detects the open drive door sets this flag to OFFH. The BIOS routine also sets the MEDIA byte in the Disk Parameter Header associated with the open-door drive to OFFH.

TABLE B-20. SCB FIELDS (Cont.)

Field	Meaning
@BFLGS	**(Read-Only Variable)** The BDOS in CP/M Plus produces two kinds of error messages: short error messages and extended error messages. Short error messages display one or two lines of text. Long error messages display a third line of text containing the filename, filetype, and BDOS system call number involved in the error. GENCPM sets this flag in the System Control Block to indicate whether the BIOS displays short or extended error messages. The BIOS error-message handler checks this byte in the System Control Block. If the high-order bit (bit 7) is set to zero, the BDOS displays short error messages. If the high-order bit is set to 1, the BDOS displays the extended three-line error messages. For example, the BDOS displays the following error message if the BIOS returns an error from READ and the BDOS is displaying long error messages: CP/M Error on d: Disk I/O BDOS Function = nn File = filename.typ In the above error message, Function nn and filename.typ represent the BDOS system call number and file specification involved, respectively.
@DATE	**(Read-Write Variable)** The number of days since 1 January 1978, expressed as a 16-bit unsigned integer, low byte first. A real-time clock interrupt updates the @DATE field to indicate the current date.
@HOUR, @MIN,	**(Read-Write Variable)** These two-digit Binary-Coded Decimal (BCD) fields indicate the current hour, minute, and second updated by a real-time clock interrupt.

TABLE B-20. SCB FIELDS (Cont.)

Field	Meaning
?ERJMP	**(Read-Write Code Label)** The BDOS calls the error message subroutine through this jump instruction. Register C contains an error code as follows: 1 Permanent Error 2 Read-Only Disk 3 Read-Only File 4 Select Error 7 Password Error 8 File Exists 9 ? in Filename Error code 1 above results in the BDOS message: **Disk I/O** The ?ERJMP vector allows the BIOS to intercept the BDOS error messages so you can display them in a foreign language. **Note:** This vector is not branched to if the application program is expecting return codes on physical errors. ?ERJMP is set to point to the default (English) error message routine contained in the BDOS. The BOOT routine can modify the address at ?ERJMP+1 to point to an alternate message routine. Your error-message handler can refer to @FX, @VINFO (if @RESEL is equal to 0FFH), @CRDMA, @CRDSK, and @USRCD to print additional error information. Your error handler should return to the BDOS with a RET instruction after printing the appropriate message.
@MXTPA	**(Read-Only Variable)** @MXTPA contains the address of the current BDOS entry point. This is also the address of the top of the TPA. The BOOT and WBOOT routines of the BIOS use this address to initialize the BDOS-entry JMP instruction at location 005H during system initialization. Each time an RSX is loaded, @MXTPA is adjusted by the system to reflect the change in the available User Memory (TPA).

CP/M PLUS BIOS OVERVIEW

The table below describes the entry points into the BIOS from the Cold Start Loader and the BDOS. Entry to the BIOS is through the BIOS jump vector, a set of 33 jump instructions that pass program control to the individual BIOS subroutines.

TABLE B-21. CP/M PLUS BIOS JUMP VECTOR

No.	Instruction	Description
0	JMP BOOT	Perform cold start initialization
1	JMP WBOOT	Perform warm start initialization
2	JMP CONST	Check for console input character ready
3	JMP CONIN	Read Console Character in
4	JMP CONOUT	Write Console Character out
5	JMP LIST	Write List Character out
6	JMP AUXOUT	Write Auxiliary Output Character
7	JMP AUXIN	Read Auxiliary Input Character
8	JMP HOME	Move to Track 00 on Selected Disk
9	JMP SELDSK	Select Disk Drive
10	JMP SETTRK	Set Track Number
11	JMP SETSEC	Set Sector Number
12	JMP SETDMA	Set DMA Address
13	JMP READ	Read Specified Sector
14	JMP WRITE	Write Specified Sector
15	JMP LISTST	Return List Status
16	JMP SECTRN	Translate Logical to Physical Sector
17	JMP CONOST	Return Output Status of Console
18	JMP AUXIST	Return Input Status of Aux. Port
19	JMP AUXOST	Return Output Status of Aux. Port
20	JMP DEVTBL	Return Address of Char. I/O Table
21	JMP DEVINI	Initialize Char. I/O Devices
22	JMP DRVTBL	Return Address of Disk Drive Table
23	JMP MULTIO	Set Number of Logically Consecutive sectors to be read or written
24	JMP FLUSH	Force Physical Buffer Flushing for user-supported deblocking
25	JMP MOVE	Memory to Memory Move
26	JMP TIME	Time Set/Get signal
27	JMP SELMEM	Select Bank of Memory
28	JMP SETBNK	Specify Bank for DMA Operation
29	JMP XMOVE	Set Bank When a Buffer is in a Bank other than 0 or 1
30	JMP TOROM	Call ROM routine
31	JMP RESERV1	Reserved for Future Use
32	JMP RESERV2	Reserved for Future Use

Each jump address corresponds to a particular subroutine that performs a specific hardware operation. Entry points 31 and 32 are reserved for future versions of CP/M. The five categories of system operations and the BIOS function calls that accomplish these operations are shown below.

TABLE B-22. SYSTEM CALLS

Operation	Function
System Initialization	BOOT, WBOOT, DEVTBL, DEVINI, DRVTBL
Character I/O	CONST, CONIN, CONOUT, LIST, AUXOUT, AUXIN, LISTST, CONOST, AUXIST, AUXOST
Disk I/O	HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ, WRITE, SECTRN, MULTIO, FLUSH
Memory Selects and Moves	MOVE, SELMEM, SETBNK, XMOVE
Clock Support	TIME

The table below is a summary showing the CP/M 3 BIOS function numbers, jump instruction names, and the entry and return parameters of each jump instruction in the table, arranged according to the BIOS function number.

TABLE B-23. CP/M PLUS BIOS FUNCTION JUMP TABLE SUMMARY

No.	Function	Input	Output
0	BOOT	None	None
1	WBOOT	None	None
2	CONST	None	A = 0FFH if ready A = 00H if not ready
3	CONIN	None	A = Con Char
4	CONOUT	C = Con Char	None
5	LIST	C = Char	None
6	AUXOUT	C = Char	None
7	AUXIN	None	A = Char
8	HOME	None	None
9	SELDSK	C = Drive 0-15 E = Init Sel Flag	HL = DPH addr HL = 000H if invalid dr.
10	SETTRK	BC = Track No	None
11	SETSEC	BC = Sector No	None
12	SETDMA	BC = .DMA	None
13	READ	None	A = 00H if no err A = 01H if nonrecov err A = 0FFH if media changed

TABLE B-23. CP/M PLUS BIOS FUNCTION JUMP TABLE SUMMARY (Cont.)

No.	Function	Input	Output
14	WRITE	C = Deblk Code	A = 00H if no err A = 01H if phys err A = 02H if disk is R/O A = 0FFH if media changed
15	LISTST	None	A = 00H if not ready A = 0FFH if ready
16	SECTRN	BC = Log Sect No	HL = Phys Sect No DE = Trans Tbl Adr
17	CONOST	None	A = 00H if not ready A = 0FFH if ready
18	AUXIST	None	A = 00H if not ready A = 0FFH if ready
19	AUXOST	None	A = 00H if not ready A = 0FFH if ready
20	DEVTBL	None	HL = Chr Tbl addr
21	DEVINI	C = Dev No 0 - 15	None
22	DRVTBL	None	HL = Drv Tbl addr HL = 0FFFFH HL = 0FFFEH HL = 0FFFDH
23	MULTIO	C = Multsec Cnt	None
24	FLUSH	None	A = 000H if no err A = 001H if phys err A = 002H if disk R/O
25	MOVE	HL = Dest Adr DE = Source Adr	HL & DE point to next bytes following MOVE
26	TIME	C = Get/Set Flag	None
27	SELMEM	A = Mem Bank	None
28	SETBNK	A = Mem Bank	None
29	XMOVE	B = Dest Bank C = Source Bank BC = Count	None
30	TOROM	E = Offset (from 0100H) to ROM R7N	
31	RESERV1	Reserved for Future Use	
32	RESERV2	Reserved for Future Use	

SYSTEM INITIALIZATION

When the BOOT routine of the BIOS gets control, it initializes two system parameters in Page Zero of memory, as shown below.

TABLE B-24.

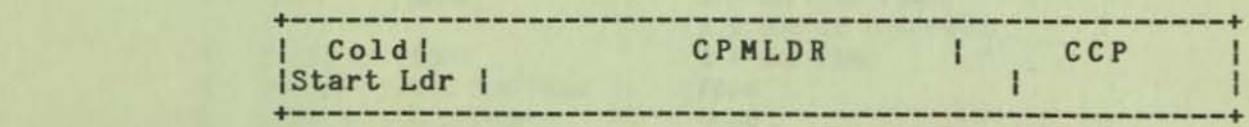
Location	Description
0,1,2	Set to JMP WBOOT (0000H: JMP BIOS+3). Location 1 and 2 must contain the address of WBOOT in the jump vector.
5,6,7	Set to JMP BDOS, the primary entry point to CP/M Plus for transient programs. The current address of the BDOS is maintained in the variable @MXTPA in the System Control Block.

The BOOT and WBOOT routine loads the CCP into the TPA in Bank 1 at location 0100H. The BIOS Cold-BOOT routine reads the CCP into memory from the system tracks.

The Cold-BOOT routine places a copy of the CCP into a reserved area of Bank 0. Then the Warm-BOOT routine copies the CCP into the TPA in Bank 1 from Bank 0 rather than reloading the CCP from disk, thus avoiding disk access during warm boot.

There is a 128-byte buffer in the resident portion of the BDOS that is used by BOOT and WBOOT. The address of this buffer is stored in the SCB variable @BNKBF. BOOT and WBOOT use this buffer as a stack area.

The system tracks for CP/M Plus are partitioned like this:



The Cold Start Loader loads CPMLDR into a constant memory location that is chosen when the system is configured. However, CPMLDR loads the BDOS and BIOS system components into memory as specified in the CPM3.SYS file generated by GENCPM, the system generation utility. Thus, CP/M Plus allows the user to configure a new system with GENCPM and then run it without having to update the system tracks of the system disk.

