PC COMO TERMINAL DO ED680
(USO E IMPLEMENTAÇÃO)

por

Sílvia Delgado Olabarriaga

RP nº 131 AGOSTO 1990

Trabalho desenvolvido com apoio da FINEP
como parte das atividades do Grupo de Microeletrônica

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99
90.210 - Porto Alegre - RS - BRASIL
Telefone: (0512) 271999
Telex: (051) 2680 - CCUF BR
FAX: (0512) 244164
E-MAIL: silvia@sbu.ufrgs.anrs.br
PGCC@sbu.ufrgs.anrs.br

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - BRASIL

Editor: Ingrid E. S. J. Pôrto

Micro computadores - SBC
Equipamento: Entrada/saida
Emulação: Terminal

CNPq 1.03.04.00-2

UFRGS

Reitor: Prof. TUISKON DICK
Pró-Reitor de Pesquisa e Pós-Graduação: Prof. ABILIO BAETA NEVES
Coordenador do CPGCC: Prof. Ricardo A. da L. Reis
Comissão Coordenadora do CPGCC: Prof. Carlos Alberto Heuser
Prof. Clesio Saraiva dos Santos
Profa. Ingrid E.S.J. Pôrto
Prof. José Mauro V. de Castilho
Prof. Ricardo A. da L. Reis
Prof. Sergio Bampi

Bibliotecária CPGCC/CPD: Margarida Buchmann

SUMARIO

RESUMO

Este relatório descreve um conjunto de programas e bibliotecas que permitem o uso de um PC como terminal gráfico ou alfanumérico de um ED680 (sob o sistema EDIX), além de possibilitar trocas de arquivos entre os dois equipamentos.

PALAVRAS-CHAVE : emulação de terminal, interfaces gráficas, transferência de arquivos.

ABSTRACT

This report presents a group of programs and libraries which permit the use of a PClike microcomputer as a terminal of the ED680 machine. This terminal can be used in graphical or alphanumerical interfaces and to exchange files between these equipments.

KEY-WORDS : terminal emulation, graphical interfaces, file transfer.

1. INTRODUÇÃO

O projeto de ligação de um PC como terminal do ED680 surgiu da necessidade de utilização de interfaces gráficas em programas desenvolvidos e executados em sistemas operacionais compatíveis com UNIX. Além de capacidade gráfica, o terminal deveria possibilitar entrada e saída de dados alfanuméricos, bem como transferência de arquivos entre os dois equipamentos.

Com este objetivo, implementou-se o "TERMINAL": conjunto de ferramentas que possibilitam o uso de um PC como terminal gráfico, alfanumérico e meio magnético auxiliar de um ED680.

1.1 Funcionamento geral do TERMINAL

A conexão PC-ED680 é feita via RS-232, sendo o PC transformado em terminal através da execução de um programa sobre o DOS. A transmissão é feita a 4800 bps, 8 bits de dados, 1 stop bit e paridade ímpar, utilizando a porta de comunicações "COM1" do PC e uma tty qualquer do ED680.

Inicialmente o terminal funciona no modo alfanumérico, simplesmente transmitindo ao ED680 todos os caracteres digitados no teclado e exibindo no vídeo os caracteres recebidos do ED680. Assim, o funcionamento se assemelha ao de um terminal comum, exceto pelos caracteres de controle do vídeo (que não são interpretados pelo PC) e teclado (apenas as teclas ASCII são reconhecidas). Aplicativos executados sobre o EDIX podem acessar o terminal através de stdin e stdout.

A utilização de recursos gráficos envolve procedimentos especiais : é necessário usar uma biblioteca de funções que programam o terminal para modo gráfico. A biblioteca chama-se "libgraf.a" e permite basicamente:

a) transformação para modo gráfico/alfanumérico através do envio de caracteres especiais;

b) desenho de linhas, texto, retângulos e áreas em diversas cores e estilos.

A transferência de arquivos é feita através de dois utilitários: "tr600pc" (transmite do ED680 para o PC) e "re600pc" (recebe no ED680 um arquivo do PC). Ambos executam sobre o EDIX e enviam caracteres especiais que transformam o terminal para "modo transferência". A transmissão é feita em

blocos contendo um caractere de controle para detecção de erros. Qualquer erro causa o término anormal do utilitário.

