```
CP/M   ver 1.4 & 2.x        Programmer's Reference Guide
=========================================================


BUILT-IN COMMANDS
=================

DIR                         Display file directory, current drive
DIR d:                      Display file directory, designated drive
DIR filename.typ            Search for file name, current drive
DIR *.typ                   Display all files of named type, curr drive
DIR filename.*              Display all types of designated filename
DIR x????.*                 Display all filenames 5 characters long and
                              starting with letter x

TYPE filename.typ           Display ASCII file, current drive
TYPE d:filename.typ         Display ASCII file, designated drive

ERA filename.typ            Erase named file, current drive
ERA *.*                     Erase all files, curr drv, ver 2.x curr user
ERA *.typ                   Erase all files, current drive
ERA d:filename.typ          Erase named file, designated drive
ERA filename.*              Erase all types of named file, current drive

REN nuname.typ=olname.typ   Rename file, current drive
REN d:nuname.typ=olname.typ Rename file, designated drive

SAVE n filename.typ         Save as named file, current drive
SAVE n d:filename.typ       Save as named file, designated drive
                              n pages (page = 256 bytes) starting at 100H

d:                          Switch to designated drive, making it current drive
                              V 1.4: A-D     V 2.x: A-P

USER n                      Change user area (n=0 to 15)  (ver 2.x)



TRANSIENT COMMANDS
==================

DDT                         Initiate Dynamic Debugging Tool
DDT filename.typ            Initiate DDT and load named file

ASM filename                Assemble named ASM file on current drive
ASM d:filename              Assemble named ASM file on designated drive
ASM filename.abc            Assemble named ASM file:
                              a = source file drive
                              b = HEX file destination drive (Z=skip)
                              c = PRN file destination drive (X=console,Z=skip)

LOAD filename               Make COM file from named HEX file on current drive
LOAD d:filename             Make COM file from named HEX file on design. drive

DUMP filename.typ           Display file in hex, current drive
DUMP d:filename.typ         Display file in hex, designated drive

MOVCPM                      Relocate and execute (max) KByte CP/M system
MOVCPM n                    Relocate and execute n KByte CP/M system
MOVCPM n *                  Create relocated image in RAM of n Kbyte
                              CP/M system, ready for SYSGEN or SAVE
```

```
MOVCPM * *                  Create relocated image in RAM of (max) Kbyte
                              CP/M system, ready for SYSGEN or SAVE

SYSGEN                      Initiate SYStem GENerate program

SUBMIT filename parameters  Execute SUB file using optional parameter(s)

XSUB                        Execute eXtended SUBmit program (V2.x)

ED filename.typ             Execute EDitor to create or edit named file
ED d:filename.typ           Execute EDitor to create or edit named file

STAT                        Display STATus (R/W or R/O)  \/ current drive
STAT d:                       and available disk space   /\ design. drive
STAT DEV:                   Display DEVice assignments
STAT VAL:                   Display VALid device assignments
STAT DSK:                   Display DISK characteristics (V2.x)
STAT USR:                   Display current USeR areas   (V2.x)
STAT filename.typ $S        Display size of file (V2.x)
STAT fiename.typ            Display file characteristics, current drive
STAT d:filename.typ         Display file characteristics, designated drive
STAT d:=R/O                 Change designated drive to Read-Only
STAT filename.typ $R/O      Change named file to Read-Only (V2.x)
STAT filename.typ $R/W      Change named file to Read-Write (V2.x)
STAT filename.COM $SYS      Change named file to System file (V2.x)
STAT filename.COM $DIR      Change named file to Directory file (V2.x)
STAT gd:=pd:                Change general device (CON:,LST:,PUN:,RDR:)
                              assignment of physical device (IOBYTE)


PIP
===

Commands
--------

PIP                         Initiate Peripheral Interchange Program
*d:=s:filename.typ          Copy named file from source drive to dest drive
*d:nuname.*=s:olname.typ    Copy & rename from source drive to dest drive
PIP d:=s:filename.typ       Initiate PIP and copy named file
PIP d:=s:*.*                  from source drive  \/ all files
PIP d:=s:filename.*                    to       || all named files
PIP d:=s:*.typ                destination drive  /\ all files named type
PIP LST:=filename.typ       Send named file to list device
PIP PUN:=filename.typ       Send named file to punch device
PIP CON:=filename.typ       Send named file to console device
PIP filename.typ=RDR:       Copy data from reader device to named file

*nuname.typ=aname.typ,bname.typ,cname.typ      ASCII copy & concatenate
*nuname.typ=aname.typ,bname.typ                ASCII copy & concatenate
*nuname.typ=aname.typ[X],bname.typ[X]          binary copy & concatenate

PIP LST:=aname.typ,bname.typ       Send files in sequence to list device
PIP LST:=s:aname.typ,s:bname.typ   Send files in sequence to list device

PIP allows access to any logical and physical devices defined in the
CP/M system.  Logical devices: CON: RDR: PUN: LST:
Physical devices:  TTY: CRT: PTR: UR1: UR2: PTP: UP1: UP2: LPT: UL1:

Special PIP devices (locations 109H to 1FFH are not used in the PIP
image and can be replaced by used drivers using DDT)
    NUL:    Send 40 NUL's (ASCII 00H) to the device
            (can be issued at the end of punched output)
    EOF:    Send a CP/M EOF (ASCII Ctrl-Z=1AH) to dest device
            (sent automatically at end of ASCII transfers thru PIP)
```

```
    INP:    Special PIP input source which can be patched into PIP:
            PIP gets input from here by calling 103H, with data
            returned at 109H)
    OUT:    Special PIP output destination which can be patched into PIP:
            PIP calls 106H with data to be output in C for each char.
    PRN:    Same as LST: except that tabbs are expanded to every 8th
            column, lines are numbered, and page ejects are inserted
            every 60 lines with an initial eject (same as PIP options [t8np])


Parameters
----------

example   *filename.typ=RDR:[B]

[B]         - read data block until ^S (ctrl-S) character
[Dn]        - delete characters past column n
[E]         - echo all copy operations to console
[F]         - remove form feeds
[Gn]        - get file from user area n (V2.x)
[H]         - check for proper HEX format
[I]         - same as H plus ignores ":00"
[L]         - change all upper case characters to lower case
[N]         - add line numbers without leading zeros
[N2]        - same as N plus leading zeros and a TAB after number
[O]         - object file transfer; ignore end-of-file (Ctrl-Z)
[P]         - insert form feed every 60 lines
[Pn]        - insert form feed every n lines
[Qstring^Z] - Quit copying after string is found
[R]         - read SYS file (V2.x)
[Sstring^Z] - Start copying when string is found
[Tn]        - expand tab space to every n columns
[U]         - change all lower case characters to upper case
[V]         - verify copied data (destination must be disk file)
[W]         - delete R/O files at destination (V2.x)
[X]         - copy non-ACII files
[Z]         - zero parity bit (hi bit) on all characters in file


Keywords
--------

CON: CONsole device (defined in BIOS)
EOF: send End-of-File (ASCII ^Z) to device
INP: INPut source (pathced in PIP)
LST: LiST device (defined in BIOS)
NUL: send 40 NUL's to device
OUT: OUTput destination (pathced in PIP)
PRN: same as LST:; tabs every 8th char, number lines & page
       ejects every 60 lines with initial eject
PUN: PUNch device (defined in BIOS)
RDR: ReaDeR device (defined in BIOS)


COMMAND CONTROL CHARACTERS
==========================

Control char   ASCII code   Function

  C               03h       Reboot - CP/M warm boot
  E               05h       Start new line
  H               08h       Backspace and delete (V2.x)
  I               09h       Tab 8 columns
```

