```
#--------------------------------------------------------------------
# Makefile para geracao de aplicacoes E286
# Autor: Jose Rufino
# Data:  OUT 94
#--------------------------------------------------------------------

.SUFFIXES : .asm .c .obj

MFILE  = makefile

IDIR   = c:\e286\include
KDIR   = c:\e286\kernel
KTASK  = c:\e286\kernel\task1.exe
KTMAP  = c:\e286\kernel\task1.map
IOLIB  = c:\e286\kernel\iopc.lib
EXLIB  = c:\e286\kernel\exints.lib

MASM   = masm
AFLAGS = /Z /ML /P
CC     = cl
CFLAGS = /c /Gs /G2 /Awsn /J /Od
LINK   = link
LFLAGS = /M /DOSSEG /NOE
LIBS   = $(IOLIB)+$(EXLIB)

OBJS   = aplmain.obj\
         comando.obj\
         monitor.obj\
         tfvideo.obj\
         cqfunct.obj\
         bqfinit.obj\
         bqmessg.obj\
         bqports.obj\
         bqexmem.obj

LOBJS  = $(KDIR)\tbegin.obj\
         $(KDIR)\tstart.obj\
         $(OBJS)\
         $(KDIR)\tend.obj

# Geracao de objectos a partir de ficheiros fonte
#
.asm.obj:
        $(MASM) $(AFLAGS) $< ;
.c.obj:
        $(CC) $(CFLAGS) /I$(IDIR) $<

# Geracao da aplicacao
apl:    $(OBJS)
    nmake mkexec

mkexec:
    $(LINK) $(LFLAGS) @<<linkobj
$(LOBJS: =+^
),
$(KTASK),
$(KTMAP),
$(LIBS);
<<NOKEEP

# Execucao de utilidades
#
purge:
    -rm *.bak

clean:
    -rm *.bak
    -rm *.obj

kclean:
    -rm $(KTASK) $(KTMAP)
```



```
# Dependencias
aplmain.obj:        aplmain.c   tfvideo.h
comando.obj:        comando.c
cqfunct.obj:        cqfunct.c   dcxapi.h
bqfinit.obj:        bqfinit.c   dcxapi.h
bqmessg.obj:        bqmessg.c   dcxapi.h
bqports.obj:        bqports.c   dcxapi.h
bqexmem.obj:        bqexmem.c   dcxapi.h
monitor.obj:        monitor.c
tfvideo.obj:        tfvideo.c   tfvideo.h
```

```
/***************************************************************************
| File: aplmain.c  -  Application startup
|
| Autor:
| Data:
***************************************************************************/
#include <stdio.h>
#include "e286.h"
#include "colors.h"
#include "assert.h"
#include "tfvideo.h"

/*------------------------------------------------------------------------+
| Headers of external functions
+-------------------------------------------------------------------------*/
extern  void    monitor         ();


/*------------------------------------------------------------------------+
| Function: aplInit
+-------------------------------------------------------------------------*/

void    aplInit ()
{
    static word status;

    bq_set_port     ( 0x208, 0x20A, 0x20C, &status);
    bq_set_scantime ( 1, &status);

    dbgPrint        ( "Status %d\n", cq_dcm_status_check() );
}

/*------------------------------------------------------------------------+
| Function: maintask        (main task called upon E286 startup)
+-------------------------------------------------------------------------*/
main ()
{
    /* clean up full screen and install debug window                */
    clearFullScreen ();
    dbgScreenOpen ( 8, 42, 13, 79, B_BLACK | F_YELLOW );

    /* init synoptic and application entities  ( keep this order )  */
    aplInit ();

    /* Main task assumes monitor functions forever                  */
    monitor ();
}
```



```
/***************************************************************************
| File: bqexmem.c
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
***************************************************************************/
#include <stddef.h>
#include "e286.h"
#include "assert.h"
#include "dcxapi.h"

/*------------------------------------------------------------------------+
| Function: bq_read_ext_mem
+-------------------------------------------------------------------------*/
bq_read_ext_mem (byte sid, word fbyte, word lbyte, byte *dat, word *status)
{
            word    i;

#pragma pack(1)
    static  struct  upload_d {
                    DcxHdr  hdr;
                    word    memptr;
                    byte    buf[DCX_BUF-sizeof(word)];
            } msg;
#pragma pack()

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+sizeof(word)+(lbyte-fbyte)+1;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_UPLOAD;
    msg.memptr      = ((fbyte << 8) & 0xFF00) | ((fbyte >> 8) & 0x00FF);

