

```

/*****
*
*                               M R C I C . C
*
*-----*
*      Task      : Show how to use an MRCI server to
*                  compress and decompress data
*-----*
*      Author     : MICHAEL TISCHER
*      developed on : 09/25/1993
*      last update  : 10/10/1994
*-----*
*      Compilers   : QuickC and VC
*      Storage model : as desired
*****/

#include "dos.h"
#include "stdio.h"

#pragma pack(1)                /* no Auto-Word-Alignment in structures */

/*-- Types and structures -----*/
typedef unsigned short WORD;
typedef int BOOL;
typedef unsigned char BYTE;

typedef struct {
    long ManufacturerID;        /* 4-byte ASCII code with
                                /* manufacturer's name
    WORD ManufacturerVersion;    /* version number of the MRC server
                                /* High byte is main version, Low byte
                                /* is sub-version
    WORD MrciVersion;           /* version number of the MRCI that is
                                /* supporting this server
    void far * EntryPoint;      /* point of entry for executing
                                /* server functions
    WORD Flags;                 /* server features
    WORD HardwareFlags;         /* server features in hardware
    WORD MaxBytes;              /* largest block the server
                                /* can process
} MRCINFO;

typedef struct {
    void far *SourcePtr;        /* calling server functions
                                /* pointer in source buffer
    WORD SourceLen;             /* size of the source buffer in bytes
    WORD Reserved1;
    void far *DestPtr;          /* pointer in destination buffer
    WORD DestLen;               /* size of the destination buffer
    WORD ChunkLen;              /* size of block for compressed data
    long IncDecomp;             /* status for incremental decompression
} MRCIREQUEST;

typedef struct { /* describes a buffer for compression/decompression */
    void far *address;          /* pointer to buffer
    WORD lenb;                  /* size of the buffer in bytes
} BUF;

/*-- Constants -----*/

#define TRUE (0==0)
#define FALSE (0==1)

/*-- Flags and command codes -----*/
#define STANDARD_COMPRESS 1 /* standard compression */
#define STANDARD_DECOMPRESS 2 /* standard decompression */
#define MAX_COMPRESS 8 /* maximal compression */
#define INC_DECOMPRESS 32 /* incremental decompression */

/*-- Type of MRCI client -----*/
#define TYP_APPLICATION 0 /* MRCI client is an application */
#define TYP_TSRDRIVER 1 /* MRCI client is a TSR or device driver */

/*-- Error codes -----*/
#define MRCI_OK 0 /* everything is o.k. */
#define MRCI_ERR_FUNKTION 1 /* wrong function code */
#define MRCI_ERR_BUSY 2 /* server is busy */
#define MRCI_ERR_OVERFLOW 3 /* destination buffer too small */
#define MRCI_ERR_NOCOMPRESS 4 /* data cannot be compressed */