J	0Ah	Line feed
M	0Dh	Carriage return
P	10h	Printer on/Printer off
R	12h	Retype current line
S	13h	Stop display outout (any char except ^C restarts)
U	15h	Delete line
X	18h	Same as Û (V1.4)
Z	1Ah	End of console input (ED & PIP)
delete/rubout	7Fh	Delete and display character (tape only)

ASM
===

Conventions

```
line#  label  operation  operant  ;comment
```

```
labels followed by colon      1-16 alphanumeric characters
symbol (eq. EQU) no colon     first must be alpha, ? or .
                              labels are case insensitive (treated as uppercase)
                              $ is insignificant and can be inserted
                              anywhere for readability
```

Assembly Program Format (space separates fields)

```
[line#]  label:   opcode  oerand(s)          ;comment
```

Constants
A number of digits with a suffix:

B	binary
O or Q	octal
D	decimal (default)
H	hexadecimal

Reserved words in operand fields

The names of the 8080 registers are reserved, and produce the following values if encountered in the operand field:

A	7
B	0
C	1
D	2
E	3
H	4
L	5
M	6
SP	6
PSW	6

Mnemonics for machine instructions are reserved and evaluate to their internal codes. Instructions which require operands will get zeroes in their operand fields, e.g. MOV will produce 40H

The symbol $ in the operand field evaulates to the address of the next instruction to generate, not including the instruction within the current logical line

