

Listing: MCBB.BAS

```

*****
' *                               M C B B . B A S                               *
' *-----*
' * Task : Displays any memory block allocated by DOS. *
' * QuickBASIC and the QB.LIB must be loaded using *
' * QB /L QB *
' * before loading and running this file. *
' *-----*
' * Author : Michael Tischer *
' * Developed on : 05/16/91 *
' * Last update : 04/07/95 *
*****
DECLARE SUB TraceMCB ()
DECLARE SUB FirstMCB (Adr AS ANY)
DECLARE SUB Dump (Adr AS ANY, Nmb%)
DECLARE FUNCTION HexByte$ (HByte%)
DECLARE FUNCTION GetDosVer% ()
DECLARE FUNCTION GetWord& (SegAdr AS LONG, OfsAdr AS LONG)
DECLARE FUNCTION HexString$ (HexVal&)

'$INCLUDE: 'qb.bi' 'Include QB file

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

TYPE AddrType 'Pointer to an address
    OfsAdr AS LONG 'Offset address
    SegAdr AS LONG 'Segment address
END TYPE
CLS 'Clear screen
PRINT "MCBB - (c) 1988, 91 by Michael Tischer": PRINT : PRINT

```

CALL TraceMCB

'Display MCB group

END

```
*****
'* DUMP      : Display a memory range as hex and ASCII dumps.      *
'* Input     : SegAdr = Segment address of memory range to be dumped *
'*           : Nmb     = Number of lines to be dumped (in 16 byte units) *
'* Output    : None                                              *
*****
SUB Dump (SegAdr AS LONG, Nmb AS INTEGER)
```

DIM HexStr AS STRING * 2

'Get 2-digit hex number

DIM Offset AS LONG

'Offset in memory range

HexStr = "zz"

'Create hex string

PRINT

PRINT "DUMP | 0123456789ABCDEF 00 01 02 03 ";

PRINT "04 05 06 07 08 09 0A 0B 0C 0D 0E 0F"

PRINT "-----+-----";

PRINT "-----"

Offset = 0

'Start with the first byte

DEF SEG = SegAdr

'Define the segment address

WHILE Nmb > 0

'Execute loop Nmb times

PRINT HexString\$(Offset); " | ";

FOR z = 0 TO 15

'Process every 16 bytes

IF PEEK(Offset + z) >= 32 THEN

'Valid ASCII characters?

PRINT CHR\$(PEEK(Offset + z));

'Yes --> Display

ELSE

PRINT " ";

'No --> Display spaces

END IF

NEXT

PRINT " ";

'Set cursor to hex portion

```

FOR z = 0 TO 15                                'Process every 16 characters
  PRINT HexByte$(PEEK(Offset + z)); " "; 'Display byte as hex string
NEXT
PRINT                                           'New line
Offset = Offset + 16                          'Set offset in the next line
Nmb = Nmb - 1                                  'Decrement number of lines
WEND
PRINT
END SUB

```

```

! *****
! * FirstMCB : Returns a pointer to the first MCB. *
! * Input : None *
! * Output : Pointer to first MCB in the MCBAdr variable *
! *****
SUB FirstMCB (MCBAdr AS AdrType)

```

```

DIM Register AS RegTypeX 'Processor registers for interrupt call

Register.ax = &H52 * 256 'Func. No: Get DOS info block's address
CALL INTERRUPTX(&H21, Register, Register) 'Call DOS interrupt
'-- (ES:BX - &H4) = ES-1:12 returns pointer address to first MCB -----
DEF SEG = Register.es - 1 'Define segment address
MCBAdr.OfsAdr = PEEK(Register.bx + 13) * 256& + PEEK(Register.bx + 12)
MCBAdr.SegAdr = PEEK(Register.bx + 15) * 256& + PEEK(Register.bx + 14)
END SUB

```

```

! *****
! * GetDosVer : Determines version of DOS in use. *
! * Input : None *
! * Output : DOS version number (30 = DOS 3.0, 33 = DOS 3.3, etc.) *
! *****