System Initialization Functions

This section defines the BIOS system-initialization routines BOOT, WBOOT, DEVTBL, DEVINI, and DRVTBL.

BIOS Function 0: **BOOT**
Get Control from Cold Start Loader and Initialize System Entry Parameters: None Returned Values: None

The BOOT entry point gets control from the Cold Start Loader in Bank 0 and is responsible for basic system initialization. Any remaining hardware initialization that is not done by the boot ROMs, the Cold Boot Loader, or the LDRBIOS is performed by the BOOT routine.

The BOOT routine must perform the system initialization outlined in the section on System Initialization. This includes initializing Page Zero jumps and loading the CCP. BOOT also prints the sign-on message. Control is then transferred to the CCP in the TPA at 0100H.

To initialize Page Zero, the BOOT routine places a jump at location 0000H to BIOS_base + 3, the BIOS warm-start entry point. The BOOT routine also places a jump instruction at location 0005H to the address contained in the System Control Block variable, @MXTPA.

BIOS Function 1: **WBOOT**
Get Control When a Warm Start Occurs Entry Parameters: None Returned Values: None

The WBOOT entry point is entered when a warm start occurs. A warm start is performed whenever a user program branches to location 0000H or attempts to return to the CCP. The WBOOT routine also performs the system initialization outlined in BIOS Function 0, including initializing Page Zero jumps and loading the CCP.

When WBOOT is complete, it transfers control to the CCP at location 0100H in the TPA.

Note: The CCP does not reset the disk system at warm start. The CCP only resets the disk system when a CTRL-C is pressed following the system prompt.

BIOS Function 20: **DEVTBL**
Return Address of Character I/O Table Entry Parameters: None Returned Values: HL = Addr of Chrtbl

The DEVTBL and DEVINI entry points allow you to support device assignment with a flexible, yet completely optional system. It replaces the IOBYTE facility of CP/M 2.2.

BIOS Function 21: **DEVINI**
Initialize Character I/O Device Entry Parameters: C=device number, 0-15 Returned Values: None

The DEVINI routine initializes the physical character device specified in register C to the baud rate

contained in the appropriate entry of the CHRTBL. It is referenced only by the DEVICE utility supplied with CP/M Plus.

BIOS Function 22: **DRVTBL**		
Return Address of Disk Drive Table		
Entry Parameters:	None	
Returned Values:	HL =	Addr of Drive Table of Disk Parameter Headers (DPH); Hashing can be utilizedif specified by the DPHs referenced by this DRVTBL.
	HL =	OFFFFH if no Drive Table;the BDOS is responsiblefor blocking/deblocking; Hashing is supported.
	HL =	OFFFEH if no Drive Table; the BDOS is responsible for blocking/deblocking; Hashing is not supported.

CHARACTER I/O

CP/M Plus assumes that all simple character I/O operations are performed in 8-bit ASCII, upper- and lowercase, with no parity. An ASCII CRTL-Z (1AH) denotes an end-of-file condition for an input device. The table below lists the characteristics of the logical devices:

TABLE B-25. CP/M PLUS LOGICAL DEVICE CHARACTERISTICS

Device	Characteristics
CONIN, CONOUT	The interactive console that communicates with the operator, accessed by CONST, CONIN. CONOUT, and CONOUTST. Typically, the CONSOLE is a device such as a CRT or teletype, interfaced serially, but it can also be a memory-mapped video display and keyboard. The console is an input and output device.
LIST	The system printer. LIST is usually a hard-copy device such as a daisywheel or dot-matrix printer.
AUXOUT	The auxiliary-character output device, such as a modem.
AUXIN	The auxiliary-character input device, such as a modem.

Character I/O Data Structures

The BIOS data structure CHRTBL is a character table describing the physical I/O devices. CHRTBL contains 6-byte physical device names and the characteristics of each physical device. These characteristics include a mode byte, and the current baud rate, if any, of the device. The DEVICE utility references the physical devices through the names and attributes contained in CHRTBL. DEVICE can also display the physical names and characteristics in CHRTBL.

The mode byte specifies whether the device is an input or output device, whether it has a selectable baud rate, whether it is a serial device, and if XON/XOFF protocol is enabled.

The listing below shows the character device table that the DEVICE utility uses to set and display I/O direction in the Osborne Executive Computer.

```
      ; sample character device table

chrtb db 'CRT   '             ; console VDT
      db mb$in$out+mb$serial+mb$soft$baud
      db baud$9600

      db 'LPT   '             ; system serial printer
      db mb$output+mb$serial+mb$soft$baud+mb$xon
      db baud$9600

      db 'TI810 '             ; alternate printer
      db mb$output+mb$serial+mb$soft$baud
      db baud$9600

      db 'MODEM '             ; 300-baud modem port
      db mb$in$out+mb$serial+mb$soft$baud
      db baud$300

      db 'VAX   '             ; interface to VAX 11/780
      db mb$in$out+mb$serial+mb$soft$baud
      db baud$9600

      db 'DIABLO'             ; Diablo 630 daisywheel printer
      db mb$output+mb$serial+mb$soft$baud+mb$xon$xoff
      db baud$1200

      db 'CEN   '             ; Centronics-type parallel printer
      db mb$output
      db baud$none

      db 0                    ; table terminator
```

The listing below shows the equates for the fields contained in the sample character device table.

```
; equates for mode byte fields

mb$input            equ 0000$0001b  ; device may do input
mb$output           equ 0000$0010b  ; device may do output
mb$in$out           equ mb$input+mb$output ; dev may do both
mb$soft$baud  equ 0000$0100b  ; software selectable
                              ; baud rates
mb$serial           equ 0000$1000b  ; device may use protocol
mb$xon$xoff         equ 0001$0000b  ; XON/XOFF protocol
                                    ; enabled
  ; equates for baud rate byte

baud$none           equ 0           ; no baud rate
                                    ; associated with device
baud$50             equ 1           ; 50 baud
```

```
baud$75         equ 2       ; 75 baud
baud$110        equ 3       ; 110 baud
baud$134        equ 4       ; 134.5 baud
baud$150        equ 5       ; 150 baud
baud$300        equ 6       ; 300 baud
baud$600        equ 7       ; 600 baud
baud$1200       equ 8       ; 1200 baud
baud$1800       equ 9       ; 1800 baud
baud$2400       equ 10      ; 2400 baud
baud$3600       equ 11      ; 3600 baud
baud$4800       equ 12      ; 4800 baud
baud$7200       equ 13      ; 7200 baud
baud$9600       equ 14      ; 9600 baud
baud$19200      equ 15      ; 19.2k baud
```

Character I/O Functions

This section defines the CP/M Plus character I/O routines CONST, CONIN, CONOUT, LIST, AUXOUT, AUXIN, LISTST, CONOST, AUXIST, and AUXOST.

CP/M Plus assumes all simple character I/O operations are performed in eight-bit ASCII, upper- and lowercase, with no parity. An ASCII CTRL-Z (1AH) denotes an end-of-file condition for an input device.

In CP/M Plus, you can direct each of the five logical character devices to any combination of up to twelve physical devices. Each of the five logical devices has a 16-bit vector in the System Control Block (SCB). Each bit of the vector represents a physical device where bit 15 corresponds to device zero, and bit 4 is device eleven. Bits 0 through 3 are reserved for future system use.

You can use the public names defined in the supplied SCB.ASM file to reference the I/O redirection bit vectors. The names are shown below.

TABLE B-26. I/O REDIRECTION BIT VECTORS IN SCB

Name	Logical Device
@CIVEC	Console Input
@COVEC	Console Output
@AIVEC	Auxiliary Input
@AOVEC	Auxiliary Output
@LOVEC	List Output

The BIOS sends an output character to all of the devices whose corresponding bit is set, and reads an input character from the first ready device whose corresponding bit is set.

BIOS-input status routines return true if any selected device is ready. Output status routines, however, return true only if all selected devices are ready.

```
+-----------------------------------------+
|                                         |
|          BIOS Function 2:  CONST        |
|                                         |
+-----------------------------------------+
|                                         |
|    Return Status of Console Input Device|
|                                         |
|                                         |
|    Entry Parameters: None               |
|                                         |
|    Returned Values:  A = OFFH if a console|
|                          character is ready|
|                      A = 00H if no console|
|                          character is ready|
|                                         |
+-----------------------------------------+
```

Read the status of the currently assigned console device and return OFFH in register A if a character is ready to read, and 00H in register A if no console characters are ready.

```
+-----------------------------------------+
|                                         |
|          BIOS Function 3:  CONIN        |
|                                         |
+-----------------------------------------+
|                                         |
|      Read a Character from the Console  |
|                                         |
|                                         |
|   Entry Parameters:  None               |
|                                         |
|   Returned Values:   A = Console Character|
|                                         |
+-----------------------------------------+
```

Read the next console character into register A with no parity. If no console character is ready, wait until a character is available before returning.

```
+-----------------------------------------+
|                                         |
|       BIOS Function 6:    AUXOUT        |
|                                         |
+-----------------------------------------+
|                                         |
|       Output a Character to the         |
|        Auxiliary Output Device          |
|                                         |
|                                         |
|      Entry Parameters:  C = Character   |
|                                         |
|      Returned Values:   None            |
|                                         |
+-----------------------------------------+
```

Send the character from register C to the currently assigned AUXOUT device. The character is in ASCII with no parity.

```
+-----------------------------------------+
|                                         |
|         BIOS Function 7:   AUXIN        |
|                                         |
+-----------------------------------------+
|                                         |
|         Read a Character from the       |
|          Auxiliary Input Device         |
|                                         |
|                                         |
|         Entry Parameters:  None         |
|                                         |
|         Returned Values:   A = Character|
|                                         |
+-----------------------------------------+
```

Read the next character from the currently assigned AUXIN device into register A with no parity. A returned ASCII CTRL-Z (1AH) reports an end-of-file.

```
+-----------------------------------------------+
|                                               |
|          BIOS Function 15:  LISTST            |
|                                               |
+-----------------------------------------------+
|                                               |
|            Return the Ready Status            |
|              of the List Device               |
|                                               |
|                                               |
|   Entry Parameters:  None                     |
|                                               |
|   Returned Values: A = 000H if list deviceis  |
|                        notready to accepta    |
|                        character              |
|                    A = 0FFHif listdevice is   |
|                        ready to accept a      |
|                      character                |
|                                               |
+-----------------------------------------------+
```

