

Listing: FIXPARTB.BAS

```

*****
' *                                FIXPARTB.BAS                                *
'-----
' *      Task                      : Displays hard disk partitioning.          *
'-----
' *      Author                   : Michael Tischer                          *
' *      Developed on              : 05/21/91                                  *
' *      Last update               : 04/07/95                                  *
'-----
' *      Call                     : FIXPARTB [Drive_number]                   *
' *                               Default is drive 0 (C:)                      *
*****
DECLARE SUB GetSecCyl (SecCyl AS LONG, Sector AS INTEGER, Cylinder AS INTEGER)
DECLARE FUNCTION ReadPartSec% (ds%, Head%, SecCyl%, PartiEntry() AS ANY)
DECLARE SUB Showpartition (ds AS INTEGER)
DECLARE FUNCTION MLNG& (TNum AS INTEGER)

'$INCLUDE: 'QB.BI'                                'Add include file for interrupt call

CONST TRUE = -1                                    'Define the truth
CONST FALSE = NOT TRUE

TYPE SecPos                                         'Describes a sector position
    Head AS INTEGER                                'Read-write head
    SecCyl AS LONG                                 'Sector and cylinder number
END TYPE

TYPE PartiEntry                                     'Partition table entry
    Status AS INTEGER                              'Partition status
    StartSec AS SecPos                             'First sector
    PartType AS INTEGER                            'Partition type

```

```

        EndSec AS SecPos                                'Last sector
        SecOfs AS LONG                                  'Offset of boot sector
        SecNum AS LONG                                  'Number of sectors
END TYPE

DIM HarDrive AS INTEGER                                'Variable for hard drive argument

CLS
PRINT " FIXPARTB - (C)";
PRINT " 1991, 92 by Michael Tischer "
HarDrive = 0                                           'Default is first hard drive
IF COMMAND$ <> "" THEN                                'User entered different argument
    HarDrive = VAL(COMMAND$)                          'Convert ASCII to decimal
END IF
CALL Showpartition(HarDrive)                          'Display partition sector
END

' *****
' * GetSecCyl : Gets the combined sector/cylinder coding of the BIOS *
' *           sector and cylinder number. *
' * Input      : Sector variable in BIOS coding *
' * Output     : References to sector and cylinder variables *
' *****
SUB GetSecCyl (SecCyl AS LONG, Sector AS INTEGER, Cylinder AS INTEGER)

    Sector = SecCyl AND 63                             'Mask bits 6 and 7
    Cylinder = SecCyl \ 256 + ((SecCyl MOD 256) AND 192) * 4
END SUB

' *****
' * MLNG      : Converts an integer to its representative long value. *
' * Input     : Integer value *

```

```

'* Output : The long number containing corresponding register data *
'*****
FUNCTION MLNG& (TNum AS INTEGER)

IF TNum >= 0 THEN                                'Bit 16 unset
    MLNG = TNum                                'Integer corresponds to register contents
ELSE                                              'Bit 16 set
    MLNG = 65536 + TNum                        'This value corresponds to register contents
END IF

END FUNCTION

'*****
'* ReadPartSec : Reads a partition sector from the hard drive. *
'* Input      : DS          = Drive code *
'*            : Head       = Number of read/write heads *
'*            : SecCyl     = Sector/cylinder numbers in BIOS format *
'* Output n   : PartEntry() = The 4 partition sector tables *
'*****
FUNCTION ReadPartSec% (ds%, Head%, SecCyl%, PrtnEntry() AS PartEntry)

DIM Register AS RegTypeX      'Processor registers for interrupt call
DIM PartSector AS STRING * 512 'Get partition sector
DIM IdCode AS LONG           'Get partition ID code
DIM i AS INTEGER             'Loop counter
DIM j AS INTEGER             'Loop counter

Register.ax = &H201           'Funct. no.: READ for first sector
Register.cx = SecCyl%         'Sector and cylinder of current partition
Register.dx = Head% * 256 + ds% ' Head and drive number
Register.es = VARSEG(PartSector) 'Partition sector segment address
Register.bx = VARPTR(PartSector) 'Partition sector offset address

