# STAT

STAT DEV: liefert Informationen über den Disk Parameter Block.

```
A>stat dsk:
    A: Drive Characteristics
 6400: 128 Byte Record Capacity
  800: Kilobyte Drive  Capacity
  192: 32  Byte Directory Entries
  192: Checked  Directory Entries
  128: Records/ Extent
   16: Records/ Block
   40: Sectors/ Track
    0: Reserved Tracks
```

Der zugehörige Disk Parameter Block DPB enthält die folgenden Werte. Daraus berechnet STAT die Werte für records per block und bytes per block und daraus die Ausgaben.

```
dpb:    dw    40        ;SPT sectors per track
    db    4         ;BSF block shift factor
    db    15         ;BLM block mask
    db    0         ;EXM null mask
    dw    399         ;DSM disk size-1
    dw    191         ;DRM directory max
    db    0E0H         ;AL0 alloc 0
    db    0         ;Al1 alloc 1
    dw    48         ;CKS check size
    dw    0         ;OFS track offset


rpb := 1<<BSF = 2^BSF;            /* records/block */    = 2^4 = 16
bpb := 1<<BSF * 128 = 2^BSF * 128;    /* bytes per block */    = 2^4*128 =
2048

Drive Characteristics       := Laufwerk        = A
128 Byte Record Capacity    := (DSM+1) * rpb    = (399+1) * 16 = 6400
Kilobyte Drive  Capacity    := (DSM+1) * bpb / 1024 = (399+1) * 2048 / 1024
= 800
32  Byte Directory Entries  := DRM+1        = 191+1 = 192
Checked  Directory Entries  := CKS<<2 = CKS*4    = 48 * 4 = 192
Records/ Extent         := (EXM+1) * 128    = (0+1) * 128 = 128
Records/ Block          := rpb            = 16
Sectors/ Track          := SPT            = 40
Reserved Tracks         := OFS            = 0
```

Der originale Codeauszug von STAT, geschrieben in PL/M, ist relativ leicht zu verstehen. Lediglich die Berechnung der Gesamtkapazität ist ewas komplexer gelöst; PLM arbeitet nur mit 16 Bit. Deshalb muss die Multiplikation über diesen Zahlenbereich hinaus anders gelöst werden.

```
/* function call 32 returns the address of the disk parameter
block for the currently selected disk, which consists of:
        scptrk      (2 by) number of sectors per track
        blkshf      (1 by) log2 of blocksize (2**blkshf=blksize)
        blkmsk      (1 by) 2**blkshf-1
        extmsk      (1 by) logical/physical extents
        maxall      (2 by) max alloc number
        dirmax      (2 by) size of directory-1
        dirblk      (2 by) reservation bits for directory
        chksiz      (2 by) size of checksum vector
        offset      (2 by) offset for operating system
*/

    dpb based dpba structure
    (spt address, bls byte, bms byte, exm byte, mxa address,
     dmx address, dbl address, cks address, ofs address),
    scptrk literally 'dpb.spt',
    blkshf literally 'dpb.bls',
    blkmsk literally 'dpb.bms',
    extmsk literally 'dpb.exm',
    maxall literally 'dpb.mxa',
    dirmax literally 'dpb.dmx',
    dirblk literally 'dpb.dbl',
    chksiz literally 'dpb.cks',
    offset literally 'dpb.ofs';

declare bpb address; /* bytes per block */

set$bpb: procedure;
    call set$dpb; /* disk parameters set */
    bpb = shl(double(1),blkshf) * sectorlen;
    end set$bpb;

getalloca: procedure address;
    /* get base address of alloc vector */
    return mon3(27,0);
    end getalloca;

add$block: procedure(ak,ab);
    declare (ak, ab) address;
    /* add one block to the kilobyte accumulator */
    declare kaccum based ak address; /* kilobyte accum */
    declare baccum based ab address; /* byte accum */
    baccum = baccum + bpb;
        do while baccum >= 1024;
        baccum = baccum - 1024;
        kaccum = kaccum + 1;
        end;
    end add$block;

count: procedure(mode) address;
```

```
    declare mode byte; /* true if counting 0's */
    /* count kb remaining, kaccum set upon exit */
    declare
        ka  address,  /* kb accumulator */
        ba  address,  /* byte accumulator */
        i   address,  /* local index */
        bit byte;     /* always 1 if mode = false */
    ka, ba = 0;
    bit = 0;
        do i = 0 to maxall;
        if mode then bit = getalloc(i);
        if not bit then call add$block(.ka,.ba);
        end;
    return ka;
    end count;

drivestatus: procedure;
    declare
        rpb address,
        rpd address;
    pv: procedure(v);
        declare v address;
        call crlf;
        call pdecimal(v,10000);
        call printchar(':');
        call printb;
        end pv;
    /* print the characteristics of the currently selected drive */
    call print(.('    ',0));
    call printchar(cselect+'A');
    call printchar(':');
    call printx(.(' Drive Characteristics',0));
    rpb = shl(double(1),blkshf); /* records/block=2**blkshf */
    if (rpd := (maxall+1) * rpb) = 0 and (rpb <> 0) then
        call print(.('65536: ',0)); else
        call pv(rpd);
        call printx(.('128 Byte Record Capacity',0));
    call pv(count(false));  /* (maxall+1)*bpb/1024 */
        call printx(.('Kilobyte Drive  Capacity',0));
    call pv(dirmax+1);
        call printx(.('32  Byte Directory Entries',0));
    call pv(shl(chksiz,2));
        call printx(.('Checked  Directory Entries',0));
    call pv((extmsk+1) * 128);
        call printx(.('Records/ Extent',0));
    call pv(rpb);
        call printx(.('Records/ Block',0));
    call pv(scptrk);
        call printx(.('Sectors/ Track',0));
    call pv(offset);
        call printx(.('Reserved Tracks',0));
```

```
    call crlf;
    end drivestatus;
```