String constants are delimieted by an apostrophe ('), and a double apostrophe ('') will produce one apostrophe

```
Operators (unsigned)

a+b         a added to b
a-b         difference between a and b
 +b         0+b (unary addition)
 -b         0-b (unary subtraction)
a*b         a multiplied by b
a/b         a divided by b (integer)
a MOD b     remainder after a/b
  NOT b     complement all b-bits
a AND b     bit-by-bit AND of a and b
a OR  b     bit-by-bit OR  of a and b
a XOR b     bit-by-bit XOR of a and b
a SHL b     shift a left b bits, end off, zero fill
a SHR b     shift a right b bits, end off, zero fill


Hierarcy of operations

highest:  * / MOD SHL SHR
          - +
          NOT
          AND
          OR XOR



Pseudo-ops

ORG  const          Set program or data origin (Default=0)
END  start          End program, optional address where excution begins

EQU  const          Define symbol value (may not be changed)
SET  const          Define symbol value (may be changed later)

IF   const          Assemble block conditionally until ENDIF
ENDIF               Terminate conditionala ssembly block

DS   const          Define storage sace for later use
DB byte[,byte...]   Define bytes as numeric or ASCII constants
DW word[,word...]   Define words (two bytes)

    const=constant  (true if bit 0 is 1, otherwise false)


Error codes
-----------

D   Data error (element cannot be placed in data area)
E   Expression error (ill-formed expression)
L   Label error
N   Not implemented
O   Overflow (expression too complicated to compute)
P   Phase error (label has different values on each pass)
R   register error (specified value not compatible with op code)
U   Undefined label (label does not exist)
V   Vaue error (operand improper)

Fatal errors
------------

NO SOURCE FILE PRESENT
NO DIRECTORY SPACE
SOURCE FILE NAME ERROR
```

```
SOURCE FILE READ ERROR
OUTPUT FILE WRITE ERROR
CANNOT CLOSE FILE


FILE TYPES
==========

ASC   ASCII text file, usually Basic source
ASM   ASseMbly langaige file (source for ASM program)
BAK   BAcKup copy file (created by editor)
BAS   BASic source program file, usually tokenized
COM   COMmand file (transient exeuctable program)
DAT   DATa file
DOC   DOCument file
FOR   FORtran source program file
INT   INTermediate Basic program file (executable)
HEX   HEXadecimal format file (for LOAD program)
LIB   Library file used by macro assembler
PLI   PL/I source file
PRN   PRiNt file (source and object produced by ASM)
REL   RELocatable file
SAV   System file (V2.x)
SUB   SUBmit text file executed by SUBMIT program
SYM   SID symbol file
TEX   TEXt formatter source file
XRF   Cross reference file
$$$   Temporary file

Filename - 8 characters maximum
Filetype - 3 characters maximum

Invalid filename and filetype characters
        < > . , ; : = ? [ ]



DDT COMMANDS
============

DDT
DDT filename.HEX
DDT filename.COM

A sad               Assemble symbolic code; start at sad

D                   Dump RAM to console from cad, 16 lines
D sad               Dump RAM to console from sad, 16 lines
D sad,ead           Dump RAM to console from sad thru ead

F sad,ead,const     Fill RAM from sad thru ead with const

G                   Start program exec. at saved PC
G sad               Start program exec. at sad
G sad,bp1           Start program exec. at sad and stop at bp1
G sad,bp1,bp2       Start program exec. at sad and stop at bp1 or bp2
G,bp1,bp2           Start program exec. at cad and stop at bp1 or bp2
G0                  Jump to 0000H ==> exits DDT (equivalent to Ctrl-C)

H a,b               Display hex a+b and a-b

I filename          Set up FCB at 5CH for user code
I filename.typ      Set up FCB at 5CH for R-command (HEX or COM file)

L                   Disassemble RAM from cad, 12 lines
```

```
L sad               Disassemble RAM from sad, 12 lines
L sad,ead           Disassemble RAM from sad thru ead

M sad,ead,nad       Move RAM block from sad thru ead to nad

R                   Read file specified by I command to RAM
R offset              at normal address + optional offset
                    The R command requires a previos I command
                    There is no W (write file) command, instead
                    exit DDT (by G0 or Ctrl-C) and then use SAVE

S sad               Examine and optionally alter RAM, byte by byte,
                    starting at sad

T                   Trace: execute 1 instruction  with register dump
T n                 Trace: Execute n instructions with register dump

U                   Untrace: same as T except that intermediate
U n                 steps are not displayed

X                   Examine register or flags, display format:
                      CfZfMfEfIf A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst
Xr                  Examine/change registers or flags
                      C  Carry flag         (0/1)
                      Z  Zero flag          (0/1)
                      M  Sign flag          (0/1)
                      E  Parity flag        (0/1)
                      I  Aux Carry flag     (0/1)
                      A  Accumulator        (0-FF)
                      B  BC reg pair        (0-FFFF)
                      D  DE reg pair        (0-FFFF)
                      H  HL reg pair        (0-FFFF)
                      S  Stack Pointer      (0-FFFF)
                      P  Program Counter    (0-FFFF)


    cad = current address
    nad = new address
    sad = start address
    ead = end address

    ? = error, can mean:
          file cannot be opened
          checksum error in HEX file
          assembler(disassembler overlayed



ED COMMANDS
===========

nA      Append n lines to buffer (n=0 - use haf of buffer)
B       Move pointer to beginning of file
-B      Move pointer to end of file
nC      Move pointer forward n characters
nD      Delete n characters forward
E       End edit, close file, return to CP/M
nFs     Find n'th occurence of string 's'
H       End edit, move pointer to beginning of file
I       Insert text at pointer until ^Z typed
Is      Insert string at pointer
nK      Kill n lines starting at pointer
nL      Move pointer n lines
nMx     Execute command string 'x' n times
```

```
nNs     Global F-command  - until end of file
O       Abort ED, start over with original file
nP      List next n pages of 23 lines (n=0 - current page)
Q       Quit without changing input file
Rfn     Read fn.LIB into buffer at current pointer
nSx^Zy  Substitute string 'y for next n forward occurrences of string 'x'
nT      Type n lines
U       Change lower case to upper case (next entry)
V       Enable internal line number generation
nW      Write n lines to output file, start at beginning of buffer
nX      Write next n lines to file 'X$$$$$$$$.LIB'
nZ      Pause n/2 seconds (2 MHz)
n       Move forward n lines
<CR>    Move forward one line and type one line
 -      Move backward
n:x     Move to n line number and perform 'x' command
:mx     Perform command 'x' from current line to line m
n::mx   Move to n line number and perform command 'x' from
          current line to line m

note:  "-" valid on all positioning and display commands
       for backward movement (e.g. -nC)
```