```

```

/*-- Constants for demo program -----*/
#define TYP_APP      TYP_APPLICATION      /* this is not a TSR */
#define CHUNK        1                   /* compress to 1 byte if possible */

/*-- Global variables -----*/

MRCINFO mrci;                          /* receives MRCI Info block after */
                                      /* calling MrciQuery */

/*****
* MrciQuery : checks for the existence of an MRCI server in hardware
*             or software and provides information about this server
*-----*
* Input : MrciInfo = if function call is successful, contains
*               an MRCI-Info block with information about the
*               server
* Output : TRUE if Mrci-Server is installed, otherwise FALSE
*****/

int MrciQuery ( MRCINFO* pmrci )
{
    BOOL server;                        /* is there an MRCI server? */
    union REGS regs;
    struct SREGS segregs;

    server = FALSE;
    regs.x.ax = 0x4A12;                  /* MUX code for MRCI server */
    regs.x.cx = ( 'M' << 8 ) + 'R';      /* CX = 'MR' */
    regs.x.dx = ( 'C' << 8 ) + 'I';      /* DX = 'CI' */
    int86x( 0x2f, &regs, &regs, &segregs);
    if ( (regs.x.cx == ( 'I' << 8 ) + 'C') &&      /* CX == 'IC' */
        (regs.x.dx == ( 'R' << 8 ) + 'M') )      /* DX == 'RM' */
        server = TRUE;                      /* CX and DX o.k. ---> server found */
    else
    {
        /* no software server found; test hardware server */
        regs.x.ax = 0xB001;                /* call Int 1A, Function B001h */
        regs.x.cx = ( 'M' << 8 ) + 'R';      /* CX = 'MR' */
        regs.x.dx = ( 'C' << 8 ) + 'I';      /* DX = 'CI' */
        int86x( 0x1A, &regs, &regs, &segregs);
        if ( (regs.x.cx == ( 'I' << 8 ) + 'C') &&
            (regs.x.dx == ( 'R' << 8 ) + 'M') )
            server = TRUE;                  /* CX and DX o.k. ---> server found */
    }

    if (server) /* if server found, then display ES:DI in the Info block */
        * (MRCINFO far *) pmrci =
            * (MRCINFO far *) (((long) segregs.es << 16) + regs.x.di);
    return server;
}

/*****
* MrciCall : Calls one of the server functions
*-----*
* Input : OperationCode = command code (see constants)
*         SourceBuf      = source buffer descriptor
*         DestBuf        = destination buffer descriptor
* Output : error code of the MRCI server
* Info    : this function can only be called after MrciQuery has been
*           successfully called
* Globals : mrci
*****/

int MRCICall( WORD OperationCode, BUF *sourcebuf, BUF *destbuf)
{
    MRCIREQUEST mrcir;                  /* to construct an MRCI request block */
    void far * mrcrp;                  /* points to MRCIR */
    int err;                          /* error code returned */

    mrcir.SourcePtr = sourcebuf->address; /* register source buffer in */
    mrcir.SourceLen = sourcebuf->lenb;    /* request block */

    mrcir.DestPtr = destbuf->address;    /* register destination buffer in */
    mrcir.DestLen = destbuf->lenb;      /* request block */

    mrcir.ChunkLen = CHUNK;             /* obtain chunk size from constants */

```

```

mrcrp = &mrcir;                                /* note pointer to request block */

_asm
{
    push    bp                                /* save register */
    push    ds

/*-- Enter into a Windows Critical Section so that -----*/
/*-- multiple VMs can't call the MRCI server at the same time -----*/
/*-- under Windows in 486 Enhanced Mode -----*/

    push    ax                                /* Windows expects exactly this */
    mov     ax,8001h                          /* series of commands and no other */
    int     2ah
    pop     ax

    /*--- Prepare the call of the MRCI server -----*/
    mov     ax,OperationCode
    mov     cx,TYP_APP

    mov     dx,seg mrci    /* pointer to MRCI Info block following ES:BX */
    mov     es,dx
    mov     bx,offset mrci
    lds     si,mrcrp        /* set DS:SI to request block */
    call    dword ptr es:[bx+8]    /* call server */

    /*-- Leave Windows Critical Section -----*/
    push    ax                                /* the command sequence must be */
    mov     ax,8101h                          /* followed exactly here as well */
    int     2ah
    pop     ax

    pop     ds                                /* retrieve register */
    pop     bp

    mov     err,ax                            /* store returned error code */
}

destbuf->lenb = mrcir.DestLen;    /* return size of destination buffer */
return err;                       /* return error code as result of the function */
}

/*****
* Compress : compress a data block with the help of the MRCI server *
*-----*
* Input   : TypCompress = one of the two constants, STANDARD_COMPRESS *
*           or MAX_COMPRESS *
*           SourceBuf   = source buffer *
*           DestBuf     = destination buffer *
* Output  : error code of the MRCI server *
* Info    : this function can only be called after MrciQuery has been *
*           successfully called *
*****/

int Compress( WORD TypCompress, BUF *sourcebuf, BUF *destbuf )
{
    return MRCICall( TypCompress, sourcebuf, destbuf );
}

/*****
* Decompress : decompresses a data block using the MRCI server *
*-----*
* Input   : SourceBuf = source buffer *
*           DestBuf   = destination buffer *
* Output  : error code of the MRCI server *
* Info    : - This function can only be called after *
*           MrciQuery has been successfully called *
*           - The data block must be compressed by the MRCI server *
*****/

int Decompress( BUF *sourcebuf, BUF *destbuf )
{
    return MRCICall( STANDARD_DECOMPRESS, sourcebuf, destbuf );
}

/*****

