UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
DEEC

Cama de Testes Para Módulos e Componentes Electrónicos

(*Relatório de Estágio* PRODEP)

José Carlos Lobinho Gomes

Porto, 05 de Novembro de 1993

CAMA DE TESTES PARA MÓDULOS E COMPONENTES ELECTRÓNICOS

José Carlos Lobinho Gomes

Porto, *Outubro* de **1993**

Original que Relatório de aval 1.38

Porto, Dezembro de 1993

PRODEP

MEDIDA 4.3 - Estágios Profissionais para Bacharelatos, Licenciaturas e Pós-Graduação

PARECER

Assunto: Estágios de

(1.38) Alberto Xavier Pires Borges da Cunha
(1.42) José Carlos Lobinho Gomes
(1.44) Manuel Fernando dos Santos Silva
(1.48) Pedro Manuel Gonçalves Campelos de Sousa Lobo.

Na qualidade de supervisores dos estágios acima mencionados e em face do trabalho desenvolvido e relatado pelos estagiários pode concluir-se que:

1- Os estagiários manifestaram entusiasmo e dedicação na execução do trabalho intitulado:

" Projecto de uma cama de testes para componentes e módulos electrónicos incluindo a concepção da interface (entradas e saídas) e do sistema operativo".

O trabalho consistiu no projecto e teste em montagem experimental de vários módulos cujas responsabilidades foram distribuidas:

Software para interface com o utilizador em PC - Manuel Fernando dos Santos Silva; "device driver" para interface software/hardware - José Carlos Lobinho Gomes; hardware de teste com microcontrolador, conversão e software operativo - Manuel Fernando dos Santos Silva e Pedro Manuel Gonçalves Campelos de Sousa Lobo.

Nesta execução revelaram apreciáveis qualidades de trabalho que mais desenvolveram. Do trabalho resultou um protótipo do hardware em "breadboard" e protótipos dos módulos de software.

2- No cumprimento dos objectivos propostos, os estagiários revelaram bons conhecimentos científicos e técnicos, capacidade de aplicação desses conhecimentos e espírito de iniciativa. A metodologia seguida consistiu no estudo da proposta base, desenvolvimento por análise das especificações do sistema objecto, projecto de soluções tecnicamente adequadas eventualmente inovadoras, implementação de protótipos e validação do projecto.

pelo que somos do parecer que foram cumpridos os objectivos que o estágio se propunha atingir.

Prof. Doutor António Maria da Rocha Quintas
Centro de CIM do Porto

Prof. Doutor Diamantino Rui da Silva Freitas
Faculdade de Engenharia da Universidade do Porto

INTRODUÇÃO

Pretendeu-se com a realização deste projecto conceber e desenvolver um sistema de teste, quer para componentes electrónicos discretos, quer mesmo para placas de circuito impresso ou para qualquer outro tipo de circuitos electrónicos.

Como fácilmente compreenderemos o teste de qualquer circuito electrónico é fundamental, não só para verificar o seu correcto funcionamento, mas também para identificar eventuais defeitos de fabrico ou avarias e possibilitar diagnósticos nas situações de avaria.

Durante a fase de concepção deste sistema de teste procuramos desenvolver um sistema o mais genérico possível, que permitisse uma grande flexibilidade ao nível dos testes possíveis de efectuar a qualquer módulo ou componente electrónico.

De forma a responder a estes objectivos optamos por desenvolver um sistema completamente novo de raiz.

SOLUÇÃO ADOPTADA

O sistema, por nós concebido, compreende fundamentalmente três módulos:

1° - um módulo de interface com o utilizador, de forma a este poder optar pelos testes pretendidos e pelo local em que os pretendia realizar;

2° - um módulo de interface entre o software e o hardware;

3° - por último, um módulo de hardware, ou seja, o sistema encarregado de executar as ordens fornecidas pelo utilizador.

O primeiro módulo consta de um pacote de software, constituído por um programa principal e várias rotinas que fazem, como já foi dito, a interface entre o utilizador e o sistema de teste propriamente dito. Todo o software foi desenvolvido em ambiente MATLAB para PC. Se bem que o programa corra perfeitamente bem num PC com µP 386 é aconselhável que este disponha de coprocessador matemático, para aumentar a velocidade dos cálculos necessários, nomeadamente no que diz respeito a cálculos de ondas e cálculos de funções de correlação e convolução e de fft's.

O software desenvolvido permite ao utilizador gerar uma das seguintes formas de onda:

- Onda sinusoidal;
- Onda quadrada;
- Onda triangular;
- Onda dente de serra;
- Seno cardinal;
- Impulso rectangular;
- Impulso gaussiano;
- Componente continua;
- Degrau de heaviside;
- Onda exponencial,

e usando as operações matemáticas mais elementares, tais como, soma, subtracção e multiplicação gerar qualquer tipo de onda a partir da execução destas operações sobre duas ou mais ondas elementares;

Permite ainda operar quer sobre as ondas elementares quer sobre as mais complexas, não só através das operações matemáticas atrás referidas, mas também através da adição de componente contínua às ondas e a possibilidade de aumentar a amplitude das ondas, amplitude essa que é por defeito definida como sendo unitária.

O segundo módulo consta de um " Device Driver " que faz a interface entre o software a que o utilizador tem acesso e o hardware.O " Device Driver "é um modulo do sistema operativo que controla o hardware, servindo para isolar os níveis mais altos do sistema operativo, das caracteristicas especificas dos diferentes periféricos que estão ligados com o processador central.

O MS-DOS através do " Device Driver " transforma o hardware, do ponto de vista do software, num ficheiro. Isto permite aceder ao hardware através das funções de trabalho com ficheiros. Desta forma o envio de dados para o exterior (para o hardware) é visto como a escrita num ficheiro e a recepção de dados é realizada efectuando a leitura de um ficheiro.

O terceiro módulo consta de uma placa especialmente desenvolvida para esta aplicação especifica. Esta placa comunica com o PC através de uma das portas série (COM1, ...,COM4), sendo uma placa com conversores A/D, D/A e com linhas de I/O.

As linhas de I/O permitem identificar as agulhas da cama de testes onde se vão aplicar as ondas e de onde vão ser recolhidas as ondas de saída. Os conversores A/D e D/A permitem, respectivamente, que os sinais recolhidas nas agulhas sejam convertidas para forma digital e enviadas através da porta série do PC para posterior visualização ou tratamento e que as ondas geradas pelo utilizador, que estão guardadas na forma digital sejam enviadas para o exterior, sendo convertidas nesta placa para forma analógica após o que são aplicadas nas agulhas pretendidas.

IMPLEMENTAÇÃO

- Device driver

Pretende-se que sirva de interface neutro entre o Hardware desenvolvido com vista a teste de circuitos, e o software de teste.

Deste modo o software fica dispensado de conhecer o Hardware permitindo assim o desenvolvimento de Software em qualquer tipo de linguagem ou de sistemas de tratamento de dados.