The BIOS LISTST function returns the ready status of the list device.

```
+-----------------------------------------------+
|                                               |
|          BIOS Function 17:  CONOST            |
|                                               |
+-----------------------------------------------+
|                                               |
|        Return Output Status of Console        |
|                                               |
|                                               |
|      Entry Parameters:  None                  |
|                                               |
|      Returned Values:   A = 0FFH if ready     |
|                         A = 00H if not ready  |
|                                               |
+-----------------------------------------------+
```

The CONOST routine checks the status of the console. CONOST returns a 0FFH if the console is ready to display another character. This entry point allows for full polled handshaking communications support.

```
+-----------------------------------------------+
|                                               |
|          BIOS Function 18:  AUXIST            |
|                                               |
+-----------------------------------------------+
|                                               |
|    Return Input Status of Auxiliary Port      |
|                                               |
|     Entry Parameters:   None                  |
|                                               |
|     Returned Values:    A = 0FFH if ready     |
|                         A = 000H if not ready |
|                                               |
+-----------------------------------------------+
```

The AUXIST routine checks the input status of the auxiliary port. This entry point allows full polled handshaking for communications support using an auxiliary port.

```
+-----------------------------------------------+
|                                               |
|          BIOS Function 19:  AUXOST            |
|                                               |
+-----------------------------------------------+
|                                               |
|    Return Output Status of Auxiliary Port     |
|                                               |
|     Entry Parameters:   None                  |
|                                               |
|     Returned Values:    A = 0FFH if ready     |
|                         A = 000H if not ready |
|                                               |
+-----------------------------------------------+
```

The AUXOST routine checks the output status of the auxiliary port. This routine allows full polled handshaking for communications support using an auxiliary port.

DISK I/O

The BDOS accomplishes disk I/O by making a sequence of calls to the various disk subroutines in the BIOS. The subroutines set up the disk number to access, the track and sector on a particular disk, and the Direct Memory Access (DMA) address and bank involved in the I/O operation. After these parameters are established, the BDOS calls the READ or WRITE BIOS function to perform the actual I/O operation. The BDOS can make a single call to SELDSK to select a disk drive, follow it with a number of read or write operations to the selected disk, and then select another drive for subsequent operations.

CP/M Plus supports multiple-sector read and write operations to optimize rotational latency on block disk transfers. The multiple-sector I/O facility is implemented in the BIOS by using the multisector count passed to the MULTIO entry point. The BDOS calls MULTIO to read or write up to 128 sectors. For every sector number 1 to n, the BDOS calls SETDMA then calls READ or WRITE.

The table below shows the sequence of BIOS calls that the BDOS makes to read or write a physical disk sector, and to read or write multiple, contiguous physical disk sectors.

TABLE B-27. SINGLE-SECTOR I/O

Call	Explanation
SELDSK	Called only when disk is initially selected or reselected.
SETTRK	Called for every read or write of a physical sector.
SETSEC	Called for every read or write of a physical sector.
SETDMA	Called for every read or write of a physical sector.
SETBNK	Called for every read or write of a physical sector.
READ, WRITE	Called for every read or write of a physical sector.

TABLE B-27. SINGLE-SECTOR I/O (Cont.)

Call	Explanation
SELDSK	Called only when disk is initially selected or reselected.
MULTIO	Called to inform the BIOS that the next n calls to disk READ or disk WRITE require a transfer of n contiguous physical sectors to contiguous memory.
SETTRK	Called for every read or write of a physical sector.
SETSEC	Called for every read or write of a physical sector.
SETDMA	Called for every read or write of a physical sector.
SETBNK	Called for every read or write of a physical sector.
READ, WRITE	Called for every read or write of a physical sector.

For example, when reading two contiguous sectors, the BIOS calls are:

Call	Explanation
SELDSK	Called to initially select disk
MULTIO	With a value of 2
SETTRK	For first sector
SETSEC	For first sector
SETDMA	For first sector
SETBNK	
READ	
SETTRK	For second sector
SETSEC	For second sector
SETDMA	For second sector
SETBNK	
READ	

The CP/M Plus BDOS performs its own blocking and deblocking of logical 128-byte records. Unlike earlier versions of CP/M, the BIOS READ and WRITE routines always transfer physical sectors as specified in the BIOS Disk Parameter Block directly to or from the DMA buffer. The BIOS Disk Parameter Header defines one or more physical sector buffers which the BDOS uses for logical record blocking and deblocking.

CP/M Plus maintains a cache of deblocking buffers and directory records using a Least Recently Used (LRU) buffering scheme. The LRU buffer is the first to be reused when the system runs out of buffer space. The BDOS maintains separate buffer pools for directory and data record caching. The BIOS contains the data structures to control the data and directory buffers and the hash tables.

CP/M Plus uses hash tables to greatly speed directory searching. The BDOS can use the hash tables to determine the location of directory entries, and therefore reduce the number of disk accesses required to read a directory entry. The hash table allows the BDOS to directly access the sector of the directory containing the desired entry without having to read the directory sequentially.

When the BIOS finds an error condition, the READ and WRITE routines perform ten retries before reporting the error condition to the BDOS. If the BIOS returns an error condition to the BDOS, the BDOS reports the error to the user in the following form:

CP/M Error on d: Disk I/O

where d: represents the drive specification of the relevant drive.

BIOS Disk Data Structures

The BIOS includes tables that describe the particular characteristics of the disk subsystem. This section describes the elements of these tables.

In general each disk drive has an associated Disk Parameter Header (DPH) that contains information about the disk drive and provides a scratchpad area for certain BDOS operations. One of the elements of this Disk Parameter Header is a pointer to the Disk Parameter Block (DPB), which contains the actual disk description.

The figure below shows the relationships between the drive table, the Disk Parameter Header, and the Data and Directory Buffer Control Block fields and their respective data structures and buffers.

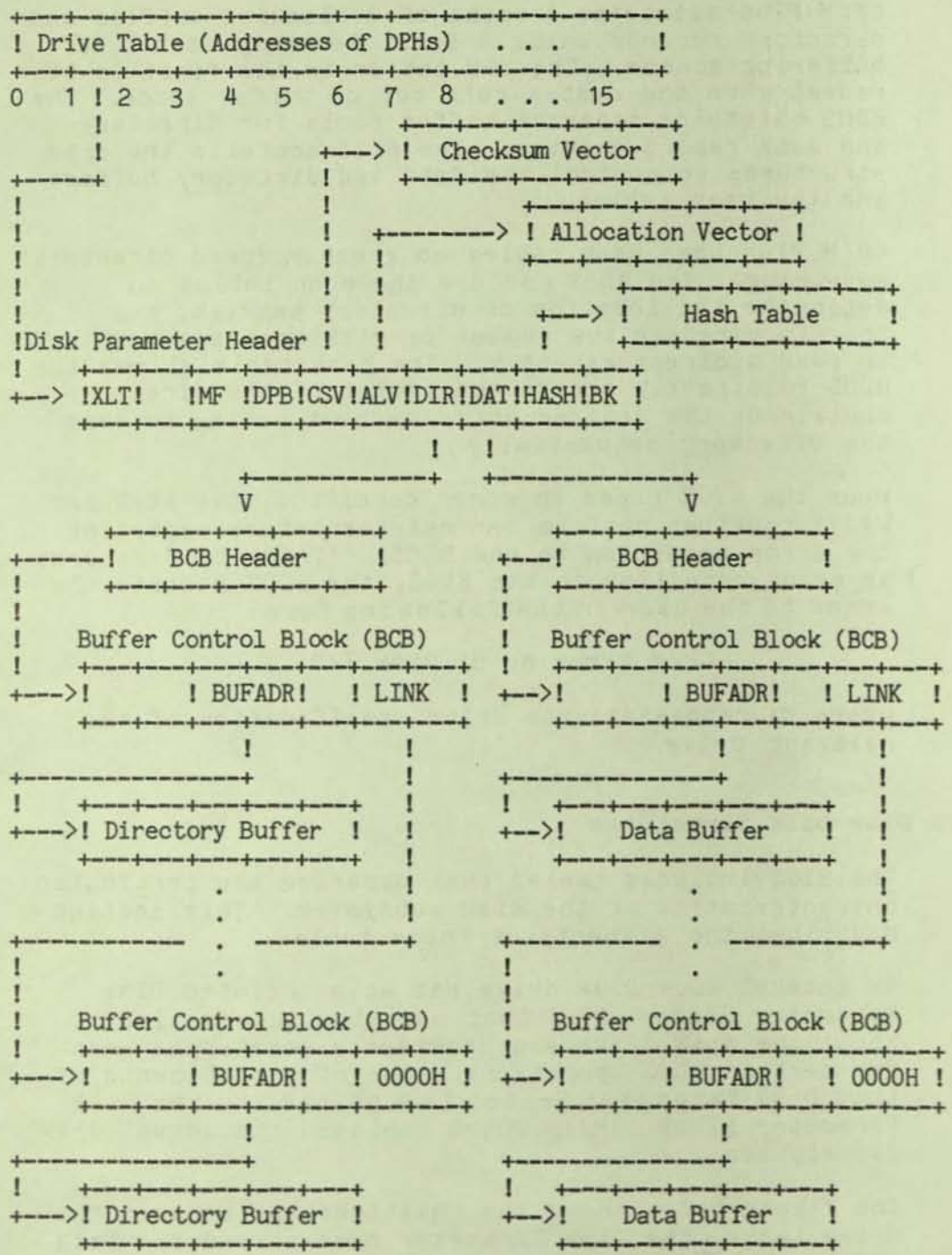


FIGURE B-9. BIOS DISK STRUCTURE

Drive Table

The drive table consists of 16 words containing the addresses of the Disk Parameter Headers for each logical drive name, A through P, and takes the general form:

```
drivetable dw    dph0
           dw    dph1
           dw    dph2
           .
           .
           .
           dw    dphF
```

For logical drives that do not exist in the Osborne Executive system, the corresponding entry in the drive table is zero.