    cq_dcm_transmit     ( (DcxMsg*) &msg);
    bq_receive_message  ( &msg, FOREVER, status);

    for (i=0; i < (lbyte-fbyte)+1; i++)
        dat[i] = msg.buf[i];

}

/*------------------------------------------------------------------------+
| Function: bq_write_ext_mem
+-------------------------------------------------------------------------*/
bq_write_ext_mem (byte sid,word fb,word lb,word xszi,byte *dat,word *status)
{

#pragma pack(1)
    static  struct  upload_d {
                    DcxHdr  hdr;
                    word    memptr;
                    byte    buf[DCX_BUF-sizeof(word)];
            } msg;
#pragma pack()
            word    i;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+sizeof(word)+(lb-fb)+1;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_DNLOAD;
    msg.memptr      = ((fb << 8) & 0xFF00) | ((fb >> 8) & 0x00FF);

    for (i=0; i < (lb-fb+1); i++)
        msg.buf[i]  = dat[i];

    cq_dcm_transmit     ( (DcxMsg*) &msg);
    bq_receive_message  ( &msg, FOREVER, status);
}
```

```
/***************************************************************************
| File: bqfinit.c
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
***************************************************************************/
#include <stddef.h>
#include "e286.h"
#include "assert.h"
#include "dcxapi.h"

/*-------------------------------------------------------------------------+
| Declaration of external variables
+-------------------------------------------------------------------------*/
extern  PcxHdr  pcx;

/*-------------------------------------------------------------------------+
| Function: bq_set_port
+-------------------------------------------------------------------------*/
bq_set_port ( word data_port, word cmd_port, word status_port, word *status)
{
    pcx.data_port   = data_port;
    pcx.cmd_port    = cmd_port;
    pcx.status_port = status_port;
}

/*-------------------------------------------------------------------------+
| Function: bq_set_scantime                 ( E286 specific function )
+-------------------------------------------------------------------------*/
bq_set_scantime ( word scantime, word *status)
{
    pcx.scan_time   = scantime;
}

/*-------------------------------------------------------------------------+
| Function: bq_reset_device
+-------------------------------------------------------------------------*/
bq_reset_device ( byte sid, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+1;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_RESET_NODE;

    cq_dcm_transmit ( &msg);
}
```



```
/***************************************************************************
| File: bqmessg.c
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
***************************************************************************/
#include <stddef.h>
#include "e286.h"
#include "assert.h"
#include "dcxapi.h"

/*-------------------------------------------------------------------------+
| Function:
+-------------------------------------------------------------------------*/
byte bq_receive_message ( DcxMsg *msgp, byte wait, word *status)
{
    if (wait) {
        cq_dcm_receive ( msgp );
        *status = E_OK;
    } else
        if ( cq_dcm_status_check() ) {
            cq_dcm_receive ( msgp );
            *status = E_OK;
        } else
            *status = E_MSG_NOT_RECEIVED;
}
```

```
/*****************************************************************************
| File: bqports.c
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
*****************************************************************************/
#include <stddef.h>
#include "e286.h"
#include "assert.h"
#include "dcxapi.h"

/*--------------------------------------------------------------------------+
| Function: bq_read_io
+--------------------------------------------------------------------------*/
byte bq_read_io ( byte sid, byte port, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_READ;
    msg.buf[0]      = port;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
    return (msg.buf[1]);
}

/*--------------------------------------------------------------------------+
| Function: bq_write_io
+--------------------------------------------------------------------------*/