2. USO DO TERMINAL

2.1 Ferramentas para modo alfanumérico

Neste modo, todos os caracteres digitados no teclado do PC são enviados ao ED680 e entram na stdin do processo ligado àquela tty. Apenas as teclas alfanuméricas e de símbolos especiais (bem como suas combinações com Ctrl e Sup) e Back Space (BS e "<-") são reconhecidas pelo terminal como caracteres que podem ser enviados ao ED680. As demais (setas, funções, etc.) são tratadas localmente:

F1	exibe informações sobre o uso do terminal.
F2	exibe o número de erros de recepção detectados.
F10	retorna ao DOS, enviando "DEL" ao ED680. (simula cancelamento de programa no EDIX)
Ctrl Break	envia "DEL" ao ED680, simulando cancelamento de programa por "Sup Elim".

As demais teclas especiais são ignoradas.

Durante a execução no modo alfanumérico, o terminal interpreta apenas comandos de "inicia modo gráfico" e "inicia modo de transferência", codificados como sequências de "escape" (ESC+comando). Outras sequências deste tipo destinadas a posicionar cursor, limpar tela, etc. não foram implementadas, o que restringe o uso do terminal a aplicativos que não utilizem formatação de tela.

O terminal é ativado no PC pela execução do arquivo "TERMINAL.BAT", que inicializa a porta de comunicações "COM1" conforme especificado em 1.1 usando o utilitário "MODE.COM". Só então o programa "TERM.EXE" é executado, iniciando a operação como terminal. Todos os arquivos mencionados devem estar no diretório corrente ou no "PATH" especificado pelo usuário.

2.2 Ferramentas para modo gráfico

Aplicativos do EDIX que precisem utilizar interface gráfica com o usuário devem chamar as rotinas da biblioteca "/usr/lib/libgraf.a". Estas rotinas permitem acesso à tela

gráfica do terminal, cujas características dependem da placa gráfica usada.

Em alguns casos (por exemplo CGA e EGA), a superfície usada para exibição de dados alfanuméricos é a mesma usada para gráficos; em outros (ex. ARTIST), existe um monitor adicional apenas para a exibição de gráficos. Entretanto, mesmo que as superfícies de exibição gráfica e alfanumérica não sejam sobrepostas, o acesso a elas é mutuamente exclusivo. Durante o modo gráfico, apenas funções da biblioteca podem ser usadas; tentativas de uso da stdout podem causar o mal funcionamento do terminal. O acesso à stdin, entretanto, pode ser feito durante o modo gráfico, porém o eco, neste caso, não é feito automaticamente (isto fica a cargo do aplicativo !).

2.2.1 Funções gráficas

As funções gráficas são a seguir apresentadas resumidamente. Para detalhes sobre o uso destas rotinas, seus parâmetros e valores retornados, consultar /JAC 88/. Os valores das coordenadas, cores e fator de texto são dependentes do dispositivo em uso (CGA, EGA ou ARTIST).

a) inic_graf()

Modifica o modo do terminal para gráfico através do envio de uma sequência de caracteres de controle, desativando a stdout. A tela é limpa e os atributos correntes da interface gráfica inicializados;

b) fim_graf()

Modifica o modo do terminal para alfanumérico através do envio de caracteres de controle, limpando a tela e reativando a stdout.

c) set_modo_acesso (modo)
```
char modo;                    /* valores de 0 a 3 */
```

Especifica o modo de acesso à placa gráfica :

0 = seta cor
1 = zera cor
2 = soma cor
3 = inverte cor

d) seta_t_lin (tipo)
```
int tipo;                /* máscara para tipo de linha */
```

Especifica tipo das linhas a serem desenhadas : a máscara indicada (0x0000 a 0xFFFF) é usada para desenhar linhas pontilhadas, cheias, tracejadas, etc.

e) seta_padrao_8X8 (pad)
```
char pad[8];
```

Especifica padrão para preenchimento de áreas retangulares. Cada byte do padrão é aplicado como máscara de uma linha do retângulo durante a pintura.