- Princípios gerais

O "Device driver" é um Software desenvolvido para trabalhar em cima de um sistema operativo de tal modo que, é tratado pelo sistema com o programa de controlo de um determinado Hardware existente e ligado ao PC, e é disponibilizado pelo sistema operativo para o utilizador como se trata-se de um ficheiro sequencial.

Permite deste modo ser usado por qualquer programa que tenha a possibilidade de trabalhar em ficheiros, permitindo comunicar com o Hardware sem ter no entanto de saber como o este funciona, nem se preocupar com questões técnicas, tais como, os endereços, das portas, os "Timings", etc. .

Ficamos assim com a possibilidade de desenvolver qualquer Software genérico que funcionará em qualquer altura independentemente de alterações que possam vir a ser introduzidas no Hardware.

Alterações no Hardware só terão reflexão no "Device Driver" permitindo assim a alteração de apenas uma parte dum Software de maiores dimensões.

Existem dois tipos de "Device Drivers" que são designados consoante a sua funcionalidade:

- "Device Driver" de blocos
- "Device Driver" de caracteres

Como o nome indica, o primeiro tem por função fazer a transferência de Blocos de Bytes agrupando os Blocos segundo uma estrutura definida pelo sistema operativo e que se pode identificar com a estrutura de um disco duro.

Este tipo de "Device Driver" era usado exactamente para instalar discos no PC. Com a evolução dos PC's deixou de ser necessário usar-se um "Device Driver" para se instalar um disco duro, ficando isso ao encargo dos "Set Up" do PC. Estamos portanto num tipo de "Device Driver" que está a cair em desuso sendo usado apenas em aplicações muito especificas.

É normalmente associado o nome do disco (exemplo de um "Device Driver" de bloco →Stacker).

O segundo tipo de "Device Driver" permite fazer transferências de sequências de caracteres, (normalmente associado a um ficheiro sequencial). É o "Device Driver" mais vulgarmente usado e servindo para todo o tipo de interface Hardware- Software.

Este tipo de " Device Driver " pode apenas suportar uma unidade de hardware.

Como exemplo de um "Device Driver" de caracteres podemos referir os seguintes " Device Driver " (ficheiros):

- PRN
- LPT1
- COM1
- COM2
- AUX
- NUL

Foi este o "Device Driver" usado por nós, e toda a descrição que se segue é referente a um "Device Driver" do tipo caracter.

- Estrutura Básica

O "Device Driver" é um bloco de Software que é instalado no momento de arranque do computador, tem as funções atrás descritas, que pode ser dividido da seguinte forma:

- Rotina "Strategy"

É a rotina que é chamada no ínicio (chama a função 00 - instalação de " Device Driver ") e sempre que um qualquer programa faz uma operação sobre o ficheiro que representa o "Device Driver". O sistema operativo chama esta rotina para passar um apontador do cabeçalho, com que "Driver" é chamado.

Os primeiros treze Bytes do cabeçalho são os mesmos para todas as funções de " Device Driver " e são referidos como a parte estática do cabeçalho, o seu significado encontra-se como comentário na

listagem do " Device Driver ". Esta rotina tem apenas como função guardar o apontador numa variável conhecida pelo "Driver", retomando de seguida o controlo para o sistema operativo.

O sistema operativo chama de seguida a rotina que representa a terceira parte do "Driver" que é a rotina de *interrupt* a qual poderemos designar por *"Dispatch"*.

- ***"Interrupt"***

Esta rotina ao ser chamada pelo sistema operativo tem como função verificar o cabeçalho passado pelo mesmo e executar uma subrotina ligada á função que o sistema operativo pretende executar, sendo esta função a operação que o utilizador pretende realizar sobre o ficheiro representativo do "Device Driver" e que se vai rerflectir no hardware, um exemplo pode ser uma operação de leitura do ficheiro que pode representar uma operação de entrada numa porta controlada pelo " Device Driver ".

A rotina ***interrupt*** consiste usualmente nos seguintes elementos:

- Uma colecção de subrotinas que implementam as várias funções que podem ser requeridas pelo MS DOS (estas rotinas podem também ser designadas por "COMMAND CODE ROUTINES").
- Centralizar o ponto de entrada permitindo assim salvar todos os registos, extrair a função a ser executada do cabeçalho e saltar para a rotina respectiva.
- Centralizar o ponto de saida, permitindo guardar o erro, caso exista e repor todos os registos afectados.

- **"Command Code Routines"**

As rotinas assim designadas, implementam as várias funções suportadas pelo " Device Driver ".

Algumas destas rotinas são apenas suportadas (têm interesse) por " Device Driver " de caracteres e outras somente por " Device Driver " de blocos. No entanto algumas são vitais para os dois tipos.

A rotina seguinte (funcão 00) é a unica aqui descrita pela sua particularidade, todas as outras funções podem ser ou não escritas na totalidade , no entanto têm de existir, retornando apenas o código de erro. A descrição destas funçõees encontram-se como comentário na listagem.

Inicialização - FUNÇÃO 00

- A Rotina de inicialização, que tem por função instalar o "Device Driver", é chamada no momento em que é instalado o "Device Driver", isto é, no arranque do computador.

Esta rotina pode ser libertada no fim da instalação pois não volta a ser chamada.

- O "Device Driver" ADC

O "Device Driver" desenvolvido por ós segue todos os passos referidos atrás definindo as subrotinas referentes ás operações sobre os ficheiros de tal modo que as operações se reflictam no Hardware.

O Software de mais alto nível pode comunicar com o Hardware através de um "Device Driver" como se de um ficheiro se tratasse, no entanto foi necessário definir alguns comandos de controlo que permitem executar funções tal como a inicialização de Hardware e Software, fazer o Reset, etc. .... Para além desses comandos de controlo foi definido um protocolo de mensagens que permite definir o formato dos dados a serem transferidos de e para o Hardware, bem como o tipo de dados.

O formato da mensagem pode ser visto na fig. seguinte.

N° de agulhas	;	N° Agulha	;	1000 amostras	...	;	N°agulhas

Agulha N°1	;	Agulha N°2	;	....

Esta mensagem comporta o nº de agulhas a ser usadas como portas onde é injectado o sinal, as amostras que devem ser aplicadas a cada agulha, em que agulhas se deve ler o sinal.

A mensagem assim composta é enviada para o Device e este encarrega-se de a transmitir ao Hardware.

No momento em que se executa uma leitura do "Device Driver" é enviada uma outra mensagem com o formato seguinte:

Nº de agulhas	;	Nº Agulha	;	1000 amostras	...	;	Nºagulhas

Nesta mensagem é especificado o nº da agulha que se realizou a do teste ou seja onde se tem o sinal e o respectivo sinal em amostras de 8 bits.