```

```

FUNCTION GetDosVer%

DIM Register AS RegType          'Processor registers for interrupt call

Register.ax = &H30 * 256          'AH = Funct. No: Get DOS version
CALL INTERRUPT(&H21, Register, Register) 'Call DOS interrupt 21H
GetDosVer = INT(Register.ax \ 256) + (Register.ax MOD 256) * 10
END FUNCTION

! *****
! * Getword   : Reads a word from a memory address as a long number.      *
! * Input     : SegAdr = Segment address of the word                     *
! *           : OfsAdr = Offset address of the word                       *
! * Output    : Contents of the word as a long number                     *
! *****
FUNCTION GetWord& (SegAdr AS LONG, OfsAdr AS LONG)

DEF SEG = SegAdr                  'Set segment address
GetWord& = PEEK(OfsAdr + 1) * 256& + PEEK(OfsAdr) 'Read memory location
END FUNCTION

! *****
! * Hexbyte   : Converts a byte into a hexadecimal string.               *
! * Input     : Hbyte = the byte                                          *
! * Output    : A string                                                  *
! * Info      : This replaces the HEX$ function, which doesn't          *
! *           : consistently return two-digit hexadecimal numbers.      *
! *****
FUNCTION HexByte$ (HByte AS INTEGER)

DIM HexSt AS STRING * 2          'Get hex string

```

```

MID$(HexSt, 1, 1) = HEX$(HByte \ 16)           'First digit
MID$(HexSt, 2, 1) = HEX$(HByte MOD 16)         'Second digit
HexByte$ = HexSt                               'Return resulting string
END FUNCTION

```

```

' *****
' * HexString : Converts a number into a hexadecimal string.      *
' * Input    : Value to be converted                             *
' * Output   : Resulting hex string                               *
' * Info     : This replaces the HEX$ function, which doesn't    *
' *           : consistently return two-digit hexadecimal numbers.*
' *****
FUNCTION HexString$ (HexValVar AS LONG)

```

```

DIM Nibble AS INTEGER           'Lowest nibble of the word
DIM HexVal AS LONG              ' Argument must be stored
                                'with reference parameters
DIM HStr AS STRING * 4         'The hex string to be converted

```

```

HexVal = HexValVar              'Store the argument of the word to be converted
HStr = "xxxx"                  'Create string
FOR counter = 0 TO 3           'Change four digits
    Nibble = HexVal AND &HF     'Get upper four bits
    MID$(HStr, 4 - counter, 1) = HEX$(Nibble) 'Convert nibble into hex
    HexVal = HexVal \ 16        'Shift hex number right by four bit positions
NEXT
HexString$ = HStr               'Pass resulting string
END FUNCTION

```

```

' *****
' * TraceMCB : Display list of MCBs.                               *
' * Input    : None                                                *

```

```

'* Output      : None
'*****
SUB TraceMCB

CONST kom = "COMSPEC="                'Declare "COMSPEC=" as a constant

DIM CurMCB AS AdrType                  'Pointer to MCB
DIM ID AS STRING * 1                  ' "M" = block follows, "Z" = End
DIM PSP AS LONG                       'Segment address of corresponding PSP
DIM Spacing AS LONG                   'Number of paragraphs - 1
DIM MemPtr AS LONG                    'Pointer in memory
DIM NrMCB AS INTEGER                  'Number of MCBs available
DIM z AS INTEGER                      'Loop counter
DIM EndIt AS INTEGER                  'Cancel condition
DIM CurOfs AS LONG

DosVer = GetDosVer                    'Get DOS version
NrMCB = 1                             'Start with first MCB
EndIt = FALSE
CALL FirstMCB(CurMCB)                 'Get pointer to first MCB
DO
    CurOfs = CurMCB.OfsAdr             'Load offset address
    DEF SEG = CurMCB.SegAdr            'Define segment address for Peek()
    ID = CHR$(PEEK(CurOfs))             'Read first MCB
    PSP = GetWord&(CurMCB.SegAdr, CurOfs + &H1)
    Spacing = GetWord&(CurMCB.SegAdr, CurOfs + &H3)

    IF ID = "Z" THEN                   'Last MCB read?
        EndIt = TRUE                  'End loop - end retrieving new MCBs
    END IF
    PRINT "MCB number      ="; NrMCB
    PRINT "MCB address    = "; HexString$(CurMCB.SegAdr); " ";