bq_write_io ( byte sid, byte port, byte value, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_WRITE;
    msg.buf[0]      = port;
    msg.buf[1]      = value;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
}

/*--------------------------------------------------------------------------+
| Function: bq_update_io
+--------------------------------------------------------------------------*/
byte bq_update_io ( byte sid, byte port, byte value, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_UPDATE;
    msg.buf[0]      = port;
    msg.buf[1]      = value;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
    return (msg.buf[1]);
}
```



```
/*--------------------------------------------------------------------------+
| Function: bq_or_io
+--------------------------------------------------------------------------*/
byte bq_or_io ( byte sid, byte port, byte value, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_OR;
    msg.buf[0]      = port;
    msg.buf[1]      = value;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
    return (msg.buf[1]);
}

/*--------------------------------------------------------------------------+
| Function: bq_and_io
+--------------------------------------------------------------------------*/
byte bq_and_io ( byte sid, byte port, byte value, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_AND;
    msg.buf[0]      = port;
    msg.buf[1]      = value;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
    return (msg.buf[1]);
}

/*--------------------------------------------------------------------------+
| Function: bq_xor_io
+--------------------------------------------------------------------------*/
byte bq_xor_io ( byte sid, byte port, byte value, word *status)
{
    static  DcxMsg  msg;

    msg.hdr.link    = 0;
    msg.hdr.length  = sizeof(DcxHdr)+2;
    msg.hdr.flags   = 0x40;
    msg.hdr.node    = sid;
    msg.hdr.tasks   = 0;
    msg.hdr.cmdresp = RAC_EXT_IO_XOR;
    msg.buf[0]      = port;
    msg.buf[1]      = value;

    cq_dcm_transmit     ( &msg);
    bq_receive_message  ( &msg, FOREVER, status);
    return (msg.buf[1]);
}
```

```
/****************************************************************************
| File: comando.c  -  Implements monitor commands.
|
| Autor:
| Data:
****************************************************************************/
#include <stddef.h>
#include <string.h>
#include "e286.h"
#include "assert.h"

/*--------------------------------------------------------------------------+
| Variable definition
+--------------------------------------------------------------------------*/


/*--------------------------------------------------------------------------+
| Function: cmd_cls - clears monitor screen         (called from monitor)
+--------------------------------------------------------------------------*/
void cmd_cls (int argc, char **argv)
{
    clearscreen ();
}

/*--------------------------------------------------------------------------+
| Function: cmd_dosboot - PC dependent hardware reboot (called from monitor)
+--------------------------------------------------------------------------*/
void cmd_dosboot (int argc, char **argv)
{
    reboot ();
}

/*--------------------------------------------------------------------------+
| Function: cmd_reset
+--------------------------------------------------------------------------*/
void cmd_reset (int argc, char** argv)
{
    static  word err;

    bq_reset_device ( atoi(argv[1]), &err );
}

/*--------------------------------------------------------------------------+
| Function: cmd_rio
+--------------------------------------------------------------------------*/
void cmd_rio (int argc, char** argv)
{
    static  word err;
            byte retval;

    retval = bq_read_io ( atoi(argv[1]), atoi(argv[2]), &err );
    if (err == E_OK)
        printf ("\n     value=%d", retval);
    else
        printf ("\n     erro=%d", err);
}

/*--------------------------------------------------------------------------+
| Function: cmd_wio
+--------------------------------------------------------------------------*/
void cmd_wio (int argc, char** argv)
{
    static  word err;