The GENCPM utility accesses the drive table to locate the various disk-parameter data structures, so that it can determine which system configuration to use, and optionally allocate the various buffers itself. If certain addresses in the Disk Parameter Headers referenced by the drive table are set to OFFFEH, GENCPM allocates the appropriate data structures and updates the DPH.

Disk Parameter Header

In the figure below which shows the format of the Disk Parameter Header, "b" refers to bits.

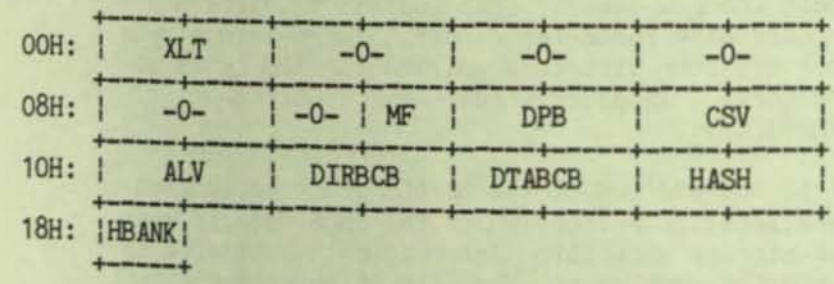


FIGURE B-10. DISK PARAMETER HEADER FORMAT

TABLE B-29. DISK PARAMETER HEADER FIELDS

Field	Comments
XLT	The XLT field contains the address of the logical-to-physical sector translation table. Disk drives with identical sector-skew factors can share the same translate table. XLT is the value passed to the SECTRN BIOS function from the BDOS in registers DE. The translation table consists of one byte per physical sector.
-0-	These 72 bits (9 bytes) of zeros are the scratch area the BDOS uses to maintain various parameters associated with the drive.
MF	MF is the Media Flag. The BDOS resets MF to zero when the drive is logged in. The BIOS sets this flag and @MEDIA in the SCB to OFFH if it detects that a drive door has been opened. If the flag is set to OFFH, the BDOS checks for a media change prior to performing the next BDOS file operation on that drive. If the BDOS determines that the drive contains a new disk, the BDOS performs a login on that drive and resets the MF flag to OOH. **Note:** The BDOS checks this flag only when a system call is made, and not during an operation.
DPB	The DPB field contains the address of a Disk Parameter Block that describes the characteristics of the disk drive.
CSV	CSV is the address of a scratchpad area used to detect changed disks. This address is different for each Disk Parameter Header. There is one byte for every four directory entries (or 128 bytes of directory). In other words, length(CSV) = (DRM/4) + 1.
ALV	ALV is the address of the scratchpad area (called the allocation vector) which the BDOS uses to keep disk-storage allocation information. This area is unique for each drive. The allocation vector requires two bits for each block on the drive. Thus, length(ALV) = (DSM/4) + 2. With double-bit allocation vectors, CP/M Plus automatically frees, at every system warm start, all file blocks that are not permanently recorded in the directory.

TABLE B-29. DISK PARAMETER HEADER FIELDS (Cont.)

Field	Comments
	Note: File space allocated to a file is not permanently recorded in a directory unless the file is closed. Therefore, the allocation vectors in memory can indicate that space is allocated although directory records indicate that space is free for allocation.
DTABCB	DTABCB contains the address of the data BCB list head in a banked system. Set DTABCB to OFFFEH for GENCPM to set up the DTABCB field. The BDOS uses data buffers to hold physical sectors so that it can block and deblock logical 128-byte records.
HASH	HASH contains the address of the directory-hashing table associated with a DPH. Set HASH to OFFFFH to disable directory hashing. Set HASH to OFFFEH to make directory hashing on the drive a GENCPM option. Each DPH using hashing must reference a unique hash table. If a hash table is supplied, it must be 4 * (DRM + 1) bytes long where DRM is one less than the length of the directory. In other words, the hash table must contain four bytes for each directory entry of the disk.
HBANK	HBANK contains the bank number of the hash table. GENCPM automatically sets HBANK when HASH is set to OFFFEH.

Extended Disk Parameter Headers (XDPHs)

An Extended Disk Parameter Header (XDPH) consists of a prefix plus a regular Disk Parameter Header as described above. The label of the XDPH references the start of the DPH. The fields of the prefix are located at relative offsets from the XDPH label.

The XDPHs for each unit of a controller are the only entry points in a particular disk drive module. They contain both the DPH for the drive and the addresses of the various action routines for that drive, including READ, WRITE, and initialization. The figure below shows the format of the Extended Disk Parameter Header.

ADDRESS	LOW BYTE / HIGH BYTE (0 7 8 15)	
XDPH-9	addr of sector INIT	
XDPH-7	addr of sector READ	
XDPH-5	addr of drive WRITE	
XDPH-3	addr of drive LOGIN	
XDPH-2	unit	
XDPH+0	addr of translate table	<--start of regular DPH
XDPH+2	0 | 0	
XDPH+4	0 | 0	
XDPH+6	0 | 0	
XDPH+8	0 | 0	
XDPH+10	Media Flag | 0	
XDPH+12	addr of DPB	
XDPH+14	addr of CSV	
XDPH+16	addr of ALV	
XDPH+18	addr of DIRBCB	
XDPH+20	addr of DTABCB	
XDPH+22	addr of HASH	
XDPH+24	hash bank	

FIGURE B-11. EXTENDED DISK-PARAMETER HEADER FORMAT

The table below describes the fields of each Extended Disk Parameter Header.

TABLE B-30. FIELDS OF EACH XDPH

Field	Meaning
WRITE	The WRITE word contains the address of the sector WRITE routine for the drive.
READ	The READ word contains the address of the sector READ routine for the drive.
LOGIN	The LOGIN word contains the address of the LOGIN routine for the drive.
INIT	The INIT word contains the address of the first-time initialization code for the drive.
UNIT	The UNIT byte contains the drive code relative to the disk controller. This is the value placed in @RDRV prior to calling the READ, WRITE, and LOGIN entry points of the drive.
Regular DPH	The remaining fields of the XDPH comprise a standard DPH.

Disk Parameter Block

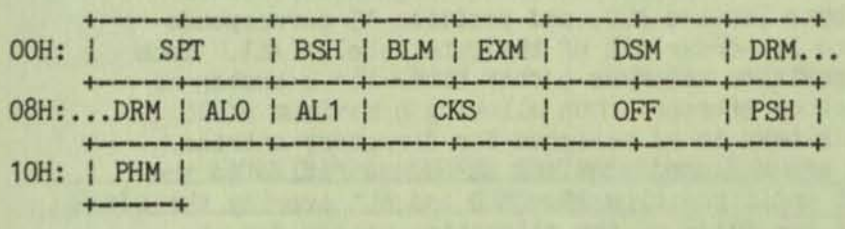
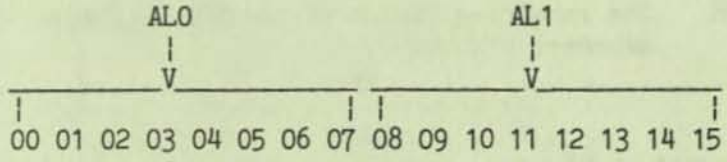


FIGURE B-12. DISK PARAMETER BLOCK FORMAT

TABLE B-32. DISK PARAMETER BLOCK FIELDS

Field	Comments
SPT	Sets SPT to the total number of 128-byte logical records per track.
BSH	Data-allocation block shift factor.
BLM	Block mask.
EXM	Extent mask.
DSM	Determines the total storage capacity of the disk drive. DSM is one less than the total number of blocks on the drive.
DRM	Total number of directory entries minus one that can be stored on this drive. The directory requires 32 bytes per entry.
ALO, AL1	Determines reserved directory blocks. The two fields ALO and AL1 can together be considered a string of 16 bits, as shown in the figure below.



Position 00 corresponds to the high-order bit of the byte labeled ALO, and position 15 corresponds to the low-order bit of the byte labeled AL1. Each bit position reserves a data block for a number of directory entries, thus allowing a maximum of 16 data blocks to be assigned for directory entries. Bits are assigned starting at 00 and filled to the right until position 15. ALO and AL1 overlay the first two bytes of the allocation vector for the associated drive.

CKS	The size of the directory check vector, (DRM/4)+1. Set bit 15 of CKS to 1 if the drive is permanently mounted. Set CKS to 8000H to indicate that the drive is permanently mounted and directory checksumming is not required. **Note:** Full directory checksumming is required on removable media to support the automatic login feature of CP/M Plus.
OFF	The number of reserved tracks at the beginning of the logical disk. OFF is the track on which the directory starts.

TABLE B-32. DISK PARAMETER BLOCK FIELDS (Cont.)

Field	Comments
PSH	Specifies the physical-record shift factor.
PHM	Specifies the physical-record mask.

CP/M allocates disk space in a unit called a block. BLS is the number of bytes in a block. The block size on the Osborne Executive is 1024 bytes.

Buffer Control Block

The Buffer Control Block (BCB) locates physical record buffers for the BDOS. The BDOS uses the BCB to manage the physical record buffers during processing. More than one Disk Parameter Header can specify the same BCB. The GENCPM utility can create the Buffer Control Block.

Note that only the DRV, BUFFAD, BANK, and LINK fields need to contain initial values. The figure below shows the form of the Buffer Control Block:

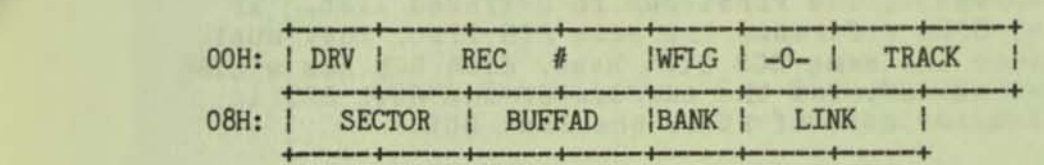


FIGURE B-13. BUFFER CONTROL BLOCK

TABLE B-32. BUFFER CONTROL BLOCK FIELDS

Field	Comment
DRV	Identifies the disk drive associated with the record contained in the buffer located at address BUFFAD.
REC#	Identifies the record position of the current contents of the buffer located at address BUFFAD. REC# consists of the absolute sector number of the record where the first record of the directory is zero.
WFLG	Set by the BDOS to OFFH to indicate that the buffer contains new data that has not yet been written to disk. When the data is written, the BDOS sets the WFLG to zero to indicate the buffer is no longer dirty.
-0-	Scratch byte used by BDOS.