HOW TO OPEN UP A NEW USER AREA

Enter PIP wait for the * prompt. Hit Return to go back to exit PIP.
Now, enter the user area, say USER 1. Type SAVE 28 PIP.COM and hit
Return (SAVE 30 PIP.COM in CP/M 3).

Now, PIP.COM is in your new user area, and you can copy any file
into your area from area 0 by typing PIP A:=<ufn>[G0] and Return.

PATCHING THE CCP TO PERFORM ONE COMMAND AT EVERY WARM BOOT

The CCP stars with the instructions:

```
        JMP     CCPSTART        ; Start the console processor
        JMP     CCPCLEAR        ; Clear the initial command
        DB      127             ; Maximum command length
CL:     DB      0               ; Current command length
        DB      '        '      ; 8 spaces
        DB      '        '      ; 8 more spaces
        DB      'COPYRIGHT...   ; Copyright notice
```

Starting at CL, patch in the command, e.g.:

```
CL:     DB      3               ; Current command length
        DB      'DIR',0         ; DIR command, NUL terminated
        DB      '    '          ; 4 spaces
        DB      '        '      ; 8 more spaces
        DB      'COPYRIGHT...   ; Copyright notice
```

and add this to the CCP image on the system tracks of your disk
(using MOVCPM, DDT and SYSTEM on most CP/M systems, or DDT amd
CPM56K.COM or CPM60K.COM on Apple CP/M). Now, after every warm boot
the CCP will execute this command (in this example a 'DIR' command)

BDOS FUNCTION CALLS

Function no in C reg			Value passed to BDOS in DE (or E) regs	Value returned in A or HL regs
Dec	Hex			
0	00	System reset	--	--
1	01	Console read	--	A = char
2	02	Console write	E = char	--
3	03	Reader read	--	A = char
4	04	Punch write	E = char	--
5	05	List write	E = char	--
6	06	Direct console I/O (V2.x)	E = FFh (input) E = char (output)	A = char --
7	07	Get IOBYTE	--	A = IOBYTE
8	08	Set IOBYTE	E = IOBYTE	--
9	09	Print string string terminated by $, tabs are expanded as in func 2	DE = string addr	--
10	0A	Read console buffer buffer: 1st byte = bufsize, 2nd byte = chars input	DE = buffer addr	A = #chars in buffer
11	0B	Get console status	--	A = 00(not rdy)/FF(rdy)
12	0C	Lift head (V1.x)	--	--
		Get version (V2.x)	--	HL = version no H: 0=CP/M, 1=MP/M L: 0=v1.4 20H-22H=v2.x
13	0D	Reset disk**	--	--
14	0E	Select disk	E = drive no 0=A, 1=B, ...0FH=P	--
15	0F	Open file	DE = FCB addr	A = dir code
16	10	Close file	DE = FCB addr	A = dir code
17	11	Search for first	DE = FCB addr	A = dir code
18	12	Search for next	--	A = dir code
19	13	Delete file	DE = FCB addr	A = dir code
20	14	Read sequential	DE = FCB addr	A = ret code
21	15	Write sequential	DE = FCB addr	A = ret code
22	16	Create file	DE = FCB addr	A = dir code
23	17	Rename file	DE = old FCB addr	A = dir code
24	18	Get login vector	-- (V1.4)	HL = drive code
25	19	Get disk no	--	A = curr disk no (0-15 for A-P)
26	1A	Set DMA addr	DE = DMA addr	--
27	1B	Get alloc vector	--	HL = ava
28	1C	Write protect disk	--	--
29	1D	Get R/O vector	--	HL = R/O vect
30	1E	Set file attrib	DE = FCB addr	A = dir code
31	1F	Get addr disk params	--	HL = dpba
32	20	Set user code	E = user code	--
32	20	Get user code	E = FFh	A = curr user code
33	21	Read random	DE = ext. FCB addr	A = ret code ***
34	22	Write random	DE = ext. FCB addr	A = ret code ***
35	23	Compute file size	DE = ext. FCB addr	A = ret code
36	24	Set random record	DE = ext. FCB addr	A = ret code
37	25	Reset drive	DE = drive vector	A = 0
38	26	(unused)	--	--
39	27	(unused)	--	--
40	28	Write random with zero fill	DE = FCB addr	A = ret code ***

```
dir code:  directory code:
           0FFH=failed (e.g. file not found, directory full)
           0,1,2,3 = success: offset into current DMA buffer, which
           contains a directory sector, where the FCB can be found

ret code:  return code -- 0=success, non-zero=failed


  *  V1.4 none
 ** V1.4 initializes system and selects A: drive
*** ret codes:
    00 - no error
    01 - reading unwritten data
    03 - cannot close current extent
    04 - seek to unwritten extent
    05 - directory overflow (write only)
    06 - seek past physical end of disk

char = ASCII character
addr = address
dir  = directory code
cdn  = current drive number (A=0, B=1, etc)
dpba = disk parameter block address in CBIOS


Function 9:  string is terminated with '$'

Function 10:  Console buffer: 1st byte = max # chars in buffer (input)
                              2nd byte = actual # chars in buffer (output)
                              remaining bytes = buffer

Function 12: CP/M version number: H=00 CP/M, H=01 MP/M
                                  L=00 ver prior to 2.0
                                  L=20,21,22... subsequent versions

Function 13: Resets DMS address to BOOT+0080h

Function 23: renames file in first 16 bytes of FCB to name in second
             16 bytes in FCB

Function 24: Returns a 16-bit value in HL - a 16-bit bit map where
             the lowest bit represents A: and the highest bit P:
             If the bit is set, that drive is present in the CP/M system

Function 29: Returns a similar bit map as func 24, except that a set
             bit marks a drive which is Read/Only.

Function 33,34: the rn (Random Record No) must be set in the FCB prior to call

Function 35: fills in the file size in rn.  If followed by a random write,
             the file will be extended in length.  Not that the "file size"
             merely is the last record # - "hole" in sparse files are not
             accounted for

Function 36: same as function 35 except that the current random record
             position is stored in rn in FCB.

Function 37: this function is buggy - avoid using it




IOBYTE (0003H)
==============
```