```

```

PRINT HexString$(CurOfs)
PRINT "Memory address = "; HexString$(CurMCB.SegAdr + 1); ":";
PRINT HexString$(CurOfs)
PRINT "ID          = "; ID
PRINT "PSP address = "; HexString$(PSP); ":0000"
PRINT "Size       = "; HexString$(Spacing); " paragraphs ( ";
PRINT Spacing * 16; " bytes )"
PRINT "Contents   = ";

'---- Handle MCB as an environment? -----
z = 0                                'Start comparison with first byte
MemPtr = CurMCB.SegAdr + 1           'Pointer in RAM
DEF SEG = MemPtr                     'Set segment address for Peek()
WHILE (z <= 7) AND MID$(kom, z + 1, 1) = CHR$(PEEK(CurMCB.OfsAdr + z))
    z = z + 1                         'next character
WEND
IF z > 7 THEN                         'COMSPEC = String found
    PRINT "Environment "
    IF DosVer > 30 THEN               'DOS Version 3.0 or higher?
        PRINT "Program name      = "; 'Yes --> List program name
        z = 0                        'Start with first byte
        DO
            z = z + 1                'Search for null string
        LOOP UNTIL PEEK(CurOfs + z) = 0 AND PEEK(CurOfs + z + 1) = 0
        IF PEEK(CurOfs + z + 2) = 1 AND PEEK(CurOfs + z + 3) = 0 THEN
            '--- Program name found -----
            z = z + 4                'Place z at first character of the program name
            DO
                PRINT CHR$(PEEK(CurOfs + z)); 'Show program names
                z = z + 1            'Display character
            LOOP UNTIL PEEK(CurOfs + z) = 0 'Next character
            PRINT                     'until end of string
        
```

```

ELSE                                                    'No program name found
    PRINT "Unknown"
END IF
END IF

'---- Display environment string -----

PRINT "Environment string:"
z = 0                                                    'Start with first byte in allocated range
WHILE PEEK(CurOfs + z) <> 0                               'Repeat until null string
    PRINT "      ";                                       'Indent line
    DO                                                    'Display string
        PRINT CHR$(PEEK(CurOfs + z));                   'Display character
        z = z + 1                                         'Next character
    LOOP UNTIL PEEK(CurOfs + z) = 0                       'Loop until end of string
    z = z + 1                                             'Set to start of next string
    PRINT                                                'End line
WEND
ELSE
'---- Handle it as a PSP (if INT 20H -----
'---- (code &HCD &H20) starts the code) -----

MemPtr = CurMCB.SegAdr + 1                               'Set pointer in RAM
IF PEEK(CurOfs) = &HCD AND PEEK(CurOfs + 1) = &H20 THEN
    PRINT "PSP (with program following)"                 'Handled as a PSP
ELSE                                                      'INT 20H could not be implemented
    PRINT "Unidentifiable as a program"
    CALL Dump(MemPtr, 5)                                 'Dump the first 5*16 bytes
END IF
END IF

PRINT "_____ Press any key to continue _____";

```



```

PRINT " _____ "
DO
a$ = INKEY$
LOOP UNTIL a$ <> ""
IF NOT EndIt THEN
    CurMCB.SegAdr = CurMCB.SegAdr + Spacing + 1
    NrMCB = NrMCB + 1
END IF
LOOP UNTIL EndIt
END SUB

```