f) seta_fator_escrita (fator)
```
int fator;                    /* valores de 0 a 16 */
```

Especifica tamanho dos caracteres para exibição de texto.

g) des_texto (x, y, cor, texto)
```
int x, y;
char cor;
char *texto;
```

Desenha um string na tela na posição, cor e tamanho indicados.

h) limpa_tela_graf ()

Pinta a tela com preto.

i) des_retangulo (x1, y1, x2, y2, cor)
```
int x1, y1, x2, y2;
char cor;
```

Desenha o contorno de um retângulo de diagonal (x1,y1) a (x2,y2) na cor e tipo de linha especificados.

j) enche_retangulo (x1, y1, x2, y2, cor)
```
int x1, y1, x2, y2;
char cor;
```

Pinta o interior de um retângulo de diagonal (x1,y1) a (x2,y2) na cor e padrão de preenchimento especificados.

k) apaga_retangulo (x1, y1, x2, y2)
```
int x1, y1, x2, y2;
```

Pinta o interior de um retângulo de diagonal (x1,y1) a (x2,y2) em preto.

l) des_linha (x1, y1, x2, y2, cor)
```
int x1, y1, x2, y2;
```

```
char cor;
```

Desenha uma linha de extremidades em (x1,y1) e (x2,y2) na cor e tipo especificados.

2.2.2 Desenvolvimento de um aplicativo gráfico

Em primeiro lugar, o projetista deve estar ciente de qual dispositivo gráfico estará conectado ao PC :

- CGA : 640 X 200 X 2 cores
- EGA : 640 X 350 X 16 cores
- ARTIST : 1024 X 1024 X 16 cores

Os limites do sistema de coordenadas e o número de cores varia significativamente de um dispositivo para outro, fazendo-se necessário o planejamento cauteloso da forma como o aplicativo acessará o equipamento gráfico a fim de garantir portabilidade.

É importante salientar também que, ao usar o terminal no modo gráfico, o aplicativo não tem mais acesso à stdout e, portanto, toda a comunicação com o usuário (eco do teclado, mensagens de erro, etc.) deve ser feita via superfície gráfica de exibição. Esta restrição é séria, porém compatível com o funcionamento normal das placas EGA e CGA no MS-DOS.

A ligação do programa à biblioteca pode ser feita através do seguinte comando :

```
$ cc fonte.c -lgraf
```

onde "-lgraf" indica ao ligador que a biblioteca "/usr/lib/libgraf.a" deve ser anexada ao objeto.

Ainda uma observação fundamental: sempre que o programa terminar sua execução (de forma normal ou via "kill"), o terminal deve ser transformado em alfanumérico novamente, sob pena de se perder o acesso à tty. Assim, o programador deve prever a execução do utilitário "/usr/bin/.termalfa", que envia a sequência de caracteres de controle necessária para colocar o terminal no modo alfanumérico. Isto pode ser feito através do seguinte arquivo de comandos para o shell:

```
a = 'stty -g '"eof ^D"
trap "/usr/bin/.termalfa" ; stty \$a" 2
(chama aplicativo)
```

Na primeira linha, os parâmetros atuais da tty são salvos na variável "a". Na segunda linha, indica-se que o

utilitário i".termalfa" e stty devem ser executados em caso de cancelamento do programa, fazendo com que o terminal volte ao modo alfanumérico, com os parâmetros anteriormente salvos.

2.3 Ferramentas para transferência de arquivos

São dois os utilitários que permitem transferência de arquivos entre o ED680 e o PC: tr600pc e re600pc. Ambos executam sobre o EDIX e permitem transferência de arquivos binários ou de texto.

2.3.1 Transmissão para o PC

Sintaxe para chamada do utilitário:

$ tr600pc [-o] <arq.fonte> <arq.destino>

Parâmetros:

-o (opcional): indica que o arquivo a ser transmitido contém dados binários. A ausência deste parâmetro indica que o arquivo contém texto, sendo os caracteres CR, LF e fim de arquivo interpretados adequadamente.