            byte retval;

    retval = bq_write_io ( atoi(argv[1]), atoi(argv[2]), atoi(argv[3]), &err );
    if (err == E_OK)
        printf ("\n     EOK");
    else
        printf ("\n     erro=%d", err);
}
```



```
/*--------------------------------------------------------------------------+
| Function: cmd_rxm
+--------------------------------------------------------------------------*/
void cmd_rxm (int argc, char** argv)
{
    static  word    err;
    static  byte    dat[13];
            int     i;

bq_read_ext_mem ( atoi(argv[1]), atoi(argv[2]), atoi(argv[3]), dat, &err);
    if (err == E_OK) {
        printf ("\n  read= ");
        for (i=0; i<(atoi(argv[3])-atoi(argv[2])+1);i++)
            printf ("%02X ", dat[i]);
    } else
        printf ("\n     erro=%d", err);
}


/*--------------------------------------------------------------------------+
| Function: cmd_wxm
+--------------------------------------------------------------------------*/
void cmd_wxm (int argc, char** argv)
{
    static  word    err;
    static  byte    dat;

dat = atoi(argv[4]);
dbgPrint ("\nData = %x %d\n", dat, dat);
bq_write_ext_mem ( atoi(argv[1]), atoi(argv[2]), atoi(argv[3]), 1, &dat, &err);
    if (err == E_OK)
        printf ("\n     E$OK");
    else
        printf ("\n     erro=%d", err);
}
```

```
/***************************************************************************
| File: cqfunct.c
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
***************************************************************************/
#include <stddef.h>
#include "e286.h"
#include "assert.h"
#include "dcxapi.h"

/*-------------------------------------------------------------------------+
| Variable definition
+--------------------------------------------------------------------------*/
PcxHdr  pcx;

/*-------------------------------------------------------------------------+
| Macro: CqWait
+--------------------------------------------------------------------------*/
#define     CqWait(mask)    if (1) { \
                while ( inb (pcx.status_port) & mask ) { \
                    rq_sleep ( pcx.scan_time, &err);    rq_error (err); \
                } \
            } else

/*-------------------------------------------------------------------------+
| Function: cq_dcm_transmit
+--------------------------------------------------------------------------*/
void cq_dcm_transmit ( DcxMsg *msgp)
{
    static  word    err;
            word    count, i;

    for (count=msgp->hdr.length-sizeof(word), i=0; count > 0; count--, i++) {
        CqWait   (UBI_TFNF);
        outb (pcx.data_port, ((char*)&(msgp->hdr.length))[i]);
    }
    CqWait   (UBI_TFNF);
    outb (pcx.cmd_port, UBI_ECMD);
}

/*-------------------------------------------------------------------------+
| Function: cq_dcm_receive
+--------------------------------------------------------------------------*/
void cq_dcm_receive ( DcxMsg *msgp)
{
    static  word    err;
            word    i;

    CqWait   (UBI_RFNE);
    for ( i=0; ; i++) {
        if ( (inb (pcx.status_port) & UBI_RCMD) == UBI_ECMD ) {
            inb (pcx.cmd_port);
            return;
        } else {
            ((char*)&(msgp->hdr.length))[i] = inb (pcx.data_port);
            CqWait   (UBI_RFNE);
        }
    }
}

/*-------------------------------------------------------------------------+
| Function:
+--------------------------------------------------------------------------*/
byte cq_dcm_status_check ( )
{
    if ( inb (pcx.status_port) & UBI_RFNE )
        return  0x00;
    else
        return  0xFF;
}
```



```
/***************************************************************************
| File: dcxapi.h                DCX-51 Application Interface
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Mar 95
***************************************************************************/
#ifndef _DCXAPI_H
#define _DCXAPI_H

/*-------------------------------------------------------------------------+
| General definitions
+--------------------------------------------------------------------------*/
#define     DCX_BUF     13          /* DCX-51 Message data size           */
#define     UBI_TFNF    0x01        /* PCX interface - Tx FIFO Not Full   */