```

```

* YesNo : checks flags and returns yes/no message *
*-----*
* Input : Status = Status in which the desired flag is situated *
* Flag = bit value of the flag *
* Output : Yes or No as a string, 4 characters long each *
*****/

char * YesNo( WORD Status, WORD Flag)
{
    if ( (Status & Flag) == Flag )
        return "Yes ";
    else
        return "No ";
}

/*****
* PrintMrciInfos: provides status information about the MRCI server *
*-----*
* Input : none *
* Output : none *
* Info : This function can only be called after *
* MrciQuery has been successfully called *
* Globals : mrci *
*****/

void PrintMrciInfos()
{
    printf( "Manufacturer : %c%c%c%c\n",
        (char) (mrci.ManufacturerID & 255),
        (char) (( mrci.ManufacturerID >> 8 ) & 255 ),
        (char) (( mrci.ManufacturerID >> 16 ) & 255 ),
        (char) ( mrci.ManufacturerID >> 24) );
    printf( "Manufacturer-Version : %d.%d\n",
        mrci.ManufacturerVersion >> 8,
        mrci.ManufacturerVersion & 255 );
    printf( "MRCI-Version : %d.%d\n\n",
        mrci.MrciVersion >> 8,
        mrci.MrciVersion & 255 );

    printf( "Features Software Hardware\n");
    printf( "-----\n");
    printf( "Standard Compression : %s %s\n",
        YesNo( mrci.Flags, STANDARD_COMPRESS ),
        YesNo( mrci.HardwareFlags, STANDARD_COMPRESS ) );
    printf( "Standard Decompression : %s %s\n",
        YesNo( mrci.Flags, STANDARD_DECOMPRESS ),
        YesNo( mrci.HardwareFlags, STANDARD_DECOMPRESS ) );
    printf( "Maximum Compression : %s %s\n",
        YesNo( mrci.Flags, MAX_COMPRESS ),
        YesNo( mrci.HardwareFlags, MAX_COMPRESS ) );
    printf( "Incremental Decompression : %s %s\n\n",
        YesNo( mrci.Flags, INC_DECOMPRESS ),
        YesNo( mrci.HardwareFlags, INC_DECOMPRESS ) );
};

/*-----*/
/*-- Main program -----*/
/*-----*/
#define SBUFLEN 12196 /* size of the data to be compressed */

BYTE sourcebuf[SBUFLEN]; /* source and destination buffers */
BYTE destbuf[SBUFLEN];

void main( int argc, char *argv[] )
{
    BUF sb, /* describe source and destination buffer */
    db;
    int err, /* error code */
    i; /* loop counter */

    printf( "MRCIC - (c) 1993, 94 by Michael Tischer\n\n");
    if (!MrciQuery( &mrci ))
    {
        printf("MRCI not installed\n");
        return;
    }
}

```

```

/*-- MRCI is installed, first output info about MRCI server -----*/

printf( "MRCI installed\n");
PrintMrciInfos();

/*-- fill buffer with data, compress and decompress -----*/
sb.address = &sourcebuf;          /* build descriptor for source and */
sb.lenb = SBUFLEN;                /* destination buffers */
db.address = &destbuf;
db.lenb = SBUFLEN;

/*-- Fill source buffer with dummy data -----*/
for (i = 0; i < SBUFLEN; sourcebuf[i++] = 1 );

printf("Compress\n");
printf("Size before: %d bytes\n", SBUFLEN);
err = Compress( STANDARD_COMPRESS, &sb, &db );
printf ( "Error code = %d\n", err );
printf("Size after: %d bytes\n\n", db.lenb);

printf("Decompress\n");
printf("Size before: %d bytes\n", db.lenb );
err = Decompress( &db, &sb );
printf ( "Error code = %d\n", err );
printf ( "Size after: %d bytes\n", sb.lenb);
}

```