<arq.fonte> : nome do arquivo no ED680 (obedece às regras de nomenclatura do EDIX).

<arq.destino>: nome do arquivo no PC. Pode conter especificação de unidade e diretórios, conforme regras do MS-DOS (no máximo 250 caracteres).

Execução:

Coloca o PC em modo "tranferência de arquivos" através do envio de caracteres de controle. A partir de então, os dois equipamentos se comunicam por meio de protocolo particular (ver seção 3.3), com as seguintes características:

- transmissão feita em blocos de 255 bytes, com um byte de verificação. O envio de um bloco só é iniciado após o anterior ter sido corretamente armazenado no disco.

- o programa termina de forma anormal após a detecção de qualquer erro e deve ser executado novamente pelo usuário para outra tentativa.

Durante a transmissão de arquivos, apenas as teclas F9, F10 e Ctrl Break são reconhecidas. Todas causam o término anormal do utilitário, sendo o arquivo destino removido do disco.

2.3.2 Recepção do ED680

Sintaxe para chamada do utilitário:

$ re600pc [-o] <arq.fonte> <arq.destino>

Parâmetros:

-o (opcional): indica que o arquivo a ser recebido contém dados binários. A ausência deste parâmetro indica que o arquivo contém texto, sendo os caracteres CR, LF e fim de arquivo interpretados adequadamente.

<arq.fonte>: nome do arquivo no PC. Pode conter especificação de unidade e diretórios, conforme regras do MS-DOS (no máximo 250 caracteres).

<arq.destino>: nome do arquivo no ED680 (obedece às regras de nomenclatura do EDIX).

Execução:

Ver seção 2.3.1.

2.3.3 Transferência de arquivos contendo texto

Arquivos de texto gerados no ED680 podem conter caracteres especiais para acentuação que seguem o padrão ABICOMP. Quando exibido em terminal que reconheça esta norma, o texto aparece acentuado; no entanto, isto não acontece num PC que simplesmente copia para a tela todos os caracteres, incluido os de controle, o que dificulta a leitura.

Assim sendo, é conveniente transformar para ASCII os arquivos que contém caracteres ABICOMP através do utilitário "conv":

$ conv <arq.ABI> <arq.ASC>

Este comando converte adequadamente caracteres como "ç" para "c", "ã" para "a", "é" para "e", etc., tornando o texto legível num PC.

2.3.4 Observações gerais

Outras teclas além de F9, F10 e Ctrl Break não devem ser digitadas, pois podem introduzir erros na transferência. Se, por algum motivo, o terminal parar de funcionar (bug!), basta teclar F10 e chamar novamente o comando "TERMINAL". Desta forma, o utilitário que executa no ED680 é cancelado e o software do terminal reinicializado; a seção, entretanto, continua aberta!!

2.4 Configuração do terminal

A ligação física do PC ao ED680 tem as seguintes características:

- 4800 bps
- 8 bits de dados
- 1 stop bit
- paridade ímpar

A configuração da porta serial do PC (COM1) é feita através do seguinte comando, contido no arquivo "TERMINAL.BAT":

```
MODE COM1:4800,o,8,1
```

No ED680, a tty deve ser incluída na arquivo "/etc/inittab" (ver /EDI 87/) através de uma linha semelhante à seguinte:

```
TY4:2:respawn:/etc/getty -h tty4 7
```

O número "7" indica o índice na tabela de configuração contida "/etc/gettydefs", que descreve as características da conexão física:

```
#tabela 7 --> 4800, 8, odd - para PC

        7 # ICRNLIXON IXANY ONLCR OPOST B4800 CREAD
CS8PARENB PARODD ECHOK # BRKINT ISTRIP IXON IXANY ICRNL
INPCK ONLCR OPOST TAB3 B4800 CREAD CS8 PARENB PARODD ECHO
ECHOK ECHOE ICANON ISIG PAGINA #

        \007? - Identificação : # 7
```

3. IMPLEMENTAÇÃO

Neste capítulo serão descritas as estruturas de dados e algoritmos usados na implementação do terminal. A parte executada no ED680 consiste basicamnte nos utilitários de transferência de arquivos e na biblioteca gráfica. No PC o software é composto de módulos independentes para cada função: emulação de terminal alfanumérico simples, transferência de arquivos e emulação de um terminal gráfico.