#define     UBI_RFNE    0x10        /* PCX interface - Rx FIFO Not Empty  */
#define     UBI_RCMD    0x20        /* PCX interface - Rx Command         */
#define     UBI_ECMD    0x00        /* PCX interface - Command byte value */

/*-------------------------------------------------------------------------+
| Definition of error constants
+--------------------------------------------------------------------------*/
#define E_DCX_CONFIGURED        0x0008
#define E_SUPPORT               0x0023
#define E_PATHNAME_SYNTAX       0x003E
#define E_CHECKSUM              0x0064
#define E_EOF                   0x0065
#define E_REC_FORMAT            0x0069
#define E_REC_TYPE              0x006B
#define E_CONTROL_C             0x0089
#define E_SPT                   0x0510
#define E_INV                   0x0511
#define E_KEY                   0x0512
#define E_DCX_EXIST             0x0580
#define E_MAX_TASKS             0x0581
#define E_REGS                  0x0582
#define E_FID                   0x0583
#define E_BUFFERS               0x0584
#define E_PRIORITY              0x0585
#define E_ITD                   0x0586
#define E_UMEM                  0x0587
#define E_TIME_OUT              0x0590
#define E_PROTOCOL_ERROR        0x0591
#define E_NO_DESTINATION_DEVICE 0x0593
#define E_ACCESS_PROTECTED      0x0594
#define E_RAC_PROTECTED         0x0595
#define E_UNKNOWN_RAC_COMMAND   0x0596
#define E_SBX_NOT_SET           0x05A0
#define E_PORT_NOT_SET          0x05A0
#define E_PDS_RESTRICTED        0x05A1
#define E_FILE_STACK            0x05A2
#define E_CLOGGED               0x05A3
#define E_MSG_NOT_RECEIVED      0x05A4
#define E_NO_HELP_AVAILABLE     0x05A5
#define E_XMIT_BUF_SIZE         0x05A6
#define E_OS_PROTECTION         0x05A7
#define E_COMMAND_ENTRY_ABORTED 0x05A8
#define E_AMBIGUOUS_SYMBOL      0x05A9
#define E_UNKNOWN_SYMBOL        0x05AA
#define E_CODE_WRITE_FAULT      0x05AB
#define E_HISM_DIFF_SYM         0x05AC
#define E_TEST_FAIL             0x05F1
#define E_TEST_NOT_FOUND        0x05F2
#define E_NOT_IMPLEMENTED       0x05FE
#define E_MISC_DCM_ERROR        0x05FF


/*-------------------------------------------------------------------------+
| Remote Access and Control Codes
+--------------------------------------------------------------------------*/
#define RAC_RESET_NODE          0x00
#define RAC_CREATE_TASK         0x01
```

```
#define RAC_DELETE_TASK         0x02
#define RAC_GET_FUNCTION_ID     0x03
#define RAC_PROTECT             0x04
#define RAC_EXT_IO_READ         0x05
#define RAC_EXT_IO_WRITE        0x06
#define RAC_EXT_IO_UPDATE       0x07
#define RAC_EXT_UPLOAD          0x08
#define RAC_EXT_DNLOAD          0x09
#define RAC_EXT_IO_OR           0x0A
#define RAC_EXT_IO_AND          0x0B
#define RAC_EXT_IO_XOR          0x0C
#define RAC_INT_READ            0x0D
#define RAC_INT_WRITE           0x0E
#define RAC_NODE_INFO           0x0F
#define RAC_NODE_OFFLINE        0x10
#define RAC_UPLOAD_CODE         0x11
#define RAC_DNLOAD_CODE         0x12
/*-----------------------------------------------------------------------+
| Definition of DCX-51 Message Format
+------------------------------------------------------------------------*/
#pragma pack (1)
typedef     struct  dcx_d {
                    word    link;
                    byte    length;
                    byte    flags;
                    byte    node;
                    byte    tasks;
                    byte    cmdresp;
            } DcxHdr;

typedef     struct  dcxmsg_d {
                    DcxHdr  hdr;
                    byte    buf[DCX_BUF];
            } DcxMsg;
#pragma pack ()

typedef     struct  pcx_d  {
                    byte    in_use_flag;
                    byte    cpu_type;
                    word    data_port;
                    word    cmd_port;
                    word    status_port;
                    word    scan_time;
            } PcxHdr;

#endif

/*-----------------------------------------------------------------------+
| Definition of DCM Functions Prototypes
+------------------------------------------------------------------------*/
void    cq_dcm_transmit     (DcxMsg*);
void    cq_dcm_receive      (DcxMsg*);