TABLE B-32. BUFFER CONTROL BLOCK FIELDS (Cont.)

Field	Comment
TRACK	Contains the physical track location of the contents of the buffer.
SECTOR	Contains the physical sector location of the contents of the buffer.
BUFFAD	Specifies the address of the buffer associated with this BCB.
BANK	Contains the bank number of the buffer associated with this BCB.
LINK	Contains the address of the next BCB in a linked list, or zero if this is the last BCB in the linked list.

The BDOS distinguishes between two kinds of buffers: data buffers referenced by DTABCB, and directory buffers referenced by DIRBCB. The DIRBCB and DTABCB fields of a Disk Parameter Header each contain the address of a BCB list head rather than the address of an actual BCB. A BCB list head is a word containing the address of the first BCB in a linked list. If several DPHs reference the same BCB list, they must reference the same BCB list head. Each BCB has a LINK field that contains the address of the next BCB in the list, or zero if it is the last BCB.

The one-byte BANK field indicates the bank in which the data buffers are located. The BANK field of directory BCBs is zero because directory buffers must be located in Bank 0 below the banked BDOS module, or in common memory.

If you set the DPH DIRBCB or the DPH DTABCB fields to OFFFEH, the GENCPM utility creates BCBs; allocates physical record buffers; and sets these fields to the address of the BCBs. This allows you to write device drivers without regard to buffer requirements.

Disk I/O Functions

This section defines the CP/M 3 BIOS disk I/O routines HOME, SELDSK, SETTRK, SETSEC, SETDMA, READ, WRITE, SECTRN, MULTIO, and FLUSH.

```
+-----------------------------------------+
|                                         |
|          BIOS Function 8:  HOME         |
|                                         |
+-----------------------------------------+
|                                         |
|   Select Track 00 of the Specified Drive|
|                                         |
|                                         |
|         Entry Parameters:  None         |
|                                         |
|         Returned Values:   None         |
|                                         |
+-----------------------------------------+
```

Return the disk head of the currently selected disk to the track 00 position. The HOME call is equivalent to a call to SETTRK with a parameter of 0.

```
+-----------------------------------------+
|                                         |
|          BIOS Function 9:  SELDSK       |
|                                         |
+-----------------------------------------+
|                                         |
|       Select the Specified Disk Drive   |
|                                         |
|                                         |
|    Entry Parameters:  C = Disk Drive (0-15)
|                       E = Initial Select Flag
|                                         |
|    Returned Values:   HL = Address of DPH
|                          if drive exists|
|                       HL = 000H if drive|
|                          does not exist |
|                                         |
+-----------------------------------------+
```

Select the disk drive specified in register C for further operations, where register C contains 0 for drive A, 1 for drive B, and so on to 15 for drive P. On each disk select, SELDSK returns in HL the base address of a 25-byte area called the Disk Parameter Header. If there is an attempt to select a nonexistent drive, SELDSK returns HL = 0000H as an error indicator.

On entry to SELDSK, if it is the first time the specified disk is selected, the BDOS sets Bit 0, the (least-significant bit in Register E) to 0.

BIOS Function 10: **SETTRK**
Set Specified Track Number Entry Parameters: BC = Track Number Returned Values: None

Register BC contains the track number for a subsequent disk access on the currently selected drive. Normally the track number is saved until the next READ or WRITE occurs.

BIOS Function 11: **SETSEC**
Set Specified Sector Number Entry Parameters: BC = Sector Number Returned Values: None

Register BC contains the sector number for the subsequent disk access on the currently selected drive. This number is the value returned by SECTRN. The BIOS delays the actual sector selection until a READ or WRITE operation occurs.

Note: The current BIOS implementation only makes use of Register C.

```
+-----------------------------------------------+
|                                               |
|        BIOS Function 12:  SETDMA              |
|                                               |
+-----------------------------------------------+
|                                               |
|    Set Address for Subsequent Disk I/O        |
|                                               |
|                                               |
|    Entry Parameters:  BC = Direct Memory      |
|                            Access Address     |
|                                               |
|    Returned Values:   None                    |
|                                               |
+-----------------------------------------------+
```

Register BC contains the DMA (Direct Memory Access) address for the subsequent READ or WRITE operation. For example, if B = 00H and C = 80H when the BDOS calls SETDMA, then the subsequent read operation reads its data starting at 80H, or the subsequent write operation gets its data from 80H, until the next call to SETDMA occurs.

```
+-----------------------------------------------+
|                                               |
|        BIOS Function 13:  READ                |
|                                               |
+-----------------------------------------------+
|                                               |
|    Read a Sector from the Specified Drive     |
|                                               |
|                                               |
|    Entry Parameters:  None                    |
|                                               |
|    Returned Values:   A = 000H if no errors   |
|                       A = 001H if nonrecoverable|
|                           error occurred      |
|                       A = 0FFH if media changed|
|                                               |
+-----------------------------------------------+
```

This function assumes the BDOS has selected the drive, set the track, set the sector, and specified the DMA address. The READ subroutine attempts to read one sector based upon these parameters, then returns one of the error codes in register A as described above.

If the value in register A is 0, then CP/M Plus assumes that the disk operation completed properly. If an error occurs, the BIOS attempts ten retries to see if the error is recoverable before returning the error code.

If an error occurs in a system that supports automatic density selection, the system verifies the density of the drive. If the density has changed, READ returns a 0FFH in the accumulator. This causes the BDOS to terminate the current operation and relog-in the disk.

BIOS Function 14: **WRITE**
Write a Sector to the Specified Disk
Entry Parameters: C = Deblocking Codes
Returned Values: A = 000H if no error A = 001H if physical error A = 002H if disk is Read-Only A = 0FFH if media has changed

This BIOS Function writes the data from the currently selected DMA address to the currently selected drive, track, and sector. Upon each call to WRITE, the BDOS provides the following information in register C:

0 = deferred write
1 = nondeferred write
2 = deferred write to the first sector of a new data block

This information is provided for those BIOS implementations that do blocking/deblocking in the BIOS instead of the BDOS.

As in READ, the BIOS attempts ten retries before reporting an error.

If an error occurs in a system that supports automatic density selection, the system verifies the density of the drive. If the density has changed, the WRITE function returns a 0FFH in the accumulator. This causes the BDOS to terminate the current operation and relog-in the disk.

BIOS Function 16: **SECTRN**	
Translate Sector Number Given Translate Table	
Entry Parameters:	BC = Logical Sector Number DE = Translate Table Addr
Returned Values:	HL = Physical Sector Number

SECTRN receives a logical sector number in BC and a translate table address in DE. The logical sector number is relative to zero. The translate table address is obtained from the Disk Parameter Block for the currently selected disk. The sector number is used as an index into the translate table, with the resulting physical sector number returned in HL.

Certain drive types either do not need skewing or perform the skewing externally from the system software. In this case the skew table address in the DPH can be set to zero, and the SECTRN routine can check for the zero in DE and return with the physical sector set to the logical sector.

BIOS Function 23: **MULTIO**	
Set Multisector Count for READ or WRITE	
Entry Parameters:	C = Multisector Count
Returned Values:	None

To transfer logically consecutive disk sectors to or from contiguous memory locations, the BDOS issues a MULTIO call, followed by a series of READ or WRITE calls. This allows the BIOS to transfer multiple sectors in a single disk operation. The BIOS can transfer up to 16K bytes of data in a single operation.

Note: The current BIOS can transfer up to one full track (5K).

BIOS Function 24: **FLUSH**	
Force Physical Buffer Flushing	
Entry Parameters:	None
Returned Values:	A = 000H if no error A = 001H if physical error A = 002H if disk is R/O

The flush-buffers entry point allows the system to force physical-sector buffer flushing. The BDOS calls the FLUSH routine to ensure that no dirty buffers remain in memory. The BIOS immediately writes any buffers that contain unwritten data.

MEMORY SELECTS AND MOVES

Four BIOS functions are provided to perform memory management. The functions are MOVE, XMOVE, SELMEM, and SETBNK. The BDOS uses the BIOS MOVE routine to perform memory-to-memory block transfers. The BDOS calls XMOVE to specify the source and destination banks to be used by the MOVE routine.

The BDOS uses SELMEM when the operating system needs to execute code or access data in other than the currently selected bank.

The BDOS calls the SETBNK routine prior to calling disk READ or disk WRITE functions. The SETBNK routine saves its specified bank as the DMA bank. When the BDOS invokes a disk I/O routine, the I/O routine saves the current bank number and selects the DMA bank prior to the disk READ or WRITE. After completion of the disk READ or WRITE, the disk I/O routine reselects the current bank. When the BDOS calls the disk I/O routines, Bank 0 is in context (selected).

Memory Select and Move Functions

This section defines the memory management functions MOVE, XMOVE, SELMEM, and SETBNK.

BIOS Function 25: **MOVE**	
Memory-to-Memory Block Move	
Entry Parameters:	HL = Destination Address DE = Source Address BC = Count
Returned Values:	HL and DE must point to next bytes following move operation

The BDOS calls the MOVE routine to perform memory-to-memory block moves to allow use of the Z80 LDIR instruction. Note that the arguments in HL and DE are reversed from the Z80 machine instruction,

necessitating the use of XCHG instructions on either side of the LDIR. The BDOS uses this routine for all large-memory copy operations. On return, the HL and DE registers are expected to point to the next bytes following the move.

Usually the BDOS expects MOVE to transfer data within the currently selected bank or common memory. However, if the BDOS calls the XMOVE entry point before calling MOVE, the MOVE routine performs an interbank transfer.

BIOS Function 27: **SELMEM**
Select Memory Bank Entry Parameters: A = Memory Bank Returned Values: None

The CP/M Plus BDOS calls SELMEM to select the current memory bank for further instruction execution or buffer references.

