

```

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

```

```

#include <dos.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

```

```

/*== Constants =====*/

```

```

#define TRUE ( 1 == 1 )
#define FALSE ( 1 == 0 )

```

```

/*== Macros =====*/

```

```

#define HI(x) ( *((BYTE *) (&x)+1) ) /* Returns high byte of word */
#define LO(x) ( *((BYTE *) &x) ) /* Returns low byte of word */

```

```

/*== Type declarations =====*/

```

```

typedef unsigned char BYTE;
typedef unsigned int WORD;

```

```

typedef struct { /* Describes a sector position */
    BYTE Head; /* Read/write head */
    WORD SecCyl; /* Sector and cylinder number */
} SECPOS;

```

```

typedef struct { /* Partition table entry */
    BYTE Status; /* Partition status */
    SECPOS StartSec; /* First sector */
    BYTE PartTyp; /* Partition type */
    SECPOS EndSec; /* Last sector */
    unsigned long SecOfs; /* Offset of boot sector */
    unsigned long SecNum; /* Number of sectors */
} PARTENTRY;

```

```

typedef struct { /* Describes partition sector */
    BYTE BootCode[ 0x1BE ];
    PARTENTRY PartTable[ 4 ];
    WORD IdCode; /* 0xAA55 */
} PARTSEC;

```

```

typedef PARTSEC far *PARSPTR;
/* Ptr. to partition sector in memory */

```

```

/*****
/* ReadPartSec : Reads a partition sector from the hard drive.
/* Input : - DS : BIOS code for drive (0x80, 0x81, etc.)
/* - Head : Number of read/write heads
/* - SctCyl : Sector/cylinder numbers in BIOS format
/* - Buf : Buffer to which sector is passed
/* Output : TRUE if sector can be read without errors,
/* otherwise FALSE
*****/

```

```

BYTE ReadPartSec( BYTE Drive, BYTE Head, WORD SecCyl, PARSPTR Buf )

```

```

{
    union REGS Regs; /* Processor registers for interrupt call */
    struct SREGS SRegs;

    Regs.x.ax = 0x0201; /* Funct. no.: READ for first sector */
    Regs.h.dl = Drive; /* Pass other parameters */

```

```

Regs.h.dh = Head; /* to their respective */
Regs.x.cx = SecCyl; /* registers */
Regs.x.bx = FP_OFF( Buf );
SRegs.es = FP_SEG( Buf );

int86x( 0x13, &Regs, &Regs, &SRegs );
/* Call hard drive interrupt */
return !Regs.x.cflag;
}

/*****
/* GetSecCyl: Gets the combined sector/cylinder coding of the
/* BIOS sector and cylinder number.
/* Input : SecCyl : Value to be decoded
/* Sector : Sector variable reference
/* Cylinder : Cylinder variable reference
/* Output : None
*****/

void GetSecCyl( WORD SecCyl, int *Sector, int *Cylinder )

{
*Sector = SecCyl & 63; /* Mask bits 6 and 7 */
*Cylinder = HI(SecCyl) + ( ( (WORD) LO(SecCyl) & 192 ) << 2 );
}

/*****
/* ShowPartition: Displays hard drive partitioning on the screen.
/* Input : DS : Number of the corresponding hard drive
/* (0, 1, etc.)
/* Output : None
*****/

void ShowPartition( BYTE LW )
{
#define AP ( ParSec.PartTable[ Entry ] )

BYTE Head, /* Head of current partition */
Entry; /* Loop counter */
int Sector, /* Get sector and */
Cylinder; /* cylinder number */
PARTSEC ParSec; /* Current partition sector */
union REGS Regs; /* Processor registers for interrupt call */

printf("\n");
LW |= 0x80; /* Prepare drive number for BIOS */
if ( ReadPartSec( LW, 0, 1, &ParSec ) ) /* Read partition sector */
{
/* Sector could be read */
Regs.h.ah = 8; /* Funct. no.: Read drive ID */
Regs.h.dl = LW;
int86( 0x13, &Regs, &Regs ); /* Call hard drive interrupt */
GetSecCyl( Regs.x.cx, &Sector, &Cylinder );
printf( "-----"
"-----+\\n");
printf( " Drive %2d: %2d Heads %4d"
" Cylinders %3d Sectors \\n",
LW-0x80, Regs.h.dh+1, Cylinder, Sector );
printf( " Partition table in Partition Sector \\n");
printf( "-----"
"-----\\n");
printf( " Start End"
" Dist fm \\n");
printf( "Nr Boot Type Head Cyl. Sec Head"
" Cyl. Sec BootSec Total\\n");
printf( "++-----+-----+-----+-----"
"-----+-----+-----\\n");

/*-- Get partition table -----*/
for ( Entry=0; Entry < 4; ++Entry )
{
printf( " %d", Entry );
if ( AP.Status == 0x80 ) /* Partition active? */
printf( " Yes");
else
printf( " No ");
printf( " ");
}
}

```

```

switch( AP.PartTyp )          /* Compute partition type */
{
    case 0x00 : printf( "Not allocated      " );
        break;
    case 0x01 : printf( "DOS, 12-bit FAT    " );
        break;
    case 0x02 :
    case 0x03 : printf( "XENIX              " );
        break;
    case 0x04 : printf( "DOS, 16-bit FAT    " );
        break;
    case 0x05 : printf( "DOS, ext. partition" );
        break;
    case 0x06 : printf( "DOS 4.0 > 32 MB    " );
        break;
    case 0xDB : printf( "Concurrent DOS     " );
        break;
    default  : printf( "Unknown (%3d)      ",
        ParSec.PartTable[ Entry ].PartTyp );
}

/*-- Get physical and logical parameters -----*/
GetSecCyl( AP.StartSec.SecCyl, &Sector, &Cylinder );
printf( "%2d %5d %3d ", AP.StartSec.Head, Cylinder, Sector );
GetSecCyl( AP.EndSec.SecCyl, &Sector, &Cylinder );
printf( "%2d %5d %3d ", AP.EndSec.Head, Cylinder, Sector );
printf( "%6lu %6lu \n", AP.SecOfs, AP.SecNum );
}
printf( "+-----"
      "-----+\n" );
}
else
    printf("Error during boot sector access\n");
}

/*****
*                               M A I N   P R O G R A M                               *
*****/

int main( int argc, char *argv[] )
{
    int Drive;

    printf( "\n_____ FIXPARTC - (c)"
        " 1989, 92 by MICHAEL TISCHER \n" );
    Drive = 0;                               /* Default is first hard drive */
    if ( argc == 2 )                          /* User entered different argument? */
    {                                          /* Yes */
        Drive = atoi ( argv[1] );
        if ( Drive == 0 && *argv[1] != '0' )
        {
            printf("\nInvalid drive number!");
            return( 1 );                    /* End program */
        }
    }
    ShowPartition( (BYTE) Drive );           /* Display partition sector */
    return( 0 );
}

```