Device	LST:	PUN:	RDR:	CON:
Bit position	7 6	5 4	3 2	1 0
Dec Binary				
0 00	TTY:	TTY:	TTY:	TTY:
1 01	CRT:	PTP:	PTR:	CRT:
2 02	LPT:	UP1:	UR1:	BAT:
3 03	UL1:	UP2:	UR2:	UC1:

TTY: TeleTYpe
CRT: Cathode Ray Tube type terminal
BAT: BATch process (RDR=inut, LST=output)
UC1: User defined Console
LPT: Line Printer
UL1: User defined List device
PTR: Paper Tape Reader
UR1: User defined Reader device 1
UR2: User defined Reader device 2
PTP: Paper Tape Punch
UP1: User defined Punch device 1
UP2: User defined Punch device 2

LOGIN BYTE (0004H)

low nibble = current drive (0=A, 1=B, etc)
high nibble = current user (V2.x only)

BIOS ENTRY POINTS

Hex addr	Vector name	Function	Value passed	Value returned
4A00H+b	BOOT	Cold start entry point	-	C=0
4A03H+b	WBOOT	Warm start entry point	-	C=drv no
4A06H+b	CONST	Check for console ready	-	A=const
4A09H+b	CONIN	Read from console	-	A=char
4A0CH+b	CONOUT	Write to console	C=char	-
4A0FH+b	LIST	Write to list device	C=char	-
4A12H+b	PUNCH	Write to punch device	C=char	-
4A15H+b	READER	Read from reader device	-	A=char
4A18H+b	HOME	Move head to track 0	-	-
4A1BH+b	SELDSK	Select drive	C=drv no	HL=dph*, HL=0 for error
4A1EH+b	SETTRK	Set track number	BC=trk no	-
4A21H+b	SETSEC	Set sector number	BC=sec no	-
4A24H+b	SETDMA	Set DMA address	BC=DMA	-
4A27H+b	READ	Read selected sector	-	A=dskst
4A2AH+b	WRITE	Write selected sector	-	A=dskst
4A2DH+b*	LOSTST	Get list status	-	A=lstst
4A30H+b*	SECTRAN	Sector translate	BC=lsecno DE=smap	HL=physec