Definiu-se ainda alguns comandos que permitem executar operações especiais sobre o Hardware. Estes comandos foram:

^R# - RESET
^C# - Clear Buffer
^S# - STOP
^I # - Inicializar
^N# - New Test
^r# - Resume
^O# - Ok

- Implementação

Desenvolveu-se um "Device Driver" de uma forma genérica, ou seja, o esqueleto do que poderia ser qualquer "Device Driver".
Como se encontrava em desenvolvimento o Hardware que seria usado no comando agulhas, deu-se mais importância à linha de comando que permite instalar o Device.

Isto permite-nos especificar determinadas características no momento de instalação, tais como os endereços das portas a usar, interrupts, velocidades de transferências, configurar o circuito, isto

é, nº de agulhas total, nº de agulhas de IN, nº de agulhas de OUT. ou nº de agulhas I/O.

Deu-se também importância à informação que o "Device Driver" dá no momento de instalação permitindo assim o utilizador verificar a instalação do "Device Driver".

Usou-se para este tipo de experiência a instalação do "Device Driver" para operar sobre as portas série (COM1, COM2, ..., COM4) que seram posteriormente usadaspara a comunicação com o Hardware da cama de teste. .

As funções implementadas no "Device Driver" estão a funcionar embora o que está implementado de momento nestas rotinas é especificar o erro ocorrido, ou seja, não ocorreu erro e retornar, no entanto a função de "open" e a função "close" incrementando e decrementando respectivamente uma variável que nos permite analisar a funcionalidade destas funções através de um programa escrito em linguagem de alto nível.

As funções de seguida a serem implementadas serão as funções "Read e Write" a partir do momento que se tenha obtido resultados no desempenho do Hardware que se está em desenvolvimento.

As listagens referntes a este "Device Driver" encontram-se em anexo e foram devididas consoante a sua funcionalidade. Podemos então encontrar:

- DRIVER.INC representa as macros e as definições de um modo geral usadas no software do "Device Driver";
- UART.INC representa as definições respectivas as portas de comunicação série, usadas também no software "Device Driver";
- DRIVER.ASM é o software propiamente dito do "Device Driver" e onde se encontram as rotinas atrás referidas;
- LINHACMD.ASM neste modulo é implemnetada a função 00 (inicialização do "Device Driver") e o software responsavel por processar a linha de comando de instalação do "Device Driver".

Todo o software encontra-se devidamente decomentado não se justificando por isso uma descrição exaustiva deste neste ponto.

BIBLIOGRAFIA

Skinnner Tom. *DATA AQUISITION: CONFIDENCE IN YOUR DATA DEPENDS ON YOUR UNDERSTANDING OF MEASUREMENT FUNDAMENTALS,* Philips Test & Measurement press article nº 609-752.

Philips, 1987. *Digital Measurement Center,* Industrial and Electro Acoustic Systems Division, October 1987, volume 15/3.

Fulke and Philips. *Math+ processing package reveals hidden information in oscilloscope waveforms,* application note.

Ardley T., Cox D., Faulkner A., Goacher S., Hipwood B., Hugo R., Lichtblau A., Maytum M., Ramsdale S., Robinson D., 1992. *Linear Design Seminar - Slide Book.*

Electrónica Elektor, 1992. *Conversor de duplo traço para osciloscópio,* Outubro 1986.

Electrónica Elektor, 1986. *Convesão A/D e D/A, Desacoplamento em circuitos digitais, Conversor analógico/digital,* Fevereiro 1986.

Electrónica Elektor, 1989. *Carta de E/S para o IBM-PC e compatíveis,* Março 1989.

Electrónica Elektor, 1992. *Gerador de interrupção para o PC,* Julho/Agosto 1992.

Electrónica Elektor, 1990. *Mini placa de E/S para IBMPC,* Julho/Agosto 1990.

Electrónica Elektor, 1990. *Amplificador de entrada para osciloscópio,* Julho/Agosto 1992.

Electrónica Elektor, 1989. *Adaptador de Barramento E/S para PC*, Julho/Agosto 1992.

Becker John, 1991. *PC - Scope Interface*, Everyday Electronics, October 1991.

Duncan Ray, 1986. *Advanced MS - DOS*, Microsoft, June 1986.

IBM, 1984. *IBM Technical Reference Publications*, IBM, February 1984.

Choisser John P. & Foster John O., 1988/89. *AT BIOS KIT " The Comprehensive Guide to Creating an AT Bios in C "*, Annabooks, November 1989.

Hall V. Douglas, 1992. *Microprocessors and interfacing " Programming and Hardware "*, MC GRAW HILL, 1992.

ÍNDICE GERAL

ANEXOS

Ficheiro:

DRIVER.INC

```
;=============================================================================
; DRIVER.MAC : Estruturas, macros e equates necessárias para
;          criar um device driver.
;=============================================================================

;
; Estrutura que permite aceder a um endereço como uma DWORD ou duas WORD.
;
UNION ENDERECO
        ENDCOMP             DD      ?       ; Endereço completo (DWORD=4bytes)
        STRUC
                ENDOFS      DW      ?       ; Offset do endereço
                ENDSEG      DW      ?       ; Segmento do endereço
        ENDS
ENDS

;
; Estrutura do cabeçalho de um device driver
;
STRUC DEV_HEADER
  PROX_DD   ENDERECO <-1>     ; Endereço do próximo device driver
  ATRIBUTO  DW      ?         ; Atributos do device driver
  STRAT_EP  DW      ?           ; Offset da rotina STRAT
  INTR_EP   DW      ?                   ; Offset da rotina INTR
  NOME      DB      8 DUP (?)   ; Nome do device driver
ENDS

;
; Campo ATRIBUTO para Character Devices
;
STDIN           EQU     0001h       ; Device is console input (stdin)
STDOUT          EQU     0002h       ; Device is console output (stdout)
NUL             EQU     0040h       ; Device supports DOS 3.2 functions
CLOCK           EQU     0008h       ; Device is CLOCK device
DOS32           EQU     0040h       ; Device supports DOS 3.2 functions
OPEN_CLOSE      EQU     0800h       ; Device understands Open/Close
OUB             EQU     2000h       ; Device supports Output Until Busy (OUB)
IOCTL_STR       EQU     4000h       ; Device supports IOCTL control strings
CHARACTER       EQU     8000h       ; Character device


;
; Estrutura do Request Header
;
STRUC REQ_HEADER
  LENGTH            DB      ?           ; Tamanho da estrutura (em bytes)
  UNIT              DB      ?           ; Sem significado para Character DD
  COMMAND           DB      ?           ; Código do comando
  STATUS            DW      ?                   ; Estado
  RESERVED          DB      8 DUP (?)   ; Reservados
```

```
ENDS

;
; Campo STATUS do Request Header (Status field)
;
ST_DONE     EQU    0100h
ST_BUSY     EQU    0200h
ST_ERROR    EQU    8000h
;
; Erro                  Significado
;
DD_ERROR_0 EQU    ST_ERROR+00h   ; 0 - Write protect violation
DD_ERROR_1 EQU    ST_ERROR+01h   ; 1 - Unknown unit
DD_ERROR_2 EQU    ST_ERROR+02h   ; 2 - Drive not ready
DD_ERROR_3 EQU    ST_ERROR+03h   ; 3 - Unknown command
DD_ERROR_4 EQU    ST_ERROR+04h   ; 4 - CRC error
DD_ERROR_5 EQU    ST_ERROR+05h   ; 5 - Bad drive request structure length
DD_ERROR_6 EQU    ST_ERROR+06h   ; 6 - Seek error
DD_ERROR_7 EQU    ST_ERROR+07h   ; 7 - Unknown media
DD_ERROR_8 EQU    ST_ERROR+08h   ; 8 - Sector not found
DD_ERROR_9 EQU    ST_ERROR+09h   ; 9 - Printer out of paper
DD_ERROR_A EQU    ST_ERROR+0Ah   ; A - Write fault
DD_ERROR_B EQU    ST_ERROR+0Bh   ; B - Read fault
DD_ERROR_C EQU    ST_ERROR+0Ch   ; C - General failure
DD_ERROR_D EQU    ST_ERROR+0Dh   ; D - Reserved
DD_ERROR_E EQU    ST_ERROR+0Eh   ; E - Reserved
DD_ERROR_F EQU    ST_ERROR+0Fh   ; F - Invalid disk change


;
; Estrutura do Request Header da função INIT (comando 0)
;
STRUC INIT_RH
                    DB      SIZE REQ_HEADER DUP (?)
  UNITS             DB          ?               ; Sem significado para Character DD
  DD_END            ENDERECO ?                  ; Endereço do fim do driver
  BPB_PTR           ENDERECO ?                  ; Sem significado para Character DD
  BLOCK_NUM         DB          ?               ; Sem significado para Character DD
ENDS

;
; Estrutura do Request Header das funções READ e WRITE
; (comandos 3, 4, 8, 9, 12 e 16)
;
STRUC RW_RH
                    DB      SIZE REQ_HEADER DUP (?)
  MEDIA             DB          ?       ; Media Descriptor from BPB
  TRANSF_ADR        ENDERECO    ?       ; Transfer address
  BYTE_CNT          DW          ?       ; Byte/sector count
  START_SCT         DW          ?       ; Sem significado para Character DD
  VOLUME_PTR        ENDERECO    ?       ; DWORD pointer to requested Volume ID
ENDS
```

```
;
; Estrutura do Request Header da função IOCtl (comando 19)
;
STRUC IOCTL_RH
                DB      SIZE REQ_HEADER DUP (?)
  CATEGORY      DB              ?       ; Category (major) code
  FUNCTION      DB              ?       ; Function (minor) code
  SI_CONT       DW              ?       ; Conteudo do registo SI
  DI_CONT       DW              ?       ; Conteudo do registo DI
  DATA_BUF      ENDERECO        ?       ; Pointer para o buffer de dados
ENDS


;
; Equates diversas
;
CR              EQU     0Dh             ; Carriage return
LF              EQU     0Ah             ; Line feed
BELL            EQU     07h             ; Bell
TAB             EQU     09h             ; Horizontal Tab
SPACE           EQU     20h             ; Space
EOM             EQU     '$'             ; End-of-message (INT 21h Serviço 9)
```

Ficheiro:

UART.INC

```
;=============================================================================
; PIC 8259
;=============================================================================
;
; Portas I/O do primeiro PIC ( 20h e 21 h )
;
PIC_PEOI      EQU   20h     ; Porta I/O do PIC para comandos EOI
                            ;   (OCW2)
PIC_PMASK     EQU   21h     ; Porta I/O do PIC para a mascara de
                            ;   interrupç_es (OCW1)
;
; Comandos
;
PIC_EOI   EQU   20h     ; Non-specific End of interrupt


;=============================================================================
; UART 8250
;=============================================================================

;
; Offset dos registos internos do 8250,16450,16550.
;
RBR             EQU   0     ; Receive buffer register (Read only)
THR             EQU   0     ; Transmit buffer register (Write only)
IER             EQU   1     ; Interrupt enable register
IIR             EQU   2     ; Interrupt ID register (Read only)
FCR             EQU   2     ; FIFO control (Write only) (16550)
LCR             EQU   3     ; Line control register
MCR             EQU   4     ; Modem control register
LSR             EQU   5     ; Line status register
MSR             EQU   6     ; Modem status register
SCR             EQU   7     ; Scratch register

DLL             EQU   0     ; Divisor latch LSB
DLM             EQU   1     ; Divisor latch MSB


;
; Bits do registo 'Line control'
;
; LCR(0) e LCR(1) : Word select

LCR_DATA5  EQU    00h     ; 5 Data bits
LCR_DATA6  EQU    01h     ; 6 Data bits
LCR_DATA7  EQU    02h     ; 7 Data bits
LCR_DATA8  EQU    03h     ; 8 Data bits

; LCR(2) : Stop bit select
```

```
LCR_STOP1       EQU   00h     ; 1 Stop bit
LCR_STOP2       EQU   04h     ; 2 Stop bits

; LCR(3) : Parity Enable

LCR_NOPAR       EQU   00h     ; No parity

; LCR(4) : Even parity select

LCR_ODDPAR      EQU   08h     ; Odd parity
LCR_EVNPAR      EQU   18h     ; Even parity

LCR_STKPAR      EQU   28h     ; LCR(5) : Stick parity
LCR_BREAK       EQU   40h     ; LCR(6) : Break control
LCR_DLAB        EQU   80h     ; LCR(7) : Divisor latch access bit


;
; Bits do registo 'Line status'
;
LSR_DR      EQU   01h     ; Data ready
LSR_OE      EQU   02h     ; Overrun error
LSR_PE      EQU   04h     ; Parity error
LSR_FE      EQU   08h     ; Framing error
LSR_BI      EQU   10h     ; Break interrupt
LSR_THRE    EQU   20h     ; Transmitter holding register empty
LSR_TEMT    EQU   40h     ; Transmitter empty

LSR_ERRO    EQU   1Eh     ; Todos os erros


;
; Bits do registo 'Modem control'
;
MCR_DTR     EQU   01h     ; Data terminal ready
MCR_RTS     EQU   02h     ; Request to send
MCR_OUT1    EQU   04h     ; Output #1
MCR_OUT2    EQU   08h     ; Output #2
MCR_LOOP    EQU   10h     ; Loopback mode bit

;
; Bits do registo 'Modem status'
;
MSR_DCTS    EQU   01h     ; Delta clear to send
MSR_DDSR    EQU   02h     ; Delta data set ready
MSR_TERI    EQU   04h     ; Trailing edge ring indicator
MSR_DDCD    EQU   08h     ; Delta data carrier detect
MSR_CTS     EQU   10h     ; Clear to send
MSR_DSR     EQU   20h     ; Data set ready
MSR_RI      EQU   40h     ; Ring indicator
MSR_DCD     EQU   80h     ; Data carrier detect


;
```

```
; Bits do registo 'Interrupt enable'
;
IER_RDA     EQU   01h     ; Receiver data available
IER_THRE    EQU   02h     ; Transmitter holding register empty
IER_RLS     EQU   04h     ; Receive line status
IER_MS      EQU   08h     ; Modem status


;
; Bits do registo 'Interrupt identification'
;
; 8250 & 16450
;
IIR_MS      EQU   0       ; Modem status (Menor prioridade)
IIR_THRE    EQU   2       ; Transmitter holding register empty
IIR_RDA     EQU   4       ; Receiver data available
IIR_RLS     EQU   6       ; Receive line status (Maior prioridade)

; 16550 ver Datasheet

;
; Bits do registo 'FIFO control' (16550)
;
FCR_FEWO    EQU   01h     ; FIFO enable (transmit & receive)
FCR_RFR     EQU   02h     ; Receiver FIFO reset
FCR_TFR     EQU   04h     ; Transmitter FIFO reset
FCR_DMS     EQU   08h     ; DMA mode select

; FCR(5) e FCR(6) : trigger level

FCR_TL01    EQU   00h     ; FIFO trigger level (01 byte)
FCR_TL04    EQU   40h     ; FIFO trigger level (04 bytes)
FCR_TL08    EQU   80h     ; FIFO trigger level (08 bytes)
FCR_TL14    EQU   C0h     ; FIFO trigger level (14 bytes)
```