```

```

CALL INTERRUPTX(&H13, Register, Register)      'Call hard drive interrupt
DEF SEG = VARSEG(PartSector)      'Set segment address for PEEK command
Pointer = VARPTR(PartSector) + &H1BD      'Pointer to first partition entry
offset = 0      'Offset of first partition entry
FOR i = 1 TO 4      'Read all 4 partition entries
    PrtnEntry(i).Status = PEEK(Pointer + offset + 1)
    PrtnEntry(i).StartSec.Head = PEEK(Pointer + offset + 2)
    PrtnEntry(i).StartSec.SecCyl = PEEK(Pointer + offset + 4) * 256& +
PEEK(Pointer + offset + 3)
    PrtnEntry(i).PartType = PEEK(Pointer + offset + 5)
    PrtnEntry(i).EndSec.Head = PEEK(Pointer + offset + 6)
    PrtnEntry(i).EndSec.SecCyl = PEEK(Pointer + offset + 8) * 256& + PEEK(Pointer
+ offset + 7)
    PrtnEntry(i).SecOfs = 0
    FOR j = 0 TO 3
        PrtnEntry(i).SecOfs = PrtnEntry(i).SecOfs * 256& + PEEK(Pointer + offset +
12 - j)
    NEXT
    PrtnEntry(i).SecNum = 0
    FOR j = 0 TO 3
        PrtnEntry(i).SecNum = PrtnEntry(i).SecNum * 256& + PEEK(Pointer + offset +
16 - j)
    NEXT
    offset = offset + 16      'Set offset of next partition entry
NEXT
IdCode = PEEK(Pointer + offset + 2) * 256& + PEEK(Pointer + offset + 1)
IF (Register.flags AND 1) = 0 THEN
    ReadPartSec = TRUE
END IF
END FUNCTION

```



```

PRINT "      End      Dis.from      "
PRINT "NoBootType      Head ";
PRINT "Cyl. Sec.Head Cyl. Sec.Boot Sec. Total  "
PRINT ";";
PRINT ""
FOR Entry = 1 TO 4
    PRINT USING "##"; Entry;
    IF PartiEntry(Entry).Status = &H80 THEN
        PRINT " Y  ";
    ELSE
        PRINT " N  ";
    END IF
    SELECT CASE PartiEntry(Entry).PartType
        CASE 0
            PRINT "Not allocated      ";
        CASE 1
            PRINT "DOS, 12-bit FAT      ";
        CASE 2 OR 3
            PRINT "Xenix      ";
        CASE 4
            PRINT "DOS, 16-bit FAT      ";
        CASE 5
            PRINT "DOS, ext. partition ";
        CASE 6
            PRINT "DOS 4.0 > 32 Meg      ";
        CASE &HDB
            PRINT "Concurrent DOS      ";
        CASE ELSE
            PRINT USING "Unknown      (###)  "; PartiEntry(Entry).PartType
    END SELECT
    CALL GetSecCyl(PartiEntry(Entry).StartSec.SecCyl, Sector, Cylinder)
    PRINT USING "###  ###  "; PartiEntry(Entry).StartSec.Head; Cylinder;

```

```
        PRINT USING "### "; Sector;
        CALL GetSecCyl(PartiEntry(Entry).EndSec.SecCyl, Sector, Cylinder)
        PRINT USING "### #### ### "; PartiEntry(Entry).EndSec.Head; Cylinder;
Sector;

        PRINT USING " ####### "; PartiEntry(Entry).SecOfs;
        PRINT USING " ####### "; PartiEntry(Entry).SecNum
    NEXT
    PRINT " ";
    PRINT " "
ELSE
    PRINT "Error during boot sector access"
END IF

END SUB
```