BIOS Function 28: **SETBNK**
Specify Bank for DMA Operation Entry Parameters: A = Memory Bank Returned Values: None

SETBNK specifies the bank that the subsequent disk READ or WRITE routine must use for memory transfers. The BDOS always makes a call to SETBNK to identify the DMA bank before performing a READ or WRITE call.

Note: The BDOS does not reference banks other than 0 or 1 unless another bank is specified by the BANK

field of a Data Buffer Control Block (BCB).

BIOS Function 29: **XMOVE**
Set Banks for Following MOVE
Entry Parameters: B = Destination Bank C = Source Bank
Returned Values: None

XMOVE supports memory-to-memory DMA transfers over the entire, extended address range. An XMOVE call affects only the following MOVE call. All subsequent MOVE calls apply to the memory selected by the latest call to SELMEM. After a call to the XMOVE function, the following call to the MOVE function is not more than 128 bytes of data.

CLOCK SUPPORT

The BIOS maintains the time of day in the System Control Block and updates the time on clock interrupts. The time of day is kept as four fields in the System Control Block. @DATE is a binary word containing the number of days since January 1, 1978. The bytes @HOUR, @MIN, and @SEC contain the hour, minute, and second in Binary-Coded Decimal (BCD) format.

Clock Support Function

This section defines the clock-support function TIME.

BIOS Function 26: **TIME**
Get and Set Time
Entry Parameters: C = Time Get/Set Flag
Returned values: None

The BDOS calls the TIME function to indicate to the BIOS whether it has just set the Time and Date fields in the SCB, or whether the BDOS is about to get the Time and Date from the SCB. On entry to the TIME function, a zero in register C indicates that the BIOS should update the Time and Date fields in the SCB. A OFFH in register C indicates that the BDOS has just set the Time and Date in the SCB and the BIOS should update its clock.

This entry point allows the systems to interrogate the clock to determine the time. Since the clock is capable of generating an interrupt, an interrupt service routine is used to set the Time and Date fields on a regular basis.

GENERATING AND MOVING CP/M: GENCPM

The GENCPM utility creates a memory image of CP/M in a file called CPM3.SYS which contains the CP/M 3.0 BDOS and the BIOS tailored for the Osborne Executive. The utility program allows you to relocate system modules and allocate physical record buffers, allocation vectors, checksum vectors, and hash tables as required by BIOS. To create the CPM3.SYS file, you must have your customized BNKBIOS3.SPR file. GENCPM creates the CPM3.SYS file from three files: RESBDOS3.SPR, BNKBDOS.SPR, and BNKBIOS3.SPR. To load CPM3.SYS into memory, you must create a LDRBIOS.SPR file and incorporate it into the CPMLDR.COM FILE.

GENCPM can get its data from the file GENCPM.DAT, which you create at some other time, or you can specify the data while running GENCPM.

To use GENCPM, enter either:

GENCPM AUTO
or
GENCPM AUTO DISPLAY

If you enter GENCPM AUTO, the program will seek the file GENCPM.DAT for input data. The new system will be generated, and the only display will be the signon and signoff messages. If AUTO is specified and GENCPM.DAT does not exist, the program will revert to manual entry. This will also happen if an error occurs while running in the AUTO mode.

If you enter GENCPM AUTO DISPLAY, you may also use GENCPM.DAT for default values, entering your own where you wish the values to be different from those in GENCPM.DAT. You can specify a value by answering the question mark prompt with the appropriate value, or use the default value (which will be shown in parentheses) by simply pressing <RETURN>.

GENCPM questions and responses are shown below:

Use GENCPM.DAT for defaults (Y) ?

Y GENCPM gets its default values from the file GENCPM.DAT.

N GENCPM uses defaults built into the system.

Create a new GENCPM.DAT file (N) ?

N GENCPM will not create a new GENCPM.DAT file.

Y After GENCPM generates the new CPM3.SYS file, it creates a new GENCPM.DAT file containing the default values.

Display Load Table at Cold Boot (Y) ?

Y Upon performing a Cold Boot, the system displays the load table containing the filename, filetype, hex starting address, length of system modules, and the Transient Program Address (TPA) size.

N System displays only the TPA size on Cold Boot.

Number of console columns (#80) ?

Enter the number of columns (characters/line) for your console.

Number of lines per console page (#24) ?

Enter the number of lines per screen for your console.

Backspace echoes erased character (N) ?

N Backspace character (^H, 08h) moves back one column and erases the previous character.

Y Backspace moves forward one column and displays the previous character.

Rubout echoes erased character (Y) ?

Y Rubout (7Fh) moves forward one column and displays the previous character.

N Rubout moves back one column and erases the previous character.

Initial default drive (A:) ?

Enter drive letter the prompt is to display at Cold Boot.

Top page of memory (FF) ?

Enter the page address that is to be the top of the operating system. OFFh is the top of a 64K system.

Bank switched memory (Y) ?

Y GENCPM uses the banked system files. The Osborne Executive uses a banked system.

N GENCPM uses the nonbanked system file

Common memory base page (CO) ?

This question is displayed only if you answered Y to the previous question. Enter the page address of the start of Common Memory.

Long error messages (Y) ?

This question is displayed only if you answered Y to bank-switched memory.

Y CP/M 3.0 error messages contain the BDOS function number and the name of the file on which the operation was attempted.

N CP/M 3.0 error messages do not display the function number or file.

Double allocation vectors (Y) ?

This question is displayed only if you answered N to bank-switched memory. Double allocation vectors are described in the ALV definition of the Disk Parameter Header.

Y GENCPM creates double-bit allocation vectors for each drive.

N GENCPM creates single-bit allocation vectors for each drive.

Accept new system definition (Y) ?

Y GENCPM proceeds to the next set of questions.

N GENCPM repeats the previous questions and displays your previous input in the default parentheses. You may modify your answers.

Number of memory segments (#3) ?

GENCPM displays this question if you answered Y to bank-switched memory.

Enter the number of memory segments in the system. Do not count Common memory or memory in Bank 1, the TPA bank, as a memory segment. A maximum of 16 (0 - 15) memory segments are allowed. The memory segments define to GENCPM the memory available for buffer and hash table allocation. Note that part of Bank 0 is reserved for the operating system.

CP/M 3 Base, size, bank (B6,3A,00)

Enter memory segment table:
Base, size, bank (10,A6,00) ?
Base, size, bank (00,C0,02) ?
Base, size, bank (00,C0,03) ?

Enter the base page, length, and bank of the memory segment.

Accept new memory segment table entries (Y) ?

Y GENCPM displays the next group of questions.

N GENCPM displays the memory-segment table definition questions again.

Setting up directory hash tables:
Enable hashing for drive d: (Y) ?

GENCPM displays this question if there is a Drive Table and if the DPHs for a given drive have an OFFFEh in the hash table address field of the DPH. The question is asked for every drive d: defined in the BIOS.

Y Space is allocated for the Hash Table. The address and bank of the Hash Table is entered into the DPH.

N No space is allocated for a Hash Table for that drive.

Setting up Blocking/Deblocking buffers:

GENCPM displays the next set of questions if either or both the DTABCB field or the DIRBCB field contain OFFFEh.

Number of directory buffers for drive d: (#2) ? 2

This question appears only if you are generating a banked system. Enter the number of directory buffers to allocate for the specified drive. In a banked system directory buffers are allocated only inside Bank 0.

Number of data buffers for drive d: (#1) ? 1

This question appears only if you are generating a banked system. Enter the number of data buffers to allocate for the specified drive. In a banked system, data buffers can only be allocated outside Bank 0, and in Common. You can only allocate data buffers in alternate banks if your BIOS supports interbank moves.

Share buffer(s) with which drive (A:) ?

This question appears only if you answer zero to either of the above questions. Enter the drive letter (A - P) of the drive with which you wish this drive to share a buffer.

Allocate buffers outside bank zero (N) ?

This question appears if the BIOS XMOVE routing is implemented.

Y GENCPM allocates data buffers in Common and Bank 0.

N GENCPM allocates data buffers in Common.

Accept new buffer definitions (Y)

Y GENCPM creates the CPM3.SYS file and terminates.

N GENCPM redisplays all of the buffer definition questions.

Example of System Generation with Banked Memory

The following section contains an example of a system generation session for a banked memory system. Where no entry follows a program question, assume <RETURN> was entered to select the default value in parentheses. Entries different from the default appear after the question mark.

Contents of Example GENCPM.DAT File

```
COMBAS = C0 <cr>
LERROR = ? <cr>
NUMSEGS = 3 <cr>
MEMSEG00 = 00,80,00 <cr>
MEMSEG01 = 0d, b3,02 <cr>
MEMSEG0f = ?00,C0,10 <cr>
HASHDRVA = Y {cr>
HASHDRVD = n <cr>
NDIRRECA = 20 <cr>
NDTARECF = 10 <cr>
```