byte    cq_dcm_status_check ();
```



```
/**************************************************************************
| File: monitor.c
|
| Autor: Jose Rufino (IST/INESC), from an original by Leendert Van Doorn
| Data:  Jan 95
**************************************************************************/
#include <stddef.h>
#include <string.h>
#include "e286.h"
#include "colors.h"
#include "tfvideo.h"

/*-----------------------------------------------------------------------+
| Headers of command functions
+------------------------------------------------------------------------*/
        void    cmd_sos     (int, char** );
extern  void    cmd_cls     (int, char** );
extern  void    cmd_dosboot (int, char** );
extern  void    cmd_reset    (int, char** );
extern  void    cmd_rio     (int, char** );
extern  void    cmd_wio     (int, char** );
extern  void    cmd_rxm     (int, char** );
extern  void    cmd_wxm     (int, char** );

/*-----------------------------------------------------------------------+
| Monitor window definition
+------------------------------------------------------------------------*/
#define     TCOLOR_MON      (B_BLUE    | F_YELLOW)
#define     WCOLOR_MON      (B_CYAN    | F_BRIGHTWHITE)
static  Twindow const monwin =
{13,38,24,79, TCOLOR_MON, WCOLOR_MON, "                Monitor"};

/*-----------------------------------------------------------------------+
| Variable and constants definition
+------------------------------------------------------------------------*/
const   char    TitleMsg[] = "\n BitBus Test Monitor\n";
const   char    InvalMsg[] = "\nInvalid command!";

struct  command_d {
        void  (*cmd_fnct)();
        char*   cmd_name;
        char*   cmd_help;
} const commands[] = {
    cmd_sos,     "sos","                     help",
    cmd_cls,     "cls","                     clear screen",
    cmd_dosboot, "dosboot","                 hard reboot",
    cmd_reset,   "reset","<nid>              reset node",
    cmd_rio,     "rio","<nid><port>          read  I/O port",
    cmd_wio,     "wio","<nid><port><val>     write I/O port",
    cmd_rxm,     "rxm","<nid><bad><ead>      read  ext mem",
    cmd_wxm,     "wxm","<nid><bad><ead><val> write ext mem"
};

#define     NCOMMANDS       (sizeof(commands)/sizeof(struct command_d))
#define     ARGVECSIZE      6
#define     MAX_LINE        50

/*-----------------------------------------------------------------------+
| Function: monitor        (called from main_task)
+------------------------------------------------------------------------*/
monitor ()
{
    static  char    *argv[ARGVECSIZE+1], *p;
            int     argc, i;

    createWindow ( (Twindow*) &monwin );
    setTitledWindowScreen ((Twindow*)&monwin, GetTwinScreenHeight(&monwin));
    printf("%s Type sos for help\n", TitleMsg);
    for ( ; ; ) {
        printf("\nCmd> ");
/* Reading and parsing command line  -------------------------------------*/
```

```
        if ((argc = getline(argv, ARGVECSIZE)) > 0) {
            for ( p=argv[0]; *p != '\0'; *p++=tolower(*p));
            for ( i = 0; i < NCOMMANDS; i++)
                if (strcmp(argv[0], commands[i].cmd_name) == 0)
                    break;
/* Executing commands ---------------------------------------------------*/
        if (i < NCOMMANDS)
            commands[i].cmd_fnct (argc, argv);
        else
            printf("%s", InvalMsg);

        } // if getline
    } // forever
}

/*----------------------------------------------------------------------+
| Function: getline        (called from monitor)
+----------------------------------------------------------------------*/
int getline (char** argv, int argvsize)
{
    static  char    line[MAX_LINE];
            char    *p, ch;
            int      argc;

/* Read command line, with some simple editing features  ------------------*/
    p = line;
    while ((ch = waitch(0)) != '\r' && ch != '\n' && (p-line) < sizeof(line)){
        if (ch == '\b') {                        /* rubout editing          */
            if (p > line) {
                printf(" \b");
                p--;
            } else
                printf(" ");
        } else
            *p++ = ch;                           /* add character to line   */
