

```

/*****
/*
/*----- M C B C . C -----*/
/* Task : Displays any memory block allocated by DOS. */
/*-----*/
/* Author : Michael Tischer */
/* Developed on : 08/23.88 */
/* Last update : 04/07/95 */
/*-----*/
/* Memory model : SMALL */
/* Info : When compiling with Microsoft C, use */
/* CL /AS /Zp MCBC.C */
*****/

/*== Add include files =====*/

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

/*== Type definitions =====*/

typedef unsigned char BYTE; /* Create a byte */
typedef unsigned int WORD;
typedef BYTE BOOLEAN;
typedef BYTE far *FB; /* FAR pointer to a byte */

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

#define TRUE ( 0 == 0 ) /* Needed when working with BOOLEANs */
#define FALSE ( 1 == 0 )

/*== Structures and unions =====*/

struct MCB { /* Describes an MCB in memory */
    BYTE id_code; /* 'M' = block follows, 'Z' = end */
    WORD psp; /* Segment address of appropriate PSP */
    WORD spacing; /* Number of reserved paragraphs */
};

typedef struct MCB far *MCBPtr; /* FAR pointer to an MCB */

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

#ifdef MK_FP /* Is MK_FP already defined? */
#undef MK_FP
#endif

#define MK_FP(seg, ofs) ((void far *) ((unsigned long) (seg)<<16|(ofs)))

/*****
* Function : F I R S T _ M C B *
*-----*
* Task : Returns a pointer to the first MCB. *
* Input : None *
* Output : Pointer to the first MCB *
*****/

MCBPtr first_mcb( void )
{
    union REGS regs; /* Store the processor registers */
    struct SREGS sregs; /* Store the segment register */

    regs.h.ah = 0x52; /* Func. no.: Get DOS info block's address */
    intdosx( &regs, &regs, &sregs ); /* Call DOS interrupt 21H */

    /*-- ES:(BX-4) points to the first MCB, create pointer -----*/

    return( *((MCBPtr far *) MK_FP( sregs.es-1, regs.x.bx+12 )) );
}

/*****
* Function : D U M P *
*-----*
* Task : Display a memory range as hex and ASCII dumps. *
* Input : DPTR : Pointer to the memory range to be dumped *
* NUML : Number of 16-byte lines to be dumped *
*****/

```

```

*   Output           : None
*****/

void dump( FB dptr, BYTE numl)
{
    FB    lptr;           /* Floating pointer for dump line display */
    WORD offset;          /* Offset address relative to dptr */
    BYTE i;               /* Loop counter */

    printf("\nDUMP | 0123456789ABCDEF      00 01 02 03 04 05 06 07 08");
    printf(" 09 0A 0B 0C 0D 0E 0F\n");
    printf("-----+-----");
    printf("-----\n");

    for (offset=0; numl-- ; offset += 16, dptr += 16)
    {
        printf("%04x | ", offset);
        for (lptr=dptr, i=16; i-- ; ++lptr) /* Display character as ASCII */
            printf("%c", (*lptr<32) ? ' ' : *lptr);
        printf(" ");
        for (lptr=dptr, i=16; i-- ; ) /* Display character as hex */
            printf("%02X ", *lptr++);
        printf("\n");
        /* Move to next line */
    }
}

/*****
*   Function          : T R A C E _ M C B
**-----**
*   Task              : Displays list of MCBs.
*   Input              : None
*   Output             : None
*****/

void trace_mcb( void )
{
    static char fenv[] = {
        /* First environment string */
        'C', 'O', 'M', 'S', 'P', 'E', 'C', '='
    };

    MCBPtr cur_mcb;        /* Pointer to the first current MCB */
    BOOLEAN endit;         /* TRUE when the last MCB is found */
    BYTE    nr_mcb,        /* Number of MCBs currently processed */
           i;              /* Loop variable */
    FB      lptr;          /* Floating pointer to environment */

    endit = FALSE;         /* Continue looping */
    nr_mcb = 1;            /* Assign the first MCB the number 1 */
    cur_mcb = first_mcb(); /* Get pointer to the first MCB */
    do /* Process individual MCBs */
    {
        if ( cur_mcb->id_code == 'Z' ) /* Last MCB reached? */
            endit = TRUE; /* Yes */
        printf("MCB number      = %d\n", nr_mcb++);
        printf("MCB address      = %Fp\n", cur_mcb);
        printf("Memory address = %Np:0000\n", FP_SEG(cur_mcb)+1);
        printf("ID              = %c\n", cur_mcb->id_code);
        printf("PSP address     = %Fp\n", (FB) MK_FP(cur_mcb->psp, 0) );
        printf("Size           = %u paragraphs ( %lu bytes )\n",
            cur_mcb->spacing, (unsigned long) cur_mcb->spacing << 4);
        printf("Contents        = ");

        /*-- Handle MCB as an environment? -----*/

        for (i=0, lptr=(FB)cur_mcb+16; /* Compare 1st ENV string w/ FENV */
            ( i<sizeof fenv ) && ( *(lptr++) == fenv[i++] ) ; )
            ;
        if ( i == sizeof fenv ) /* String found? */
        {
            /* Yes --> Handle as an environment */
            printf("Environment\n");
            if ( _osmajor >= 3 ) /* DOS Version 3.0 or higher? */
            {
                /* Yes --> List program name */
                printf("Program name = ");
                for ( ; !(*(lptr++)==0 && *lptr==0) ; )
                    ; /* Search for last ENV string */
                if ( *(int far *) (lptr + 1) == 1 )

```

```

        {
            for ( lptr += 3; *lptr ; )
                printf( "%c", *(lptr++) );
        }
        else
            printf("Unknown");
        printf("\n");

/*-- Display environment string -----*/

printf("Environment string\n");
for (lptr=(FB) cur_mcb +16; *lptr ; ++lptr)
{
    printf(" ");
    for ( ; *lptr ; )
        printf( "%c", *(lptr++) );
    printf("\n");
}
else
{

/*- Handle it as a PSP? -----*/
/*-- (If INT 20 (Code=0xCD 0x20) starts the code) -----*/

if (*(unsigned far *) MK_FP( cur_mcb->psp, 0 )) == 0x20cd)
    printf("PSP (with program following)\n");
else
{
    printf("Unidentifiable as program or data\n");
    dump( (FB) cur_mcb + 16, 5);
}
}

printf("-----");
printf("----- Please press a key ----\n");
if ( !endit )
{
    cur_mcb = (MCBPtr)
        MK_FP( FP_SEG(cur_mcb) + cur_mcb->spacing + 1, 0 );
    getch();
}
while ( !endit );

/*****
**                               **
MAIN PROGRAM
**
*****/

void main( void )
{
    printf("\nMCBC (c) 1988, 92 by Michael Tischer\n\n");
    trace_mcb();
}

```