

C listing: EXTC.C

```

/*****
*
*                               E X T C . C
*
**-----**
* Demonstration of accessing extended memory with the BIOS-
* Function of Interrupt 15h taking into consideration RAM disks.
**-----**
* Author          : MICHAEL TISCHER
* Developed on    : 05/18/1989
* Lastest update on: 04/07/1995
**-----**
* memory model    : SMALL
**-----**
* Microsoft C     : The warning "Segment lost in conversation"
*                  can't be avoided.
*****/

/*-- Include-files-----*/

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

/*== Typedefs =====*/

typedef unsigned char BYTE;           /* we build it in one byte */
typedef unsigned int WORD;
typedef BYTE BOOL;                   /* BOOLEAN in Pascal */

#define TRUE ( 0 == 0 )
```

```

#define FALSE ( 0 == 1 )

/*-- Macros -----*/

#ifndef __TURBOC__
    #define random(x) rand()
    #define randomize() srand(1)
#endif

/*-- globale variables -----*/

int  RdLen;                                /* size of the RAM-Disks in KB */
BOOL ExtAvail;                             /* extended Memory available? */
long ExtStart; /* Start addresse of the extended memory as linear Adr. */
int  ExtLen;                                /* size of the extended memory in KByte */

/*****
*   ExtAdrConv : convert a single FAR-Pointer in one 32-Bit large
*               linear address in the form of one return word
*-----*/
*   Input   : Adr = to the converted pointer
*   Output  : the converted address
*****/

long ExtAdrConv( void far *Adr )
{
    return (((long) Adr >> 16) << 4 ) + (unsigned int) Adr;
}

/*****
*   ExtCopy : copy data between any two buffers within
*            the 16-MB size address range of th 80286/386/486.
*****/

```

```

**-----**
*   Input   : Start = address of start buffer as 32-Bit linear Adr.   *
*             Target = address of target buffer as 32-Bit linear Adr. *
*             Len    = Number of bytes ot copy                        *
*   Output   : none                                                  *
*   Info    : - The number of bytes to be copied must be an even number. *
*****/

```

```

void ExtCopy( long Start, long Target, WORD Len )

```

```

{
    /*-- Data structure for accessing the extended RAM -----*/

```

```

    typedef struct {
        WORD Length,           /* Segment descriptor */
        ADR Lo;                /* Length of the segments in byte */
        BYTE ADR Hi,           /* Bit 0 to 15 of the Segment adr. */
        ATTRIB;                /* Bit 16 to 23 of the Segment adr. */
        WORD Res;              /* Segment attribute */
        WORD Res;              /* reserved for 80386 */
    } SDES;

```

```

    typedef struct {
        SDES Dummy,           /* Global Descriptor Table */
        GDTS,
        Start,                 /* copy from ... */
        Target,                /* ... to */
        Code,
        Stack;
    } GDT;

```

```

#define LOWORD(x) ((unsigned int) (x))

```

```

#define HIBYTE(x) (*(BYTE *)&x+2)

```

```

GDT          GTab;                                /* Global Descriptor Table */
union REGS   Regs;                                /* Define Processor regs. Interrupt */
struct SREGS SRegs;                               /* Segment register */
long         Adr;                                  /* the conversion of the address */

memset( &GTab, 0, sizeof GTab );                  /* all fields to 0 */

/*-- Create segment descriptor start segments -----*/

GTab.Start.AdrLo   = LOWORD(Start);
GTab.Start.AdrHi   = HIBYTE(Start);
GTab.Start.Attribut = 0x92;
GTab.Start.Length  = Len;

/*-- Create segment descriptor of the second segments -----*/

GTab.Target.AdrLo   = LOWORD(Target);
GTab.Target.AdrHi   = HIBYTE(Target);
GTab.Target.Attribut = 0x92;
GTab.Target.Length  = Len;

/*-- Copy memory scope with help of Function 0x87 the cassette-----*/
/*-- Interrupts 0x15 -----*/

Regs.h.ah = 0x87;                                /* Function number for 'Memory copy' */
SRegs.es = (long) (void far *) &GTab >> 16;    /* address of the GDT */
Regs.x.si = (int) &GTab;                          /* to ES:SI */
Regs.x.cx = Len >> 1;                             /* Number of copied words to CX */
int86x( 0x15, &Regs, &Regs, &SRegs );           /* Call function */
if ( Regs.h.ah )                                   /* error? */
{
    /* Yes, the display error code */
    printf( "\nError in access time to extended-Ram (%d)\n", Regs.h.ah);
}

```

```

    exit( 1 );
}
}

/* End program with error code */

/*****
*   ExtRead : Read a definite number of bytes from the extended
*             Memory in the main memory.
*   ****
*   Input  :  ExtAdr = Source address in extended-RAM (linear address)
*             BuPtr  = Pointer to the Target buffer in main memory
*             Len    = Number of bytes ot copy
*   Output : none
*****/

void ExtRead( long  ExtAdr,  void far *BuPtr, WORD Len )
{
    ExtCopy( ExtAdr, ExtAdrConv( BuPtr ), Len );
}

/*****
*   ExtWrite : Write a specified number of bytes from main memory
*             to extended Memory.
*   ****
*   Input  :  BuPtr  = Pointer to the source buffer in main memory
*             ExtAdr = Target address in extended-RAM (linear address)
*             Len    = Number of bytes ot copy
*   Output : none
*****/

void ExtWrite( void far *BuPtr, long ExtAdr, WORD Len)
{
    ExtCopy( ExtAdrConv( BuPtr ), ExtAdr, Len );
}