    } // while getch
    *p = '\0';

/* Break command line into an o.s. like argument vector,
   i.e. compliant with the (int argc, char **argv) specification ----------*/

    for (argc=0,p=line; (*line != '\0') && (argc < argvsize); p=NULL,argc++) {
        p = strtok ( p, " \t");
        argv[argc] = p;
        if (p == NULL) return argc;
    }
    argv[argc] = p;
    return argc;
}

/*----------------------------------------------------------------------+
| Function: cmd_sos - provides a rudimentary help
+----------------------------------------------------------------------*/
void cmd_sos (int argc, char **argv)
{
    int     i;

    clearscreen ();
    printf("%s", TitleMsg);
    for (i = 0; i < NCOMMANDS; i++)
        printf("%s %s\n", commands[i].cmd_name, commands[i].cmd_help);
}
```



```
/***************************************************************************
| File: tfvideo.h   -   Utilities for screen management
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Out 94
***************************************************************************/
#include "e286.h"
#include "colors.h"
#include "tfvideo.h"

/*----------------------------------------------------------------------+
| Function: clearFullScreen
+----------------------------------------------------------------------*/
void clearFullScreen ( )
{
    setwindow     ( 0, 0, 24, 79);
    setvideocolor ( B_BLACK | F_BLACK );
    clearscreen   ( );
}

/*----------------------------------------------------------------------+
| Function: createWindow
+----------------------------------------------------------------------*/
void createWindow ( Twindow* twp )
{
    if (twp->title == '\0')
        createUntitledWindow ( twp );
    else
        createTitledWindow   ( twp );
}

/*----------------------------------------------------------------------+
| Function: createUntitledWindow
+----------------------------------------------------------------------*/
void createUntitledWindow ( Twindow* twp )
{
    setwindow     ( twp->linei, twp->colmi, twp->linef, twp->colmf );
    setvideocolor ( twp->wcolor );
    clearscreen   ( );
    setvideocursor( twp->linei, twp->colmi);
}

/*----------------------------------------------------------------------+
| Function: createTitledWindow
+----------------------------------------------------------------------*/
void createTitledWindow ( Twindow* twp )
{
    setwindow     ( twp->linei, twp->colmi, twp->linei, twp->colmf );
    setvideocolor ( twp->tcolor );
    clearscreen   ( );
    setvideocursor( twp->linei, twp->colmi);
    printf        ("%s", twp->title);

    setwindow     ( twp->linei+1, twp->colmi, twp->linef, twp->colmf );
    setvideocolor ( twp->wcolor );
    clearscreen   ( );
    setvideocursor( twp->linei+1, twp->colmi );
}

/*----------------------------------------------------------------------+
| Function: setTitledWindowScreen
+----------------------------------------------------------------------*/
void setTitledWindowScreen ( Twindow* twp , int line)
{
    setwindow     ( twp->linei+1, twp->colmi, twp->linef, twp->colmf );
    setvideocolor ( twp->wcolor );
    setvideocursor( twp->linei+line, twp->colmi );
}
```

```
/*****************************************************************************
| File: tfvideo.h   -   Utilities for screen management
|
| Autor: Jose Rufino (IST/INESC)
| Data:  Out 94
****************************************************************************/
#ifndef _TFVIDEO_H
#define _TFVIDEO_H

typedef struct twin_d {
                int     linei;
                int     colmi;
                int     linef;
                int     colmf;
                int     tcolor;
                int     wcolor;
                char*   title;
} Twindow;

#define GetTwinScreenHeight(p) (((Twindow*)p)->linef-((Twindow*)p)->linei-1)
#define GetTwinScreenWidth(p)  (((Twindow*)p)->colmf-((Twindow*)p)->colmi)

extern  void    clearFullScreen         ( void );
extern  void    createWindow            ( Twindow* );
extern  void    createTitledWindow      ( Twindow* );
extern  void    createUntitledWindow    ( Twindow* );
extern  void    setTitledWindowScreen   ( Twindow*, int );

#endif
```