BOOT: gets control after the cold start loader
Basic system initalization
Send sign-on message
Set IOBYTE
Set the WBOOT parameters
Jump to CCP at its entry point (at its first address 3400H+b)

```
WBOOT: gets control after Ctrl-C or JP 0000 or CPU reset
        Reload CP/M CCP and BDOS
        Setup JMP WBOOT at 0000H-0002H (JMP 4A03H+b)
        Set inital value of IOBYTE at 0003H
        Set 0004H hi nibble = current user no, lo nibble = current drive no
        Setup JMP BDOS at 0005H-0007H (JMP 3C06H+b)
        Set C=current drive, then branch to CCP at 3400H+b


const  = console status:  00=idle, FF=data avail

dph = disk parameter/header address

dskst = disk status:  00=OK, 01=error

lstst = list status:  00=busy, FF=ready

lsecno = logical sector number       \
physec = physical sector number      | (standard skew factor = 6)
smap = sector interlace map address  /

char = 7-bit ASCII char with parity bit (=hi bit) zero

drv no = drive number: 0=A, 1=B, etc, max 15=P
trk no = track number (0-76 std CP/M floppy, 0-65535 non-standard)
sec no = sector number (1-25 std CP/M floppy, 1-65535 non-standard)
DMA = DMA address (default 0080H)

*  = not used in V1.4
** = contents of location 0002Hz



FILE CONTROL BLOCK (FCB)
========================

Byte      Function
offset

 0        dr     Drive code (0=current, 1=A, 2=B, ...., 16=P)
 1-8      f1-f8  File name, hi but = 0
 9-11     t1-t3  File type + status (hi bits)
                   t1: 1=R/O  t2: 1=SYS  t3:  1=archived
 12       ex     Current extent number
 13       s1     reserved   (V1.4: not used)
 14       s2     =0 on BDOS call to Open/Make/Search  (v1.4: always 0)
 16       rc     extent record count: 0-127
 16-31    d0-dn  Disk map
 32       cr     Current record for R/W
 33-35    rn     Random record number, 0-65535, overflow into 3rd byte


MEMORY ALLOCATION
=================

V1.4:  b = memsize-16K
0000   - 00FF     System scratch area
0100   - 28FF+b   TPA (Transient Program Area) - COM file area
2900+b - 30FF+b   CCP - Console COmmand Processor
3100+b - 3DFF+b   BDOS
3E00+b - 3FFF+b   CBIOS

V2.2:  b = memsize-20K
0000   - 00FF     System scratch area
```

```
0100   - 33FF+b   TPA (Transient Program Area) - COM file area
3400+b - 3BFF+b   CCP - Console COmmand Processor
3C00+b - 49FF+b   BDOS
4A00+b - 4FFF+b   CBIOS


System scratch area, "page zero":

00 - 02     Jump to BIOS warm start entry point
03          IOBYTE
04          Login byte: Login drive number, current user number
05 - 07     Jump to BDOS
08 - 37     Reserved; interrupt vectors & future use
38 - 3A     RST7 - used by DDT and SID programs, contains JMP into DDT/SID
3B - 3F     Reserved for interrupt vector
40 - 4F     Scratch area for CBIOS; unused by distribution version of CP/M
50 - 5B     Not used, reserved
5C - 7C     Default FCB (File Control Block) area
7D - 7F     Optional Default Random Record Position (V2.x)
80 - FF     Default DMA buffer area (128 bytes) for disk I/O
            Also filled with CCP commandline at the start of a program



CP/M STANDARD DISK FORMAT (8" SSSD)
===================================

Media:  8" soft-sectored floppy-disk single density (IBM 3740 standard)
Tracks: 77, numbered 0 thru 76
Sectors/track: 26 (numbered 1 thru 26)
Bytes/sector: 128 data bytes (one logical record)
Storage/disk: 256256 bytes (77*26*128)
File size: any number of sectors from zero to capacity of disk
Extent: 1 kBytes - 8 sectors (smallest file space allocated)
Skew:  6 sectors standard (space between consecutive physical sectors
         on track):
       1-7-13-19-25-5-11-17-23-3-9-15-21-2-8-14-20-26-6-12-18-24-4-10-16-22

System: Track 0 & 1 (optional)
        Track 0 sector 1: boot loader
        Track 0 sectors 2-26:  CCP & BDOS
        Track 1 sectors 1-17:  CCP & BDOS
        Track 1 sectors 18-26: CBIOS

Directory: Track 2:
           16 sectors typical
           32 bytes/entry
           64 entries typical
           extents 0 and 1

User file area: Remaining sectors on Track 2 and 3 to 76, extents 2
and above

A Standard CP/M 8" SSSD floppy contains:

Track#  Sector# Page#   Mem address     CP/M module name

  00      01            (boot addr)     Cold start loader

  00      02     00      3400H+b            CCP
  00      03      .      3480H+b            CCP
  00      04     01      3500H+b            CCP
  00      05      .      3580H+b            CCP
  00      06     02      3600H+b            CCP
  00      07      .      3680H+b            CCP
```