Sample Run of GENCPM

```
CP/M 3.0 System Generation
Copyright (C) 1982, Digital Research

Default entries are shown in (parens).
Default base is Hex, precede entry with # for
decimal.

Use GENCPM.DAT for defaults (Y) ?

Create a new GENCPM.DAT file (N) ?

Display Load Map at Cold Boot (Y) ?

Number of console columns (#80) ?
Number of lines in console page (#24) ?
Backspace echoes erased character (N) ?
Rubout echoes erased character (N) ?

Initial default drive (A:) ?

Top page of memory (FF) ?
Bank switched memory (Y) ?
Common memory base page (C0) ?

Long error messages (Y) ?

Accept new system definitions (Y) ?

Setting up Allocation vector for drive A:
Setting up Checksum vector for drive A:
Setting up Allocation vector for drive B:
Setting up Checksum vector for drive B:
Setting up Allocation vector for drive C:
Setting up Checksum vector for drive C:
Setting up Allocation vector for drive D:
Setting up Checksum vector for drive D:

***Bank 1 and Common are not included ***
***In the memory segment table.       ***
```

```
Number of memory segments (#3) ?

CP/M 3 Base,size,bank (8B,35,00)

Enter memory segment table:
  Base,size,bank (00,8B,00) ?
  Base,size,bank (00,B3,02) ?
  Base,size,bank (00,C0,03) ?

CP/M 3 Sys      8B00h 3500h   Bank 00
Memseg No. 00   0000h 8B00h   Bank 00
Memseg No. 01   0D00h B300h   Bank 02
Memseg No. 02   0000h C000h   Bank 03

Accept new memory segment table entries (Y) ?

Setting up directory hash tables:
  Enable hashing for drive A:  (Y) ?
  Enable hashing for drive B:  (Y) ?
  Enable hashing for drive C:  (Y) ?
  Enable hashing for drive D:  (Y) ?

Setting up Blocking/Deblocking buffers:

The physical record size is 0200h:

  Available space in 256 byte pages:
  TPA = 00F4h, Bank 0 = 008Bh, Other banks = 0166h

     Number of directory buffers for drive A:  (#32) ?

  Available space in 256 byte pages:
  TPA - 00F4h, Bank = 0049h, Other banks = 0166h

     Number of data buffers for drive A:  (#2) ?
     Allocate buffers outside bank zero (N) ?

  Available space in 256 byte pages:
  TPA = 00F0h, Bank 0 = 0049h, Other banks = 0166h

     Number of directory buffers for drive B: (#32) ?

  Available space in 256 byte pages:
  TPA = 00F0h, Bank 0 = 0007h, Other banks = 0166h

     Number of data buffers for drive B:  (#0) ?
     Share buffer(s) with which drive (A:) ?

The physical record size is 0080h:

  Available space in 256 byte pages:
  TPA = 00F0h, Bank 0 = 0001h, Other banks = 0166h

     Number of directory buffers for drive C: (#01) ?
```

```
Available space in 256 byte pages:
TPA = 00F0h, Bank 0 = 0001h, Other banks = 0166h

   Number of directory buffers for drive D: (#0) ?
   Share buffer(s) with which drive (C:) ?

Available space in 256 byte pages:
TPA 00F0h, Bank 0 =  0001h, Other banks = 0166h

Accept new buffer definitions (Y) ?

BNKB10S3 SPR  F600h  0600h
BNKB10S3 SPR  B100h  0F00h
RESBDOS3 SPR  F000h  0600h
BNLBDOS3 SPR  8700h  2A00h

***CP/M 3.0 SYSTEM GENERATION DONE ***
```

APPENDIX A: SYSTEM CONTROL BLOCK

The System Control Block (SCB) is a CP/M Plus data structure located in the BDOS. CP/M Plus uses this region primarily for communication between the BDOS and the BIOS. However, it is also available for communication between application programs, RSXs, and the BDOS. Note that programs that access the System Control Block are not version independent. They can run only on CP/M Plus.

The following list describes the fields of the SCB that are available for access by application programs and RSXs. The location of each field is described as the offset from the start address of the SCB (see the S_SCB system call). The RW/RO column indicates if the SCB field is Read-Write or Read-Only.

TABLE B-33. SCB FIELDS AND DEFINITIONS

Offset	RW/RO	Definition
00 - 04	RO	Reserved for system use.
05	RO	BDOS Version Number.
06 - 09	RW	Reserved for user use. Use these four bytes for your own flags or data.
0A - 0F	RO	Reserved for system use.
10 - 11	RW	Program-Error Return Code. This 2-byte field can be used by a program to pass an error code or value to a chained program. CP/M Plus's conditional command facility also uses this field to determine if a program executes successfully. The P_RETCODE system call is used to get/set this value.
12 - 19	RO	Reserved for system use.

TABLE B-33. SCB FIELDS AND DEFINITIONS (Cont.)

Offset	RW/RO	Definition
1A	RW	Console Width. This byte contains the number of columns (characters per line) on your console relative to zero. Most systems default this value to 79. You can set this default value by using the GENCPM or the DEVICE utility. The console width value is used by the banked version of CP/M Plus in C_READSTR, CP/M Plus's console-editing input system call. Note that typing a character into the last position of the screen, as specified by the Console Width field, must not cause the terminal to advance to the next line.
1B	RO	Console Column Position. This byte contains the current console column position.
1C	RW	Console Page Length. This byte contains the page length, lines per page, of your console. Most systems default this value to 24 lines per page. This default value may be changed by using the GENCPM or the DEVICE utility (see the CP/M Plus User's Guide).
1D - 21	RO	Reserved for system use.
22 - 2B	RW	Redirection flags for each of the five logical character devices. If your system's BIOS supports assignment of logical devices to physical devices, you can direct each of the five logical character devices to any combination of up to 12 physical devices. The 16-bit word for each device represents the following: Each bit represents a physical device where bit 15 corresponds to device 0 and bit 4 corresponds to device 11. Bits 0 through 3 are reserved for system use. You can redirect the input and output logical devices with the DEVICE command
22 - 23	RW	CONIN: Redirection Flag.
24 - 25	RW	CONOUT: Redirection Flag.
26 - 27	RW	AUXIN: Redirection Flag.
28 - 29	RW	AUXOUT: Redirection Flag.
2A - 2B	RW	LSTOUT: Redirection Flag.

TABLE B-33. SCB FIELDS AND DEFINITIONS (Cont.)

Offset	RW/RO	Definition
2C	RW	Page Mode. If this byte is set to zero, some CP/M Plus utilities and CCP built-in commands display one page of data at a time; you display the next page by pressing any key. If this byte is not set to zero, the system displays data on the screen without stopping. To stop and start the display, you can press CTRL-S and CTRL-Q, respectively.
2D	RO	Reserved for system use.
2E	RW	Determines if CTRL-H is interpreted as a rub/del character. If this byte is set to 0, then CTRL-H is a backspace character (moves back and deletes). If this byte is set to 0FFH, then CTRL-H is a rub/del character (echoes the deleted character).
2F	RW	Determines if rub/del is interpreted as CTRL-H character. If this byte is set to 0, then rub/del echoes the deleted character. If this byte is set to OFF, then rub/del is interpreted as a CTRL-H character (moves back and deletes).
30 - 32	RO	Reserved for system use.
33 - 34	RW	Console Mode. This is a 16-bit system parameter that determines the action of certain BDOS Console I/O system calls.
35 - 36	RO	Reserved for system use.
37	RW	Output delimiter character. The default-output delimiter character is $, but you can change this value by using the C_DELIMIT system call.
38	RW	List Output Flag. If this byte is set to 0, console output is not echoed to the list device. If this byte is set to 1, console output is echoed to the list device.
39 - 3B	RO	Reserved for system use.
3C - 3D	RO	Current DMA Address. This address can be set by the F_DMASET system call. The CCP initializes this value to 0080H. DRV_ALLRESET system call also sets the DMA address to 0080H.

TABLE B-33. SCB FIELDS AND DEFINITIONS (Cont.)

Offset	RW/RO	Definition
3E	RO	Current Disk. This byte contains the currently selected default disk number. This value ranges from 0 to 15 corresponding to drives A - P, respectively. The DRV_GET system call can be used to determine the current disk value.
3F - 43	RO	Reserved for system use.
44	RO	Current User Number. This byte contains the current user number. This value ranges from 0 to 15. The F_USERNUM system call can change or interrogate the currently active user number.
45 - 49	RO	Reserved for system use.
4A	RW	BDOS Multisector Count. This field is set by the F_MULTISEC system call.
4B	RW	BDOS Error Mode. This field is set by the F_ERRMODE system call. If this byte is set to OFFH, the system returns to the current program without displaying any error messages. If it is set to OFEH, the system displays error messages before returning to the current program. Otherwise the system terminates the program and displays error messages. See description of the F_ERRMODE system call for a discussion of the different error modes.
4C - 4F	RW	Drive Search Chain. The first byte contains the drive number of the first drive in the chain, the second byte contains the drive number of the second drive in the chain, and so on, for up to four bytes. If less than four drives are to be searched, the next byte is set to OFFH to signal the end of the search chain. The drive values range from 0 to 16, where 0 corresponds to the default drive, while 1 to 16 corresponds to drives A - P, respectively. The drive search chain can be displayed or set by using the SETDEF utility.
50	RW	Temporary File Drive. This byte contains the drive number of the temporary file drive. The drive number ranges from 0 to 16, where 0 corresponds to the default drive, while 1 to 16 corresponds to drives A - P, respectively.

TABLE B-33. SCB FIELDS AND DEFINITIONS (Cont.)

Offset	RW/RO	Definition
51	RO	Error drive. This byte contains the drive number of the selected drive when the last physical or extended error occurred.
52 - 56	RO	Reserved for system use.
57	RO	BDOS Flags. Bit 7 applies to banked systems only. If bit 7 is set, then the system displays expanded error messages. The second error line displays the system call number and FCB information. Bit 6 applies only to nonbanked systems. If bit 6 is set, it indicates that GENCPM has specified single allocation vectors for the system. Otherwise double allocation vectors have been defined for the system. The DRV_FREEBLOCKS system call returns temporarily allocated blocks to free space only if bit 6 is reset.
58 - 59	RW	Date in days in binary since 1 Jan 78.
5A	RW	Hour in BCD (2-digit Binary Coded Decimal).
5B	RW	Minutes in BCD.
5C	RW	Seconds in BCD.
5D - 5E	RO	Common-Memory Base Address. This value is zero for nonbanked systems and non-zero for banked systems.
5F - 63	RO	Reserved for system use.

APPENDIX B: PRL FILE GENERATION

PRL Format

A Page Relocatable Program has an origin offset of 100H bytes that is stored on-disk as a file of type PRL. The format is shown in the table below.

TABLE B-34. PRL FILE FORMAT

Address	Contents
0001-0002H	Program size
0004-0005H	Minimum buffer requirements (additional memory)
0006-00FFH	Currently unused, reserved for future allocation
0100H + Program size	equals start of bit map

The bit map is a string of bits identifying those bytes in the source code that require relocation. There is one byte in the bit map for every eight bytes of source code. The most significant bit (bit 7) of the first byte of the bit map indicates whether or not the first byte of the source code requires relocation. If the bit is on, it indicates that relocation is required. The next bit (bit 6) of the first byte corresponds to the second byte of the source code, and so forth.

