

Listing: FIXPARTP.PAS

```

{*****}
{*                F I X P A R T P . P A S                *}
{*-----}
{* Task          : Displays hard disk partitioning.      *}
{*-----}
{* Author        : Michael Tischer                       *}
{* Developed on   : 04/26/89                               *}
{* Last update    : 04/07/95                               *}
{*-----}
{* Call          : FIXPARTP [ Drive_number ]              *}
{*               Default is drive 0 (C:)                  *}
{*-----}
{*****}

```

```

uses Dos;                                     { Add DOS unit }

```

```

{== Type declarations =====}

```

```

type  SecPos    = record                      { Describes a sector position }
                        Head : byte;           { Read/write head }
                        SecCyl : word;         { Sector and cylinder number }
                    end;

```

```

    PartEntry = record                      { Partition table entry }
                        Status   : byte;      { Partition status }
                        StartSec : SecPos;    { First sector }
                        PartTyp  : byte;      { Partition type }
                        EndSec   : SecPos;    { Last sector }
                        SecOfs   : longint;   { Offset of boot sector }
                        SecNum   : longint;   { Number of sectors }
                    end;

```

```

PartSec    = record                                { Describes partition sector }
    BootCode : array [0..$1BD] of byte;
    PartTable : array [1..4] of PartEntry;
    IdCode    : word;                               { $AA55 }
end;

```

```

{*****}
*  ReadPartSec : Reads a partition sector from the hard drive.  *
**-----**
*  Input   : - DS          : BIOS code for drive (80H, 81H, etc.) *
*           - Head        : Number of read/write heads          *
*           - SctCyl      : Sector/cylinder numbers in BIOS format *
*           - Buf         : Buffer to which sector is passed      *
{*****}

```

```

function ReadPartSec( DS, Head : byte;
                     SctCyl    : word;
                     var Buf    : PartSec ) : boolean;

```

```

var Regs : Registers;      { Processor registers for interrupt call }

```

```

begin

```

```

    Regs.AX := $0201;          { Funct. no.: READ for first sector }
    Regs.DI := DS;             { Pass other parameters }
    Regs.DI := Head;          { to their respective }
    Regs.CX := SctCyl;         { registers }
    Regs.ES := seg( Buf );
    Regs.BX := ofs( Buf );

```

```

    Intr( $13, Regs);          { Call hard drive interrupt }
    ReadPartSec := ( Regs.Flags and 1 ) = 0;    { Carry flag = error }

```

```

end;

```

```

{*****}
*   GetSecCyl: Gets the combined sector/cylinder coding of the BIOS   *
*   sector and cylinder number.                                     *
**-----**
*   Input   : SctCyl   : Value to be decoded                         *
*             Sector   : Sector variable reference                  *
*             Cylinder : Cylinder variable reference                *
{*****}

```

```

procedure GetSecCyl( SctCyl : word; var Sector, Cylinder : integer );

```

```

begin

```

```

    Sector := SctCyl and 63;           { Mask bits 6 and 7 }

```

```

    Cylinder := hi( SctCyl ) + ( lo( SctCyl ) and 192 ) shl 2;

```

```

end;

```

```

{*****}
*   ShowPartition: Displays hard drive partitioning on the screen.*
**-----**
*   Input   : DS : Number of the corresponding hard drive          *
{*****}

```

```

procedure ShowPartition( ds : byte );

```

```

var Head       : byte;           { Head of current partition }
    SecCyl     : byte;           { Sector and cylinder of current partition }
    ParSec     : PartSec;        { Current partition sector }
    Entry      : byte;           { Loop counter }
    Sector,    : integer;        { Get sector and }
    Cylinder   : integer;        { cylinder number }
    Regs       : Registers;      { Processor registers for interrupt call }

```

```

begin
  writeln;
  ds := ds + $80;                                { Prepare drive number for BIOS }
  if ReadPartSec( ds, 0, 1, ParSec ) then          { Read partition sector }
  begin                                             { Sector could be read }
    Regs.AH := 8;                                { Funct. no.: Read drive ID }
    Regs.DL := ds;
    Intr( $13, Regs);                             { Call hard drive interrupt }
    GetSecCyl( Regs.CX, Sector, Cylinder );
    writeln('-----'+
      '-----+');
    writeln('      Drive ', ds-$80, ': ', Regs.DH+1:2,
      ' heads, ', Cylinder:5, ' cylinders, ',
      Sector:3, ' sectors ');
    writeln('      Partition Table in Partition Sector ');
    writeln('      ');
    writeln('-----'+
      '-----');
    writeln('      End      Dis.from      Start ');
    writeln('      No      Boot      Type      Head Cyl. Sec. ');
    writeln('      Head Cyl. Sec. Boot Sec.      Total ');
    writeln('-----+-----+-----+-----+');
    writeln('-----+-----+-----');
  for Entry:=1 to 4 do                             { Show entries }
  with ParSec.PartTable[ Entry ] do
  begin
    write(' ', Entry, ' ');
    if Status = $80 then write ( ' Y ' )
    else write ( ' N ' );
    write(' ');
    case PartTyp of
      { Compute partition type }

```



```

if ParamCount = 1 then          { User entered different argument? }
begin
    val( ParamStr(1), DS, ErrArg );    { Convert ASCII to decimal }
    if ErrArg <> 0 then                { Error during conversion? }
    begin
        writeln(#13#10'Invalid drive number');
        exit;                        { End program }
    end;
end;
ShowPartition( DS );            { Display partition sector }
end.

```