

```

/*****
/*
/*----- D V I C -----*/
/* Task : Demonstrates direct access to video RAM. */
/*-----*/
/* Author : Michael Tischer */
/* Developed on : 10/01/88 */
/* Last update : 02/26/92 */
/*-----*/
/* (MICROSOFT C) */
/* Compilation : CL /AS /W0 DVIC.C */
/* Call : DVIC */
/*-----*/
/* (BORLAND TURBO C) */
/* Compilation : Use Run command (no project file needed) */
*****/

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

#include <dos.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <bios.h>

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

typedef unsigned char BYTE; /* Create a byte */
typedef struct velb far * VP; /* VP = FAR pointer to video RAM */
typedef BYTE BOOL; /* like BOOLEAN in Pascal */

/*== Structures =====*/

struct velb { /* Describes a screen position as 2 bytes */
    BYTE ASCCode, /* ASCII code */
    charattr; /* Corresponding attribute */
};

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

/*-- MK_FP creates a FAR pointer to an object -----*/
/*-- from a segment address and offset address -----*/

#ifndef MK_FP /* MK_FP still not defined? */
#define MK_FP(seg, ofs) ((void far *) ((unsigned long) (seg)<<16|(ofs)))
#endif

#define COLOR(VG, HG) ((VG << 3) + HG)

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

#define TRUE 1 /* Constants for working with BOOL */
#define FALSE 0

/*-- The following constants return pointers to BIOS -----*/
/*-- variables at segment address 0x40 -----*/

#define CRT_START ((unsigned far *) MK_FP(0x40, 0x4E))
#define ADDR_6845 ((unsigned far *) MK_FP(0x40, 0x63))

#define NORMAL 0x07 /* Define character attributes */
#define HIINT 0x0f /* for monochrome video card */
#define INVERSE 0x70
#define UNDERSCORED 0x01
#define BLINKING 0x80

#define BLACK 0x00 /* Color attributes for color card */
#define BLUE 0x01
#define GREEN 0x02
#define CYAN 0x03
#define RED 0x04
#define VIOLET 0x05
#define BROWN 0x06
#define LGHTGRAY 0x07
#define DARKGRAY 0x01
#define LGHTBLUE 0x09

```

```

#define LGHTGREEN      0x0A
#define LGHTCYAN       0x0B
#define LGHTRED        0x0C
#define LGHTVIOLET     0x0D
#define YELLOW         0x0E
#define WHITE          0x0F

/*== Global variables =====*/

VP vptr;                /* Pointer to first character in video RAM */

/*****
* Function              : D P R I N T
**-----**
* Task                  : Writes a string directly to video RAM.
*
* Input parameters : - COLUMN    = The display column
*                   - SCROW      = The display row
*                   - DCOLR      = Character color (attribute)
*                   - STROUT      = The string to be displayed
* Return values       : None
*****/

void dprint(BYTE column, BYTE scrow, BYTE dcolr, char * string)
{
    register VP lptr;          /* Floating pointer to video RAM */
    register BYTE i;          /* Points to number of characters */

    /*-- Set pointer to display position in video RAM -----*/
    lptr = (VP) ((BYTE far *) vptr + *CRT_START) + scrow * 80 + column;
    for (i=0 ; *string ; ++lptr, ++i) /* Execute string */
    {
        lptr->ASCCode = *(string++); /* Character in video RAM */
        lptr->charattr = dcolr;      /* Set character attribute */
    }
}

/*****
* Function              : I N I T _ D P R I N T
**-----**
* Task                  : Gets the segment address for DPrint.
* Input parameters : None
* Return values       : None
* Info               : Segment address is stored in VPTR.
*****/

void init_dprint( void )
{
    vptr = (VP) MK_FP( (*ADDR_6845 == 0x3B4) ? 0xB000 : 0xB800, 0 );
}

/*****
* Function              : C L S
**-----**
* Task                  : Clears screen with help from DPrint.
*
* Input parameters : CCOLR    = Character attribute
* Return values       : None
*****/

void cls( BYTE ccolr )
{
    static char blankline[] =
        "                                " \
        "                                ";
    register BYTE i;          /* Loop counter */

    for (i=0; i<24; ++i) /* Execute individual lines */
        dprint(0, i, ccolr, blankline); /* Display blank line */
}

/*****
* Function              : N O K E Y
**-----**

```

```

* Task : Checks for a keypress. *
* Input parameters : None *
* Return values : TRUE when a key is pressed, otherwise FALSE. *
*****/

BOOL nokey( void )

{
#ifdef __TURBOC__ /* Turbo C used for compiling? */
return( bioskey( 1 ) == 0 ); /* Yes --> Read keyboard using BIOS */
#else /* No --> Microsoft C used */
return( _bios_keybrd( _KEYBRD_READY ) == 0 ); /* Read using BIOS */
#endif
}

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

void main()
{
BYTE firstcol, /* Color for first square on screen */
dcolr, /* Current color */
column, /* Current screen position */
scrow;

init_dprint(); /* Get segment address of video RAM */
cls( COLOR(BLACK, GREEN) ); /* Clear the screen */
dprint(22, 0, WHITE, "DVIC - (c) 1988, 1992 by Michael Tischer");
firstcol = BLACK; /* Start with basic BLACK */
while( nokey() ) /* Repeat until user presses a key */
{
if (++firstcol > WHITE) /* Last color found? */
firstcol = BLUE; /* Yes --> Continue with BLUE */
dcolr = firstcol; /* Place first color on the screen */

/*-- Fill the screen with squares -----*/

for ( column=0; column < 80; column += 4)
for (scrow=1; scrow < 24; scrow += 2)
{
dprint( column, scrow, dcolr, "UUUU");
dprint( column, scrow+1, dcolr, "UUUU");
dcolr = ++dcolr & 15;
}
}
}

```