Ficheiro:

DRIVER.ASM

```
IDEAL
P8086

NAME      ADC
%PAGESIZE  60,132
%TITLE    "DRIVER.ASM: Esqueleto de um device driver"

;=============================================================================
; DRIVER.ASM : Esqueleto de um 'character' device driver
;
;                 Autor : José Carlos Lobinho Gomes
;
;              Data ínicio : 93/03/22
;
; Data da última modificação : 93/09/24
;
;=============================================================================
; Para criar o device driver fazer:
;
;         > tasm driver+linhacmd
;         > tlink /t driver+linhacmd,driver.sys
;
;=============================================================================

MODEL     TINY

INCLUDE   "DRIVER.INC"


PUBLIC    HEADER
PUBLIC    RHPTR
EXTRN     INIT:NEAR

;-----------------------------------------------------------------
CODESEG
;-----------------------------------------------------------------

      org    0000h

IFDEF     DEBUG
        DISPLAY "*********************************************"
        DISPLAY "*   A D C - Versão de desenvolvimento       *"
        DISPLAY "*********************************************"
ENDIF


MaxCmd    EQU    24          ; Maximum allowed command code
                             ;  DOS 2.0      : 12
                             ;  DOS 3.0,3.1  : 16
                             ;  DOS 3.2,3.3  : 24
```

```
                        ;  DOS 4.0      : 24
                        ;  DOS 5.0      : 24 ?


Cr      equ     0dh
Lf      equ     0ah
eom     equ     '$'

;
;       device Driver Header
;

Header  DEV_HEADER <-1,         \ Link to next device driver
                CHARACTER+  \ Device attribute word
                IOCTL_STR+  \
                OPEN_CLOSE+ \
                DOS32,      \
                STRAT,      \ "Strategy" routine entry point
                INTR,       \ "Interrupt" routine entry point
                'ADC    '> \ Logical device name



        ;
        ; Pointer to request header, passed by MS-DOS kernel
        ;  to strategy routine
        ;
RHPtr   ENDERECO <?>


    ;
    ; Interrupt-routine command-code dispatch table
    ;
LABEL DISPATCH WORD
        DW      INIT                    ; 0 = Inicialize Driver
        DW      NOT_USED                        ; 1 = Media Check (Block driver)
        DW      NOT_USED                        ; 2 = Build BPB (Block driver)
        DW      IOCTLRD                 ; 3 = IOCTL Read
        DW      READ                    ; 4 = Read
        DW      NDREAD                  ; 5 = Nondesdructive Read (Character driver)
        DW      INPSTAT                 ; 6 = Input Status (Character driver)
        DW      INPFLUSH                ; 7 = Flush Input Buffers (Character driver)
        DW      WRITE                   ; 8 = Write
        DW      WRITE                   ; 9 = Write with Verify (Block driver)
        DW      OUTSTAT                 ; 10 = Output Status (Character driver)
        DW      OUTFLUSH                        ; 11 = Flush Output Buffers (Character
driver)
        DW      IOCTLWT                 ; 12 = IOTCL Write
        DW      DEVOPEN                 ; 13 = Device Open
        DW      DEVCLOSE                ; 14 = Device Close
        DW      NOT_USED                        ; 15 = Removable Media (Block devices)
        DW      NOT_USED                        ; 16 = Output Until Busy (Character driver)
        DW      ILLEGAL_CMD             ; 17 = Not Documented
```

```
        DW      ILLEGAL_CMD         ; 18 = Not Documented
        DW      GENIOCTL            ; 19 = Generic IOTCL
        DW      ILLEGAL_CMD         ; 20 = Not Documented
        DW      ILLEGAL_CMD         ; 21 = Not Documented
        DW      ILLEGAL_CMD         ; 22 = Not Documented
        DW      NOT_USED                    ; 23 = Get Logical Device (Block devices)
        DW      NOT_USED                    ; 24 = Set Logical Device (Block devices)

DD_ABERTO DB    0                   ; Se Aberto então DD_ABERTO = 1
                                    ;  senão DD_ABERTO = 0

        EVEN
;--------------------------------------------------------------
; STRAT:  Device-driver Strategy routine, called by MS-DOS
;       kernel with ES:BX = address of request header
;--------------------------------------------------------------
PROC STRAT FAR
        MOV     [CS:RHPTR.ENDOFS],BX    ; Save pointer to
        MOV     [CS:RHPTR.ENDSEG],ES    ;  request header
        RET
ENDP

        EVEN
;--------------------------------------------------------------
; INTR:   Device-driver Interrupt routine, called by MS-DOS
;       kernel immediately after call to Strategy routine
;--------------------------------------------------------------
PROC INTR FAR
        PUSH    AX BX CX DX DS ES DI SI BP

        PUSH    CS              ; Make local data addressable
        POP     DS              ;  by setting DS=CS

        LES     DI,[RHPTR.ENDCOMP]  ; ES:DI -> request header

        ;
        ; Ver se o comando é válido (0..24)
        ;
        MOV     BL,[(REQ_HEADER PTR ES:DI).COMMAND]
        XOR     BH,BH
        CMP     BX,MAXCMD           ; Ver se o comando é válido
        JLE     @@L1
        CALL    ILLEGAL_CMD         ; Comando inválido
        JMP     SHORT @@L2

    @@L1:
        SHL     BX,1
        CALL    [WORD PTR BX+DISPATCH]

        LES     DI,[RHPTR.ENDCOMP]  ; ES:DI -> request header

    @@L2:
        OR      AX,ST_DONE              ; Merge DONE BIT
```

```
        ; Store STATUS BIT into request header
        MOV     [(REQ_HEADER PTR ES:DI).STATUS],AX

        POP     BP SI DI ES DS DX CX BX AX
        RET
ENDP


;---------------------------------------------------------
; Função 3 = IOCTL READ
;---------------------------------------------------------
PROC IOCTLRD NEAR
        XOR     AX,AX
        RET
ENDP

;---------------------------------------------------------
; Função 4 = READ (INPUT)
;---------------------------------------------------------
PROC READ  NEAR
        XOR     AX,AX
        RET
ENDP

;---------------------------------------------------------
; Função 5 = NONDESTRUCTIVE READ
;---------------------------------------------------------
PROC NDREAD NEAR
        XOR     AX,AX
        RET
ENDP

;---------------------------------------------------------
; Função 6 = INPUT STATUS
;---------------------------------------------------------
PROC INPSTAT NEAR
        XOR     AX,AX
        RET
ENDP

;---------------------------------------------------------
; Função 7 = FLUSH INPUT BUFFERS
;---------------------------------------------------------
PROC INPFLUSH NEAR
        XOR     AX,AX
        RET
ENDP

;---------------------------------------------------------
; Função 8 = WRITE (OUTPUT)