00	08	03	3700H+b	CCP
00	09	.	3780H+b	CCP
00	10	04	3800H+b	CCP
00	11	.	3880H+b	CCP
00	12	05	3900H+b	CCP
00	13	.	3980H+b	CCP
00	14	06	3A00H+b	CCP
00	15	.	3A80H+b	CCP
00	16	07	3B00H+b	CCP
00	17	.	3B80H+b	CCP
00	18	08	3C00H+b	BDOS
00	19	.	3C80H+b	BDOS
00	20	09	3D00H+b	BDOS
00	21	.	3D80H+b	BDOS
00	22	10	3E00H+b	BDOS
00	23	.	3E80H+b	BDOS
00	24	11	3F00H+b	BDOS
00	25	.	3F80H+b	BDOS
00	26	12	4000H+b	BDOS
01	01	.	4080H+b	BDOS
01	02	13	4100H+b	BDOS
01	03	.	4180H+b	BDOS
01	04	14	4200H+b	BDOS
01	05	.	4280H+b	BDOS
01	06	15	4300H+b	BDOS
01	07	.	4380H+b	BDOS
01	08	16	4400H+b	BDOS
01	09	.	4480H+b	BDOS
01	10	17	4500H+b	BDOS
01	11	.	4580H+b	BDOS
01	12	18	4600H+b	BDOS
01	13	.	4680H+b	BDOS
01	14	19	4700H+b	BDOS
01	15	.	4780H+b	BDOS
01	16	20	4800H+b	BDOS
01	17	.	4880H+b	BDOS
01	18	21	4900H+b	BDOS
01	19	.	4980H+b	BDOS
01	20	22	4A00H+b	BIOS
01	21	.	4A80H+b	BIOS
01	22	23	4B00H+b	BIOS
01	23	.	4B80H+b	BIOS
01	24	24	4C00H+b	BIOS
01	25	.	4C80H+b	BIOS
01	26	25	4D00H+b	BIOS

02	01-08	Directory block 1
02	09-16	Directory block 2
02	17-26	Data
03-76	01-26	Data

DISK PARAMETER TABLES

Each disk drive has an associated 16-byte (8-word) DPH - Disk Parameter Header, containing:

Offset	Contents	
00H	XLT	Addr of logical-to-physical sector translation vector

```
                        or 0000H of no translation (i.e. they are the same)
                        Disk drives with identical sector skew factors
                        share the same table
 02H          0000H     \
 04H          0000H     |  Scratchpad values for use within BDOS
 06H          0000H     /   (initial value unimportant)
 08H          DIRBUF    Addr of scratchpad 128-byte directory buffer.
                        All DPH's share the same DIRBUF.
 0AH          DPB       Addr of Disk Parameter Block for this drive
 0CH          CSV       Addr of scratchpad area used for software check for
                        changed disks. Each DPH has its own CSV.
 0EH          ALV       Addr of scratchpad area used for disk storage
                        allocation information. Each DPH has its own ALV.
```

If the system has n disk drives, the n DPH's are arranged one after another, from drive 0 to drive n-1, starting at DPBASE:

DPBASE:

```
    +--------+------+------+------+--------+--------+--------+--------+
 00 | XLT 00 | 0000 | 0000 | 0000 | DIRBUF | DPB 00 | CSV 00 | ALV 00 |
    +--------+------+------+------+--------+--------+--------+--------+
 01 | XLT 01 | 0000 | 0000 | 0000 | DIRBUF | DPB 01 | CSV 01 | ALV 01 |
    +--------+------+------+------+--------+--------+--------+--------+
 ..............................................................................
    +--------+------+------+------+--------+--------+--------+--------+
n-1 | XLTn-1 | 0000 | 0000 | 0000 | DIRBUF | DPBn-1 | CSVn-1 | ALVn-1 |
    +--------+------+------+------+--------+--------+--------+--------+
```

The SELDSK subroutine is responsible for returning the base address of the DPH for the selected drive, or 0000H if there is no such drive:

```
NDISKS      EQU     4           ; Number of disk drives
.........
SELDSK:     ; Select disk given by BC
            LXI     H,0000H     ; Error return
            MOV     A,C         ; Drive OK?
            CPI     NDISK       ; Carry if so
            RNC                 ; Return if error
            ; No error, continue
            MOV     L,C         ; Low (disk)
            MOV     H,B         ; Hi (disk)
            DAD     H           ; *2
            DAD     H           ; *4
            DAD     H           ; *8
            DAD     H           ; *16
            LXI     D,DPBASE    ; First DPH
            DAD     D           ; DPH(disk)
            RET
```

The translation vectors (XLT 00 thru XLTn-1) are located elswehere in the BIOS and simply correspond one-for-one with the logical sector number zero through the sector count.

The Disk Parameter Block (DPB) for each drive type contains:

```
Offset      Contents
------      --------
  00H         SPT 16b       Total number of sectors per track
  02H         BSH  8b       Data allocation block shift factor, determined
                            by the data block allocation size
  03H         BLM  8b       Data allocation block mask (2[BSH-1])
  04H         EXM  8b       Extent mask, determined by data block allocation
                            size and number of disk blocks
  05H         DSM 16b       Total storage capacity of disk drive
```

```
  07H         DRM 16b       Total number of directory entries minus one
  09H         AL0  8b       Determines reserved directory blocks
  0AH         AL1  8b       Determines reserved directory blocks
  0BH         CKS 16b       Size of directory check vector
  0DH         OFF 16b       No of reserved tracks at beginning of logical disk
  0FH       (end of table)
```

BSH and BLM are determined by BLS, the block size or data allocation size

BLS	BSH	BLM	EXM	
			DSM<256	DSM>=256
1024	3	7	0	n/a
2048	4	15	1	0
4096	5	31	3	1
8192	6	63	7	3
16384	7	127	15	7

i.e. BLS = 2**n where n = 10 to 14
BSH = n-7
BLM = 2**BSH - 1
EXM = 2**(BHS-2) - 1 if DSM<256
EXM = 2**(BHS-3) - 1 if DSM>=256

DSM = maximum data block number supported by this particular drive, measured in BLS (BLock Size) units, or simply "number of allocation blocks on drive". Blocks are counted from 0 to DSM, and thus BLS*(DSM+1) = the number of bytes on the drive (excluding the system tracks). If DSM<256, the disk map in the directory entry of the file will be 1 byte/block. If DSM>=256 it will be 2 bytes/block.