3.1 Estrutura interna do terminal no PC

A parte do terminal que executa no PC corresponde ao arquivo "TERM.EXE", gerado a partir da ligação dos seguintes módulos:

a) TERMPRINC.C: módulo escrito em C que contém rotinas para emulação do terminal alfanumérico e reconhecimento das sequências de caracteres de controle;

b) TERMCOM.C: módulo escrito em C que contém procedimentos para transferência de arquivos;

c) TERMGRAF.C: módulo escrito em C que contém rotinas para interpretação de comandos gráficos e seus parâmetros;

d) CGA.C ou ART.C ou EGA.C: módulo escrito em C que contém os procedimentos para acesso a uma placa gráfica específica;

e) TERMASM.ASM: módulo escrito em assembler do 8086 que contém rotinas para acessar o sistema de interrupções do PC.

Os módulos TERMPRINC TERMCOM e TERMGRAF utilizam constantes definidas no arquivo "TERMDEFS.H"; TERMCOM utiliza, também, o arquivo "COMPCDEFS.H". CGA, ART ou EGA utilizam arquivos "placa.h", "placa.glb" e "placa.frw" (placa corresponde ao nome do adaptador gráfico em questão).

Durante a execução, o arquivo "placa.drv" deve estar presente no diretório corrente, pois é carregado pelo terminal. Este arquivo contém as rotinas gráficas básicas (em assembler).

O compilador C usado foi Microsoft versão 4.0 (modelo de memória "small", sendo a compilação, montagem e ligação dos módulos feitas através do programa "REFAZ.BAT".

Cada módulo será descrito a seguir, com ênfase em suas estruturas de dados e algoritmos mais importantes. Maiores detalhes sobre parâmetros, valores retornados e funcionamento das rotinas podem ser obtidos no código fonte.

3.1.1 TERMPRINC.C

a) buffer de recepção (buf): o buffer é circular, sendo gerenciado através das variáveis abaixo:

phead: ponteiro para início do buffer (primeiro caractere a ser retirado)
ptail: ponteiro para fim do buffer (onde inserir caracteres)
nbuf: número de caracteres armazenados no buffer

Este buffer é preenchido pela rotina de atendimento de interrupções da RS232 ("cintcom1") e consumido no laço principal do programa ("retira"). O primeiro caractere do buffer pode ser consultado através da rotina "consulta".

b) número de erros: a variável "nerros" armazena o número de erros detectados na recepção de caracteres (paridade, overrun, etc.). Pode ser consultada pelo usuário através da tecla F2, quando o terminal está em modo alfanumérico;

c) tratamento de interrupções de RS232: a recepção de caracteres é feita via interrupção de "buffer full", gerada pela 8255. As interrupções são atendidas inicialmente pela rotina "asintcom1" (TERMASM) e depois por "cintcom1". Essa rotina tem que funcionar sem verificação automática de stack overflow, pois a pilha é mudada pela rotina em assembler para uma área interna (ver seção 3.1.4). O mecanismo usado foi baseado em /HUN 85/ e /BIG 86/.

d) tratamento especial da tecla "Ctrl Break": o significado habitual desta tecla no DOS é o de encerrar a execução do programa, mas foi alterado no terminal. Para isto, as interrupções geradas por ela no BIOS e no DOS são desprezadas através da modificação das respectivas rotinas de atendimento. Estas foram substituídas por simples "RETI", o que permite que o código de Ctrl Break possa ser obtido pela rotina que lê o teclado e interpretado conforme o novo significado (emulação de Sup Elim);

e) gerenciamento de interrupções em geral: a rotina "prog_int_vect" salva os valores do vetor de interrupção correspondente a COM1 (interface RS232), Ctrl Break do BIOS e do DOS. Depois reprograma-os, fazendo com que apontem

rotinas do terminal. Antes de retornar ao DOS, o terminal restaura os valores originais através da rotina "rest_int_vect";

f) leitura do teclado: a rotina "lecomteclado" usa a função "letecla" (ver seção 3.1.4) para ler o código (ASCII ou iscan). Todos os códigos ASCII são reconhecidos e enviados ao ED680. As teclas especiais (código ASCII = 0 e scan diferente de 0) são interpretadas localmente da forma apresentada no capítulo 3. As teclas especiais sem significado para o terminal são desprezadas.