;---------------------------------------------------------
PROC WRITE NEAR
        XOR     AX,AX
        RET
```

```
ENDP

;--------------------------------------------------------
; Função 10 = OUTPUT STATUS
;--------------------------------------------------------
PROC OUTSTAT NEAR
        XOR     AX,AX
        RET
ENDP

;--------------------------------------------------------
; Função 11 = FLUSH OUTPUT BUFFERS
;--------------------------------------------------------
PROC OUTFLUSH NEAR
        XOR     AX,AX
        RET
ENDP

;--------------------------------------------------------
; Função 12 = IOCTL WRITE
;--------------------------------------------------------
PROC IOCTLWT NEAR
        XOR     AX,AX
        RET
ENDP

;--------------------------------------------------------
; Função 13 = DEVICE OPEN
;--------------------------------------------------------
PROC DEVOPEN NEAR
        CMP     [DD_ABERTO],0
        JNE     @@JA_ABERTO
        INC     [DD_ABERTO]
        XOR     AX,AX
        RET
@@JA_ABERTO:
        MOV     AX,DD_ERROR_C  ; General failure
        RET
ENDP

;--------------------------------------------------------
; Função 14 = DEVICE CLOSE
;--------------------------------------------------------
PROC DEVCLOSE NEAR
        CMP     [DD_ABERTO],0
        JE      @@JA_FECHADO
        DEC     [DD_ABERTO]
        XOR     AX,AX
        RET
@@JA_FECHADO:
        MOV     AX,DD_ERROR_C  ; General failure
        RET
ENDP
```

```
;-------------------------------------------------------------
; Função 19 = GENERIC IOCTL
;-------------------------------------------------------------
PROC GENIOCTL NEAR
        XOR     AX,AX
        RET
ENDP

;-------------------------------------------------------------
; NOT_USED : Funções não usadas em device drivers de caracter.
;-------------------------------------------------------------
PROC NOT_USED NEAR
        XOR     AX,AX
        RET
ENDP

;-------------------------------------------------------------
; ILLEGAL_CMD : Funções não documentadas
;           ( Funções: 17, 18, 20, 21, 22 e >MAXCMD )
;-------------------------------------------------------------
PROC ILLEGAL_CMD NEAR
        MOV     AX,DD_ERROR_3       ; Error Bit +
                                    ; "Unknown command" code
        RET
ENDP

;=============================================================
;=============================================================
; Rotinas de gestão da porta série
;=============================================================
;=============================================================

INCLUDE   "UART.INC"
;-------------------------------------------------------------
;-------------------------------------------------------------
EVEN
PROC UART_ISR FAR
        PUSH    AX BX CX DX SI DI BP DS ES

        ;
        ;
        ;
        STI                 ; Permite a ocurrência
                            ; de interrupç_es

NOVA_UART_INT:
        MOV     DX,[UART_IIR]       ; Ler registo IIR
        IN      AL,DX               ; da UART


        TEST    AL,01h
        JNZ     @@FIM_INT

        ;
```

```
        ; Saltar para a rotina de atendimento da interrupção
        ;
        MOV     BL,AL
        XOR     BH,BH
        JMP     [BX+TABELA_INT]


 @@FIM_INT:
        ;
        ; Sinalizar fim da interrupção para o PIC 8259
        ;
        MOV     AL,PIC_EOI          ; Carrega comando EOI
        OUT     PIC_PEOI,AL         ; Manda comando para o PIC

        POP     ES DS BP DI SI DX CX BX AX
        IRET
ENDP

;-------------------------------------------------------------
; Modem status
;-------------------------------------------------------------
PROC UART_ISR_MS NEAR
        MOV     DX,[UART_MSR]       ; Limpar pedido de interrupção
        IN      AL,DX

        JMP     SHORT NOVA_UART_INT
ENDP

;-------------------------------------------------------------
; Transmitter holding register empty
;-------------------------------------------------------------
PROC UART_ISR_THRE NEAR


        JMP     SHORT NOVA_UART_INT
ENDP


;-------------------------------------------------------------
; Receiver data available
;-------------------------------------------------------------
PROC UART_ISR_RDR NEAR

        MOV     DX,[UART_RBR]       ; Limpar pedido de interrupção
        IN      AL,DX

        JMP     SHORT NOVA_UART_INT
ENDP


;-------------------------------------------------------------
; Receive line status
;-------------------------------------------------------------
PROC UART_ISR_RLS NEAR
```

```
        MOV     DX,[UART_LSR]       ; Limpar pedido de interrupção
        IN      AL,DX

        JMP     SHORT NOVA_UART_INT
ENDP


;
; Portas da UART
; Nota: Falta somar o endereço I/O base da UART.
;
EVEN
LABEL UART_RBR WORD             ; Receive buffer register (Read only)
LABEL UART_THR WORD             ; Transmit buffer register (Write only)
LABEL UART_DLL WORD             ; Divisor latch LSB
        DW      RBR
LABEL UART_IER WORD             ; Interrupt enable register
LABEL UART_DLM WORD             ; Divisor latch MSB
        DW      IER
LABEL UART_IIR WORD             ; Interrupt ID register (Read only)
LABEL UART_FCR WORD             ; FIFO control (Write only) (16550)
        DW      IIR
LABEL UART_LCR WORD             ; Line control register
        DW      LCR
LABEL UART_MCR WORD             ; Modem control register
        DW      MCR
LABEL UART_LSR WORD             ; Line status register
        DW      LSR
LABEL UART_MSR WORD             ; Modem status register
        DW      MSR
LABEL UART_SCR WORD             ; Scratch register
        DW      SCR


TABELA_INT
        DW      UART_ISR_MS     ; Modem status
        DW      UART_ISR_THRE   ; Transmitter holding register empty
        DW      UART_ISR_RDR    ; Receiver data available
        DW      UART_ISR_RLS    ; Receive line status

;
; Para o tamanho dos buffers de leitura usar potências de 2
;    (exemplo: 32, 64, 128, 256, 512, ...
;
COMP_BUF_LEITURA  EQU 256
COMP_BUF_ESCRITA  EQU 128
                ; Buffer de leitura
BL_READ           DW    0         ; > apontador de leitura
BL_WRITE          DW    0         ; > apontador de escrita
BUF_LEITURA       DB    COMP_BUF_LEITURA DUP('L')

                ; Buffer de escrita
BE_READ           DW    0         ; > apontador de leitura
```

```
BE_WRITE            DW    0          ; > apontador de escrita
BUF_ESCRITA         DB    COMP_BUF_ESCRITA DUP('E')


;===============================================================
; Fim do código residente do device driver
;===============================================================

IFDEF DEBUG
        DB      '*** FIM DA PARTE RESIDENTE DO DEVICE DRIVER ***'
ENDIF

        END
```