```

```

}

/*****
*   ExtGetInfo : Determine the start address of extended RAM and its      *
*               size considering that there may be RAM disks of          *
*               Type VDISK                                              *
**-----**
*   Input      : none                                                  *
*   Output     : none                                                  *
*   Globals    : ExtAvail/W, ExtStart/W, ExtLen/W                      *
*****/

```

```
void ExtGetInfo( void )
```

```

{
    typedef struct {
        BYTE dummy1[3];
        char Name[5];
        BYTE dummy2[3];
        WORD BpS;
        BYTE dummy3[6];
        WORD Sectors;
        BYTE dummy4;
    } BOOT_SECTOR;

    static char VDiskName[5] = { 'V', 'D', 'I', 'S', 'K' };

    BOOT_SECTOR BootSec;
    union REGS Regs;

    /* takes alleged Boot Sector */
    /* Processor regs. for interrupt call */

    /*-- Determine the size of extended Memory and whether */
    /*-- extended memory is available */

```

```

Regs.h.ah = 0x88; /* Function nr.: "determine size of extended-RAM"*/
int86( 0x15, &Regs, &Regs );          /* call cassette interrupt */
if ( Regs.x.ax == 0 )
{
    /* no extended RAM present */
    ExtAvail = FALSE;
    ExtLen   = ExtStart = 0;
    return;                                /* return to caller */
}

ExtAvail = TRUE;                          /* extended Memory available */
ExtLen   = Regs.x.ax;                    /* extended RAM present, mark size */

/*-- Search RAM-Disks for Typ V6DISK -----*/

ExtStart = 0x1000001;                    /* start at 1 MB */
while ( TRUE )                          /* check loop */
{
    ExtRead( ExtStart, &BootSec, sizeof BootSec );
    if ( memcmp( BootSec.Name, VDiskName,
        sizeof VDiskName ) == 0 )        /* is Boot sector a RAM disk? */
        ExtStart += (long) BootSec.Sectors * BootSec.BpS;    /* Yes */
    else
        break;                          /* loop end */
}

/*-- Subtract the size of the RAM disks from free extended RAM---*/

ExtLen -= (int) ((ExtStart - 0x1000001) >> 10);
}

/*****
* CheckExt : examine the consistency of free extended RAM
*
*****/

```

```

**-----**
*   Input   : none                                     *
*   Output  : none                                     *
*****/

void CheckExt( void )
{
    long   AdrTest;                                     /* address of the test blocks */
    int    i, j;                                       /* loop counter */
    BYTE   WriteBuf[1024],                             /* test block */
           ReadBuf[1024];
    BOOL   Ferror = FALSE;                             /* pointer to memory error */

    randomize();                                       /* initialize random number generator*/
    AdrTest = ExtStart;
    for ( i = 1; i <= ExtLen; ++i, AdrTest += 1024 )
    {
        for ( j = 0; j < 1024; )                     /* run through memory in 1 KB-Blocks */
            WriteBuf[ j++ ] = random( 255 );          /* Fill block with random number */

        printf("\r%ld", AdrTest );                    /* address of the examined KB-Blocks */

        /*-- Write buffer WriteBuf, then return to ReadBuf to read      */

        ExtWrite( WriteBuf, AdrTest, 1024 );
        ExtRead( AdrTest, ReadBuf, 1024 );

        /*-- determine identity from WriteBuf and ReadBuf -----*/

        for ( j = 0; j < 1024; ++j )                  /* Fill block with random value */
            if ( WriteBuf[j] != ReadBuf[j] )          /* Buffer content identical? */
            {
                /* No, Error! */
            }
    }
}

```



```

        printf( "\n Error! Memory location %ld\n", AdrTest + j - 1);
        Ferror = TRUE;
    }
}

/* set */
printf( "\n\n" );
if ( !Ferror ) /* did an error occur? */
    printf( "All o.k.!\n" ); /* No */
}

/*****
*   M A I N   P R O G R A M
*****/

void main( void )
{
    printf ( "EXTC - (c) 1989, 92 by Michael Tischer\n\n");
    ExtGetInfo(); /*Determine availability and size of extended Memory */
    if ( ExtAvail ) /* is extended RAM available? */
    {
        /*Yes */
        RdLen = (int) ( (ExtStart - 0x1000001 ) >> 10 );
        if ( RdLen == 0 ) /* is RAM disks installed? */
            printf( "No RAM disks installld.\nThe free boundry.\n" );
        else /* Yes, RAM-Disks present */
            printf( "One or more RAM disks have reserved %d KB of extended " \
                "RAM.\nThe free extended RAM begins at %d KB beyond the " \
                "1 MB memory boundry.\n", RdLen, RdLen);
        printf( "The size of the free extended RAM is %d KB\n", ExtLen);
        printf( "\nThe extended RAM has also been examined for " \
            " consistency...\n\n" );
        CheckExt();
    }
}
else

```

```
    printf( "There is no extended RAM installed in this computer!\n" );  
}
```