DRM = total number of directory entries minus one.

AL0/AL1 = the directory allocation vector. Consider it a bit map of bits 16 bits, bit 0-15, where 0=hi bit of AL0, 7=lo bit of AL0, 8=hi bit of AL1, 15=lo bit of AL1. Bits are assigned starting at bit 0 up until bit 15. Suppose nbits is the number of bits seet to 1:

BLS	Directory entries
1024	32 * nbits
2048	64 * nbits
4096	128 * nbits
8192	256 * nbits
16384	512 * nbits

Example: if DRM=127 (128 directory entries) and BLS=1024 bytes, there are 32 directory entries per block, requiring 4 reserved blocks. Thus the 4 hi bits if AL0 are set, and AL0=0FH, AL1=00H

CKS = size of directory check vector
If drive media is removable, then CKS = (DRM+1)/4
If drive media is fixed, then CKS=0 (no dir records checked)

OFF = number of reserved tracks. This value is automatically added whenever SETTRK is called. It can be used to skip reserved system tracks, or for partitioning a large disk into smaller segmented sections.

Several DPH's can address the same DPB if the drive characteristics are identical. The DPB can be dynamically changed when a new drive is addressed by simply changing the pointer in the DPH since the BDOS copies the DPB values to a local area whenever the SELDKS function is invoked.

The size of the CSV (scratchpad area to check changed disks) is CKS

```
bytes.  If CKS=(DRM+1)/4, this area must be reserved. If CKS=0, no
storage is reserved.

The size of the ALV (scratchpad area for disk storage allocation info)
is (DSM/8)+1 bytes where DSM is the disk size in allocation blocks.


DISK PARAMETER TABLES FOR SPECIFIC DISKS
========================================

Standard CP/M 8" SSSD disk
--------------------------

    128 bytes/sector
    26 sectors/track
    77 tracks - 2 system tracks
    75 used tracks ==> 243.75 user KBytes/disk
    1024 bytes/block ==> 243 blocks/disk ==> DSM=242
    Directory in 2 first blocks ==> 64 directory entries ==> 241.75 KBytes data

Sector skew table (1 byte/sector):
    1, 7, 13, 19, 25, 5, 11, 17, 23, 3, 9, 15, 21,
    2, 8, 14, 20, 26, 6, 12, 18, 24, 4, 10, 16, 22

DPB

SPT 16b     26    Sectors per track
BSH  8b      3    Block shift factor
BLM  8b      7    Block shift mask
EXM  8b      0    Extent mask - null
DSM 16b    242    Disk size - 1 (in blocks)
DRM 16b     63    directory mask = dir entries - 1
AL0  8b   0C0H    Dir Alloc 0
AL1  8b      0    Dir Alloc 1
CKS 16b     16    Directory check vector size
OFF 16b      2    Track offset: 2 system tracks

Dirbuf 128 bytes
ALV     31 bytes
CSV     16 bytes

Block size 1024 bytes ==> BSH=3, BLM=7

DSM = 242 blocks

Disk size:  243.75 KBytes excluding system tracks
            250.25 KBytes including system tracks



Apple CP/M 5.25" disks
----------------------

Physical format:      A           B               C

                   ---- Standard -----     ----- Special ------
                   13-sect     16-sect     80-trk/16-sec/2-side

Bytes/sector         256         256             256
Sectors/track         13          16              16
Tracks                35          35              80
Heads                  1           1               2

Sector skew table (1 byte/sector):  no sector skew in CP/M BIOS
13-sector disks: hard sector skew
```

16-sector disks: soft sector skew in 6502 code (CP/M RWTS)

DPB	A	B	C	
SPT 16b	26	32	32	Sectors per track
BSH 8b	3	3	4	Block shift factor
BLM 8b	7	7	15	Block shift mask
EXM 8b	0	0	0	Extent mask
DSM 16b	103	127	313	Disk size - 1 (in blocks)
DRM 16b	47	63	255	Directory mask = dir entries - 1
AL0 8b	0C0H	0C0H	0F0H	Dir Alloc 0
AL1 8b	0	0	0	Dir Alloc 1
CKS 16b	12	16	64	Directory check vector size
OFF 16b	3	3	3	Track offset: 3 system tracks
Block size	1024	1024	2048	
Dir entries	48	64	256	
Dir blocks	2	2	4	
DSM+1	104	128	314	blocks
Disk size	104	128	628	KBytes (excluding system tracks)
	113.75	140	640	KBytes (including system tracks)
Dirbuf	128	128	128	bytes
ALV	14	17	40	bytes
CSV	12	16	64	bytes