Ficheiro:

LINHACMD.ASM

```
IDEAL
P8086
MODEL    TINY

INCLUDE   "DRIVER.INC"

;=============================================================================
; Interface
;=============================================================================

PUBLIC    INIT

EXTRN     HEADER:DEV_HEADER
EXTRN     RHPTR:ENDERECO

;=============================================================================
; Implementation
;=============================================================================



;-----------------------------------------------------------------
DATASEG
;-----------------------------------------------------------------

IDENTIF   DB     CR, LF, LF
          DB     'MS-DOS CHARACTER DEVICE DRIVER (RS232485)', CR,LF
          DB     'Copyright (c) 1993 by FEUP-INESC', CR, LF, LF, EOM


MSGERRO             DB     '  Erro:', BELL, CR, LF
                    DB     '     Device driver não instalado. '
                    DB     'Demasiados argumentos.', CR, LF, EOM
MSGSINTAXE          DB     LF, '  Sintaxe:', CR, LF
                    DB     ' DEVICE=[drive:][path]DRIVER.SYS [COMn] [DEBUG]'
                    DB     CR, LF, EOM

MSGDEBUG            DB     '   Assemblado em ', ??DATE, ' às ', ??TIME, CR, LF
                    DB     '   Endereço do Device Driver: desde '
  ENDER:            DB     'XXXX:0000 até '            ; Segmento:Offset
 ENDERFIM:          DB     'XXXX:XXXX', CR, LF         ; Segmento:Offset

                    DB     '   Número de portas série detectadas (INT 11h): '
  MINT11:           DB     'XXXX', CR, LF
                    DB     ' BIOS Low Memory Area (Segmento 0040h)', CR, LF, TAB
                    DB     'COM1  COM2  COM3  COM4', CR, LF, TAB
 MSGBIOS:           DB     'XXXX  XXXX  XXXX  XXXX', CR, LF

INVOCSIZE           EQU    128
```

```
                DB      '  DEVICE='
INVOC2:         DB      INVOCSIZE DUP (EOM)


MAXARG          EQU     2                       ; Número máximo de argumentos
                                                ;   permitidos
ARGC            DB      -1                      ; Número de argumentos existentes
ARGV            DW      MAXARG+1 DUP (0)        ; Array de apontadores para os
                                                ;  argumentos (terminados com '\0')

                DB      'Argumentos:'
BUFSIZE         EQU     64                      ; Tamanho do BUFFER
BUFFER          DB      BUFSIZE DUP (0FFh)      ; Buffer para conter os argumentos


;---------------------------------------------------------------
CODESEG
;---------------------------------------------------------------

;===============================================================
; Código de inicialização
;===============================================================
;       O código que se segue é descartado depois da
;       inicialização do device driver.
;           BREAK ADDRESS = INIT ADDRESS
;===============================================================

;---------------------------------------------------------
; Função 0 = INIT
;
; Entrada  : ES:DI -> Request Header
;          DS    -> Segmento do device driver (DS = CS)
;
;        Nota: A rotina INIT só pode chamar as funç_es
;              01h-0Ch, 25h, 30h e 25h do INT 21h
;
; Saída    :
;
;
;  Abortar instalação
;  ------------------
;  Um device driver de caracter que queira abortar a sua
;  instalação deve limpar o bit 15 do seu ATRIBUTO e em seguida
;  inicializar o campo UNITS e o endereço de MEM_RIA LIVRE como
;  se tratasse de um device driver de bloco.
;
;  Resumindo:
;    HEADER.ATRIBUTO = HEADER.ATRIBUTO AND NOT HEADER.ATRIBUTO
;    INIT_RH.UNITS   = 0
;    INIT_RH.DD_END  = CS:0000h
;---------------------------------------------------------
```

```
PROC INIT NEAR
        ;
        ; Set address of free memory above driver (break address)
        ;
        MOV     [(INIT_RH PTR ES:DI).DD_END.ENDSEG],CS
IFDEF DEBUG
        MOV     [(INIT_RH PTR ES:DI).DD_END.ENDOFS],OFFSET FIM_DRIVER
ELSE
        MOV     [(INIT_RH PTR ES:DI).DD_END.ENDOFS],OFFSET INIT
ENDIF
        ;
        ; Mensagem de identificação do device driver
        ;
        MOV     AH,9
        MOV     DX,OFFSET IDENTIF
        INT     21H

        ;
        ; Processar linha de invocação do device driver
        ;
        CALL    PROCESSAR_LINHA_COMANDO
        AND     AL,AL           ; Se AL<>0 houve um erro
        JZ      @@CONT1

        MOV     AH,9h           ; Imprime mensagem de erro
        MOV     DX,OFFSET MSGERRO
        INT     21h

        MOV     AH,9h           ; Imprime Sintaxe correcta
        MOV     DX,OFFSET MSGSINTAXE
        INT     21h
        ;
        ; Abortar instalação do device driver
        ;  (Ler NOTA no cabeçalho desta função)
        MOV     AX,[HEADER.ATRIBUTO]    ; Fazer reset ao bit 15
        AND     AX,NOT CHARACTER        ; do campo ATRIBUTO DO
        MOV     [HEADER.ATRIBUTO],AX    ; cabeçalho do driver.

        LES     DI,[RHPTR.ENDCOMP]  ; ES:DI -> Init request header
        MOV     [(INIT_RH PTR ES:DI).UNITS],0
        MOV     [(INIT_RH PTR ES:DI).DD_END.ENDOFS],0
        JMP     SHORT @@FIM
  @@CONT1:

        ;
        ; Endereço do device driver
        ;
        MOV     AX,CS           ; Início do device driver
        MOV     BX,OFFSET ENDER
        CALL    HEXASC