Generating a PRL

The preferred technique for generating a PRL file is to use the CP/M LINK-80, which can generate a PRL file from a REL relocatable object file.

```
A>link dump[op]
```

APPENDIX C: SPR GENERATION

System Page Relocatable (SPR) files are similar in format to PRL files except that SPR files have an origin offset of 0000H (see Appendix B). SPR files are provided as part of the standard CP/M Plus System: the resident and banked portions of the banked BDOS (named RESBDOS3.SPR and BNKBDOS3.SPR) and the nonbanked BDOS (named BDOS3.SPR). The customized BIOS must also be generated in SPR format before GENCPM can create a CP/M Plus system. The BIOS.SPR file is named BNKBIOS3.SPR for banked systems and BIOS3.SPR for nonbanked systems. A detailed discussion of the generation of BIOS3.SPR or BNKBIOS3.SPR is provided in the CP/M Plus BIOS section.

The method of generating an SPR is analogous to that of generating a Page Relocatable Program (described in Appendix B) with the following exceptions:

- If LINK-80 is used, the output file of type SPR is specified with the [os] or [b] option. The [b] option is used when linking BNKBIOS3.SPR.
- The code in the SPR is ORGed at 000H rather than 100H.

APPENDIX D: ASCII AND HEXADECIMAL CONVERSIONS

This appendix contains tables of the ASCII symbols, including their binary, decimal, and hexadecimal conversions.

TABLE B-35. ASCII SYMBOLS

Symbol	Meaning	Symbol	Meaning
ACK	acknowledge	FS	file separator
BEL	bell	GS	group separator
BS	backspace	HT	horizontal tabulation
CAN	cancel	LF	line feed
CR	carriage return	NAK	negative acknowledge
DC	device control	NUL	null
DEL	delete	RS	record separator
DLE	data link escape	SI	shift in
EM	end of medium	SO	shift out
ENQ	enquiry	SOH	start of heading
EOT	end of transmission	SP	space
ESC	escape	STX	start of text
ETB	end of transmission	SUB	substitute
ETX	end of text	SYN	synchronous idle
FF	form feed	US	unit separator
		VT	vertical tabulation

TABLE B-36. CONVERSION TABLE

Binary	Decimal	Hexadecimal	ASCII
0000000	000	00	NUL
0000001	001	01	SOH (CTRL-A)
0000010	002	02	STX (CTRL-B)
0000011	003	03	ETX (CTRL-C)
0000100	004	04	EOT (CTRL-D)
0000101	005	05	ENQ (CTRL-E)
0000110	006	06	ACK (CTRL-F)
0000111	007	07	BEL (CTRL-G)
0001000	008	08	BS (CTRL-H)
0001001	009	09	HT (CTRL-I)
0001010	010	0A	LF (CTRL-J)
0001011	011	0B	VT (CTRL-K)
0001100	012	0C	FF (CTRL-L)
0001101	013	0D	CR (CTRL-M)
0001110	014	0E	SO (CTRL-N)
0001111	015	0F	SI (CTRL-O)
0010000	016	10	DLE (CTRL-P)
0010001	017	11	DC1 (CTRL-Q)
0010010	018	12	DC2 (CTRL-R)
0010011	019	13	DC3 (CTRL-S)

TABLE B-36. CONVERSION TABLE (Cont.)

Binary	Decimal	Hexadecimal	ASCII
0010100	020	14	DC4 (CTRL-T)
0010101	021	15	NAK (CTRL-U)
0010110	022	16	SYN (CTRL-V)
0010111	023	17	ETB (CTRL-W)
0011000	024	18	CAN (CTRL-X)
0011001	025	19	EM (CTRL-Y)
0011010	026	1A	SUB (CTRL-Z)
0011011	027	1B	ESC (CTRL-[)
0011100	028	1C	FS (CTRL-\)
0011101	029	1D	GS (CTRL-])
0011110	030	1E	RS (CTRL-^)
0011111	031	1F	US (CTRL-_)
0100000	032	20	(SPACE)
0100001	033	21	!
0100010	034	22	"
0100011	035	23	#
0100100	036	24	$
0100101	037	25	%
0100110	038	26	&
0100111	039	27	'
0101000	040	28	(
0101001	041	29	)
0101010	042	2A	*
0101011	043	2B	+
0101100	044	2C	,
0101101	045	2D	-
0101110	046	2E	.
0101111	047	2F	/
0110000	048	30	0
0110001	049	31	1
0110010	050	32	2
0110011	051	33	3
0110100	052	34	4
0110101	053	35	5
0110110	054	36	6
0110111	055	37	7
0111000	056	38	8
0111001	057	39	9
0111010	058	3A	:
0111011	059	3B	;
0111100	060	3C	<
0111101	061	3D	=
0111110	062	3E	>
0111111	063	3F	?
1000000	064	40	@
1000001	065	41	A
1000010	066	42	B
1000011	067	43	C
1000100	068	44	D
1000101	069	45	E
1000110	070	46	F
1000111	071	47	G

TABLE B-36. CONVERSION TABLE (Cont.)

Binary	Decimal	Hexadecimal	ASCII
1001000	072	48	H
1001001	073	49	I
1001010	074	4A	J
1001011	075	4B	K
1001100	076	4C	L
1001101	077	4D	M
1001110	078	4E	N
1001111	079	4F	O
1010000	080	50	P
1010001	081	51	Q
1010010	082	52	R
1010011	083	53	S
1010100	084	54	T
1010101	085	55	U
1010110	086	56	V
1010111	087	57	W
1011000	088	58	X
1011001	089	59	Y
1011010	090	5A	Z
1011011	091	5B	[
1011100	092	5C	\
1011101	093	5D	]
1011110	094	5E	^
1011111	095	5F	<
1100000	096	60	'
1100001	097	61	a
1100010	098	62	b
1100011	099	63	c
1100100	100	64	d
1100101	101	65	e
1100110	102	66	f
1100111	103	67	g
1101000	104	68	h
1101001	105	69	i
1101010	106	6A	j
1101011	107	6B	k
1101100	108	6C	l
1101101	109	6D	m
1101110	110	6E	n
1101111	111	6F	o
1110000	112	70	p
1110001	113	71	q
1110010	114	72	r
1110011	115	73	s
1110100	116	74	t
1110101	117	75	u
1110110	118	76	v
1110111	119	77	w
1111000	120	78	x
1111001	121	79	y
1111010	122	7A	z
1111011	123	7B	{

TABLE B-36. CONVERSION TABLE (Cont.)

Binary	Decimal	Hexadecimal	ASCII
1111100	124	7C	|
1111101	125	7D	}
1111110	126	7E	~
1111111	127	7F	DEL

APPENDIX E: THE SYSCALLS.ASM FILE

```
C_AUXIN         equ     3       ; Auxiliary Input
C_AUXINST       equ     7       ; Auxiliary Input Status
C_AUXOUT        equ     4       ; Auxiliary Output
C_AUXOUTST      equ     8       ; Auxiliary Output Status
C_DELIMIT       equ     110     ; Get/Set Output Delimiter
C_MODE          equ     109     ; Get/Set Console Mode
C_RAWIO         equ     6       ; Raw Console I/O
C_READ          equ     1       ; Read Console
C_READSTR       equ     10      ; Read Console Buffer
C_STAT          equ     11      ; Get Console Status
C_WRITE         equ     2       ; Write to Console
C_WRITEBLK      equ     111     ; Write Block
C_WRITESTR      equ     9       ; Write String
DRV_ALLOCVEC    equ     27      ; Get Addr(Alloc)
DRV_ALLRESET    equ     13      ; Reset Disk System
DRV_DPB         equ     31      ; Get Addr(DPB)
DRV_FREEBLOCKS  equ     98      ; Free Blocks
DRV_GET         equ     25      ; Return Current Disk
DRV_GETLABEL    equ     101     ; Return Directory Label Data
DRV_LOGINVEC    equ     24      ; Return Login Vector
DRV_RESET       equ     37      ; Reset Drive
DRV_ROVEC       equ     29      ; Get R/O Vector
DRV_SET         equ     14      ; Select Disk
DRV_SETLABEL    equ     100     ; Set Directory Label
DRV_SETRO       equ     28      ; Write Protect Disk
DRV_SPACE       equ     46      ; Get Disk Free Space
F_ATTRIB        equ     30      ; Set File Attributes
F_CLOSE         equ     16      ; Close File
F_DELETE        equ     19      ; Delete File
F_DMASET        equ     26      ; Set DMA Address
F_ERRMODE       equ     45      ; Set BDOS Error Mode
F_FLUSH         equ     48      ; Flush Buffers
F_LOCK          equ     42      ; Lock Record
F_MAKE          equ     22      ; Make File
F_MULTISEC      equ     44      ; Set Multisector Count
F_OPEN          equ     15      ; Open File
F_PARSE         equ     152     ; Parse Filename
F_PASSWD        equ     106     ; Set Default Password
F_RANDREC       equ     36      ; Set Random Record
F_READ          equ     20      ; Read Sequential
F_READRAND      equ     33      ; Read Random
F_RENAME        equ     23      ; Rename File
F_SFIRST        equ     17      ; Search for First
F_SIZE          equ     35      ; Compute File Size
F_SNEXT         equ     18      ; Search for Next
F_TIMEDATE      equ     102     ; Read Date Stamps, Password Mode
F_TRUNCATE      equ     99      ; Truncate File
F_TSTWRITE      equ     41      ; Test and Write Record
F_UNLOCK        equ     43      ; Unlock Record
F_USERNUM       equ     32      ; Set/Get User Code
F_WRITE         equ     21      ; Write Sequential
F_WRITERAND     equ     34      ; Write Random
```

```
F_WRITEXFCB     equ     103     ; Write File XFCB
F_WRITEZF       equ     40      ; Write Random with Zero Fill
L_WRITE         equ     5       ; Write to List
L_WRITEBLK      equ     112     ; List Block Write
P_CHAIN         equ     47      ; Chain to Program
P_LOAD          equ     59      ; Load Overlay
P_RETCODE       equ     108     ; Get/Set Program Return Code
P_TERMCPM       equ     0       ; System Reset
S_BDOSVER       equ     12      ; Return Version Number
S_BIOS          equ     50      ; Direct Bios Calls
S_RSX           equ     60      ; Call Resident System Extension
S_SCB           equ     49      ; Get/Set System Control Block
S_SERIAL        equ     107     ; Return Serial Number
T_GET           equ     105     ; Get Date and Time
T_SET           equ     104     ; Set Date and Time
```

OSBORNE™
COMPUTER CORPORATION

2F00213-01