

Listing: DDPTC.C

```

/*****
/*
/*          D D P T C . C
/*
/*-----*/
/* Task      : Enables selective modification of single
/*            values in the disk drive parameter table.
/*-----*/
/* Author     : Michael Tischer
/* Developed on : 08/22/91
/* Last update  : 04/07/95
/*-----*/
/* Memory model : SMALL
/*-----*/
*****/

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

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

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

#ifdef MK_FP
/* Macro MK_FP already defined? */
#undef MK_FP
/* Yes --> delete macro */
#endif

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

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



```

/*****
/* hex_byte : Converts a hex string to a byte.
/* Input      : See below
/* Output     : Number
*****/

byte hex_byte( char *hex )          /* Hex string to be converted */
{
    if ( hex[ 1 ] == 0x58 )          /* Test for X then number 0x.. */
        hex += 2;                   /* Pointer to first numeral */
    return( ( D_Numeral( *hex ) < 4 ) | D_Numeral( hex[ 1 ] ) );
}

/*****
/* GetIntVec: Gets an interrupt vector.
/* Input      : NUMBER = Interrupt number
/* Output     : Interrupt vector
*****/

void far *GetIntVec( int Number )
{
    return( *( ( void far * far * ) MK_FP( 0, Number * 4 ) ) );
}

/*****
/* RAM_DDPT : Tests whether DDPT is in RAM or in ROM.
/* Input      : DDPT = FAR pointer to DDPT variable
/* Output     : TRUE, if DDPT is in RAM
/* Info       : The function writes a value in the DDPT, reads
/*              it out again and checks whether the value
/*              was written, meaning the DDPT is located in RAM.
*****/

```

```

int RAM_DDPT( DDPT_Typ far *DDPT )          /* Pointer to DDPT */
{
    byte buffer;          /* Memory for current value of the DDPT */
    int Flag;             /* Memory for return value */

    buffer = *DDPT[ 0 ];          /* Save value of DDPT */
    *DDPT[ 0 ] = buffer ^ 0xFF;   /* Invert value */
    Flag = ( *DDPT[ 0 ] == ( buffer ^ 0xFF ) );
    *DDPT[ 0 ] = buffer;          /* Restore old value */
    return( Flag );              /* Return value */
}

/*****
/* DisplayValues: Displays values of the DDPT */
/* Input      : DDPT = FAR pointer to DDPT variable */
/* Output     : None */
*****/

void DisplayValues( DDPT_Typ far *DDPT )      /* Pointer to DDPT */
{
    printf( "Step rate          (SR): 0x%02x\n\n",
        ( *DDPT )[ 0 ] >> 4 );
    printf( "Head unload time   (HU): 0x%02x\n",
        ( *DDPT )[ 0 ] & 0x0f );
    printf( "Head load time     (HL): 0x%02x\n",
        ( *DDPT )[ 1 ] >> 1 );
    printf( "Head settle time    (HS): 0x%02x\n\n", ( *DDPT )[ 9 ] );
    printf( "Motor postrun time   (MP): 0x%02x\n", ( *DDPT )[ 2 ] );
    printf( "Motor startup time   (MS): 0x%02x\n", ( *DDPT )[ 10 ] );
}

```

```

/*****
/* NewValues: Set new values of the DDPT.
/* Input      : See below
/* Output     : None
*****/

void NewValues( int NumCmd,          /* Number of commands */
                char *CmdFd[],       /* Field with commands */
                DDPT_Typ far *DDPT ) /* Pointer to DDPT */
{
    int i,j;                        /* Loop counter */
    char PCh[ 4 ],                 /* Parameter to be changed */
          CmdPar[ 8 ];              /* Command line parameter */
    byte NewV,                      /* New value to be set */
          AuxiVar;                  /* Auxiliary value to be saved */

    /*-- Loop: Execute all parameters -----*/

    for ( i = 1; i < NumCmd; i++ )
    {
        strcpy( CmdPar, CmdFd[ i ] );          /* Get parameters */
        j = 0;
        while ( CmdPar[ j ] != 0 )
            CmdPar[ j++ ] = upcase( CmdPar[ j ] );

        PCh[ 0 ] = CmdPar[ 0 ];                 /* Parameter to be set */
        PCh[ 1 ] = CmdPar[ 1 ];
        PCh[ 2 ] = 0;
        NewV = hex_byte( &CmdPar[ 3 ] );       /* Value to be set */
        if ( !strcmp( PCh, "SR" ) )             /* Step rate? */
        {
            NewV = NewV << 4;                   /* Value in upper nibble */

```

```

    AuxiValue = ( *DDPT )[ 0 ] & 0x0F;          /* Read lower nibble */
    ( *DDPT )[ 0 ] = NewV | AuxiValue;          /* Write value */
}
else if ( !strcmp( PCh, "HU" ) )                /* Head unload time? */
{
    NewV = NewV & 0x0F;                          /* Value in lower nibble */
    AuxiValue = ( *DDPT )[ 0 ] & 0xF0;          /* Read upper nibble */
    ( *DDPT )[ 0 ] = NewV | AuxiValue;          /* Write value */
}
else if ( !strcmp( PCh, "HL" ) )                /* Head load time? */
{
    ( *DDPT )[ 1 ] = NewV << 1;                /* Save value in bit 1 - 7 */
}
else if ( !strcmp( PCh, "HS" ) )                /* Head settle time? */
{
    ( *DDPT )[ 9 ] = NewV;                      /* Save value */
}
else if ( !strcmp( PCh, "MP" ) )                /* Motor postrun time */
{
    ( *DDPT )[ 2 ] = NewV;                      /* Save value */
}
else if ( !strcmp( PCh, "MS" ) )                /* Motor starting time */
{
    ( *DDPT )[ 10 ] = NewV;                    /* Save value */
}
}
}

```

```

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

```

```

void main( int argc, char *argv[] )
{
    DDPT_Typ far *DDPT;                      /* Pointer to the current DDPT */

    printf( "DDPTC - (c) 1992 by Michael Tischer\n" );
    printf( "Allows user defined changed to current DDPT\n" );

    DDPT = GetIntVec( 0x1E );                /* Get pointer to the DDPT */
}

```

```

if ( RAM_DDPT )                                /* DDPT is in RAM, can be changed? */
{
    if ( argc > 1 )                            /* Set values? */
    {
        NewValues( argc, argv, DDPT );        /* Set new values */
        printf( "\n\nNew DDPT contents:\n" );
        DisplayValues( DDPT );                /* Display new values of the DDPT */
        exit( 0 );
    }
}
else                                            /* DDPT is in ROM, changes not possible */
    printf( "Disk drive parameter table in ROM - cannot be changed\n" );
printf( "\nContents of DDPT:\n" );
DisplayValues( DDPT );                        /* Display old values of the DDPT */
}

```