        MOV     AX,CS           ; Fim do device driver
        MOV     BX,OFFSET ENDERFIM
        CALL    HEXASC
```

```
    MOV    AX,OFFSET FIM_DRIVER
    MOV    BX,OFFSET ENDERFIM+5
    CALL   HEXASC
    ;
    ; Processar linha de invocação do device driver
    ;
    LES    DI,[RHPTR.ENDCOMP]
    MOV    BX,DS
    LDS    SI,[(INIT_RH PTR ES:DI).BPB_PTR.ENDCOMP]
    MOV    ES,BX          ; BX=ES=DATA SEGMENT
    MOV    DI,OFFSET INVOC2
    MOV    CX,INVOCSIZE - 4
    CLD
@@L1:
    LODSB
    CMP    AL,CR
    JE     @@L2
    CMP    AL,LF
    JE     @@L2
    STOSB
    LOOP   @@L1
@@L2:
    MOV    AX,CR + LF SHL 8
    STOSW
    MOV    AX,LF + EOM SHL 8
    STOSW

    MOV    DS,BX

    ;
    ; Determinar o número de UARTs instaladas no sistema
    ;  através do interrupt 11h
    ;
    INT    11h
    AND    AX,0E00h
    MOV    CL,9
    SHR    AX,CL
    MOV    BX,OFFSET MINT11
    CALL   HEXASC
    ;
    ; Portas I/O das diversas UARTs
    ;
    MOV    AX,0040h        ; BIOS Low Memory Area 00400h
    MOV    ES,AX           ;   COM1 I/O Port = [40:0h]
    MOV    DI,0            ;   COM2 I/O Port = [40:2h]
    MOV    BX,OFFSET MSGBIOS  ;   COM3 I/O Port = [40:4h]
    MOV    CX,4            ;   COM4 I/O Port = [40:6h]
@@Portas:
    MOV    AX,[ES:DI]      ; Obter porta I/O da COMn
    CALL   HEXASC          ; Converter para hexadecimal
    ADD    DI,2
    ADD    BX,2
    LOOP   @@Portas
```

```
        MOV     AH,9h
        MOV     DX,OFFSET MSGDEBUG
        INT     21h


    @@FIM:
        XOR     AX,AX
        RET

ENDP



;---------------------------------------------------------------
; HEXASC   : Converts word to hex ASCII
;          Call with AX = value, DS:BX = address for string
;          Returns AX, BX destroyed
;---------------------------------------------------------------
PROC HEXASC NEAR

        PUSH    CX DX

        MOV     DX,4
    @@L1:
        MOV     CX,4
        ROL     AX,CL
        MOV     CX,AX
        AND     CX,0FH
        ADD     CX,'0'
        CMP     CX,'9'
        JBE     @@L2
        ADD     CX,'A'-'9'-1

    @@L2:
        MOV     [BX],CL
        INC     BX

        DEC     DX
        JNZ     @@L1

        POP     DX CX
        RET
ENDP


;---------------------------------------------------------------
; PROCESSAR_LINHA_COMANDO
;
; Entrada  : ES:DI -> Request header da função INIT (INIT_RH)
;          DS    -> Segmento do device driver (DS = CS)
;
;          Nota
;          ----
;          A linha de comando com o seguinte formato:
```

```
;          [drive:][path]nome_device_driver [[parametro]...]
;       começa sempre com um caracter não branco e
;       termina com um caracter Carriage Return (CR) ou Line
;         Feed (LF).
;
; Saída   : Destroi registos AX,BX,CX,DX,DI,SI,ES
;         Mantêm registo DS
;         AL = 0 - Nada a assinalar
;              1 - Erro : Demasiados argumentos
;              2 -
;-------------------------------------------------------------
PROC PROCESSAR_LINHA_COMANDO NEAR
        ;
        ; Processar linha de invocação do device driver
        ;
        MOV    AX,DS
        LDS    SI,[(INIT_RH PTR ES:DI).BPB_PTR.ENDCOMP]
        MOV    ES,AX            ; ES = DATA SEGMENT
        MOV    DI,OFFSET BUFFER
        XOR    AH,AH            ; AH = Ultimo caracter
        XOR    CX,CX            ; Contador de letras
        MOV    DX,CX            ;
        CLD
@@LER_CARACTER:
        LODSB                   ; Ler caracter
        CMP    AL,CR            ; Carriage Return ?
        JE     @@FIM_LC
        CMP    AL,LF            ; Line Feed ?
        JE     @@FIM_LC
        CMP    AL,SPACE         ; Espaço ?
        JBE    @@CARACTER_BRANCO
        ;
        ;
        ;
        OR     AH,AH            ; Ultimo caracter era Branco?
        JNZ    @@CAR_NB2
        XOR    BH,BH
        MOV    BL,[CS:ARGC]
        INC    BL
        CMP    BL,MAXARG
        JBE    @@CAR_NB1
        MOV    AL,1             ; ERRO: Demasiados argumentos
        JMP    SHORT @@FIM
 @@CAR_NB1:
        MOV    [CS:ARGC],BL     ; Numero de argumentos
        SHL    BL,1             ; ARGC = ARGC*2
        ADD    BX,OFFSET ARGV
        MOV    [CS:BX],DI       ; ARGV[ARGC] = Endereço do argumento
 @@CAR_NB2:
        STOSB                   ; Guarda caracter
        MOV    AH,AL            ; Ultimo caracter = caracter
        JMP    SHORT @@LER_CARACTER
        ;
        ;
```

```
        ;
@@CARACTER_BRANCO:              ; Código ASCII <= 32
        OR      AH,AH           ; Ultimo caracter era Branco?
        JZ      @@LER_CARACTER  ; Se Sim então salta
        XOR     AL,AL           ; Caracter = '\0'
        STOSB                   ; Guarda caracter
        MOV     AH,AL           ; Ultimo caracter = caracter
        JMP     SHORT @@LER_CARACTER


  @@FIM_LC:                     ; Fim da linha de comando
        XOR     AL,AL
        OR      AH,AH           ; Ultimo caracter era Branco?
        JZ      @@FIM
        STOSB                   ; Guarda caracter

    @@FIM:
        MOV     BX,ES           ; Recupera DS
        MOV     DS,BX
        RET                     ; AL = Código de retorno
ENDP


IFDEF DEBUG
;-----------------------------------------------------------------------
UDATASEG
;-----------------------------------------------------------------------

LABEL FIM_DRIVER WORD

ENDIF

END
```