

# Listing: FIXPARTC.C

```

/*****
/*                                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 {
    BYTE Kopf;           /* Describes a sector position */
    WORD SecZyl;          /* Read/write head */
    } SECPOS;           /* Sector and cylinder number */

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

typedef struct {
    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 Laufwerk, BYTE Kopf, WORD SekZyl, 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 = Laufwerk;    /* Pass other parameters */
    Regs.h.dh = Kopf;        /* to their respective */
    Regs.x.cx = SekZyl;      /* 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;
}

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

void GetSecZyl( WORD SekZyl, int *Sektor, int *Zylinder )

```

```

{
    *Sektor    = SekZyl & 63;                /* Mask bits 6 and 7 */
    *Zylinder = HI(SekZyl) + ( ( (WORD) LO(SekZyl) & 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      Kopf,                /* Head of current partition */
             Entry;                /* Loop counter */
    int       Sektor,              /* Get sector and */
             Zylinder;            /* 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 */
        GetSecZyl( Regs.x.cx, &Sektor, &Zylinder );
        printf( "+-----\n" );
        printf( "-----+\n" );
    }
}

```

```

printf( "      Drive %2d:      %2d Heads %4d"
        " Cylinders %3d Sectors                                \n",
        LW-0x80, Regs.h.dh+1, Zylinder, Sektor );
printf( "      Partition table in Partition Sector              \n");
printf( "      -----" );
printf( "      ----- \n");
printf( "      End      Dist fm      Start      " );
printf( "      Nr      Boot      Typ      Head Cyl. Sec      " );
printf( "      Head Cyl. Sec      BootSec Total      \n");
printf( "      -+-----+-----+-----+-----+-----+-----+
      -----+-----+-----+-----+-----+-----+----- \n");

/*-- Get partition table -----*/
for ( Entry=0; Entry < 4; ++Entry )
{
    printf( "      %d      ", Entry );
    if ( AP.Status == 0x80 )
        printf( " Yes" );
    else
        printf( " No " );
    printf( " " );
    switch( AP.PartTyp )
    {
        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 -----*/
GetSecZyl( AP.StartSec.SecZyl, &Sektor, &Zylinder );
printf( " %2d %5d %3d ", AP.StartSec.Kopf, Zylinder, Sektor );
GetSecZyl( AP.EndSec.SecZyl, &Sektor, &Zylinder );
printf( " %2d %5d %3d ", AP.EndSec.Kopf, Zylinder, Sektor );
printf( " %6lu %6lu  \n", AP.SecOfs, AP.SecAnz );
}
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 Laufwerk;

```

```

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

```