g) recepção de caracteres vindos pela RS232: a rotina "rec_carac" consome caracteres do buffer de receção e interpreta- os. Caracteres ASCII diferentes de ESC (1BH) são simplesmente ecoados no vídeo. Se um caractere ESC for recebido, a rotina espera o próximo para determinar o tipo de operação:

- "T": muda para modo "transferência de arquivos"
- "G": muda para modo gráfico

Outras sequências de escape são desprezadas.

h) "limpa_buf": esta rotina é usada pelos outros módulos para esvaziar o buffer de recepção quando há cancelamento do programa durante o modo gráfico ou de transferência de arquivos. Todos os caracteres são consumidos até que a sequência ESC + "A" seja recebida, indicando a volta ao modo alfanumérico.

3.1.2 TERMCOM.C

a) recepção de arquivos: feita pela rotina "rec_arq", em blocos de até 255 bytes, obedecendo protocolo descrito na seção 3.3. A qualquer instante a transferência pode ser abortada pela detecção de um erro ou pela recepção da sequência de controle que volta ao modo alfanumérico;

b) transmissão de arquivos: feita pela rotina "trans_arq" nos mesmos moldes da recepção. Os bytes de texto enviados ao ED680 são codificados em dois caracteres (cada um é formado por um nibble somado a 30H). Isto impede sua interpretação como caracteres de controle pelo EDIX. Esta codificação é feita pela rotina "trans_carac", que envia em primeiro lugar o nibble mais significativo;

c) volta ao modo alfanumérico: pode acontecer por fim OK da transferência, cancelamento do programa via "Ctrl Break" ou pela recepção de uma sequência de controle ESC +

'A'. Em qualquer destas situações a sequência de controle é necessária para que o terminal volte ao modo alfanumérico.

3.1.3 TERMGRAF.C

a) funcionamento geral: uma vez em modo gráfico, o terminal passa a interpretar os caracteres recebidos como comandos gráficos. A execução de cada comando é feita por uma rotina independente, chamada através de uma tabela (tab_com) que relaciona o byte recebido ao procedimento correspondente. Os comandos de maneira geral esperam a chegada dos parâmetros e desviam para a rotina adequada. Os parâmetros são codificados em binário (quando têm 16 bits, primeiro vem o byte mais significativo). As rotinas gráficas são descritas em detalhes em /JAC 88/;

b) volta ao modo alfanumérico: pode acontecer por recepção do comando "fimgraf", por cancelamento do programa (Ctrl Break) ou pela recepção da sequência ESC + 'A'.

3.1.4 TERMASM.ASM

a) "asintbreak": rotina de atendimento da interrupção gera da quando Ctrl Break é teclado. Simplesmente executa RETI, fazen do com que o sistema gere o seguinte código para esta tecla : ASCII = 0 , SCAN = 0;

b) "asintcom1": rotina de atendimento de interrupções da interface serial, descrita em /HUN 85/;

c) "cominit": inicializa o ambiente de interrupções da interface serial, seguindo descrição de /BIG 86/;

d) "habint" e "desabint": rotinas que permitem habilitar a desabilitar o atendimento de interrupções externas pelo 8086. São usadas para controle de acesso exclusivo às estruturas de dados do buffer de recepção;

e) "letecla": lê o código de uma tecla através do BIOS. Inicialmente verifica se algo foi digitado, lê o código e analisa-o, indicando se é tecla ASCII ou especial.

3.2 Estrutura interna do TERMINAL no ED680

A parte do terminal que executa no ED680 é composta de dois utilitários (re600pc e tr600pc) e uma biblioteca de funções (libgraf.a).

re600pc é um programa escrito em "shell" que chama os utilitários ".re600pc" e ".termalfa". tr600pc também é um programa escrito em "shell" e chama os utilitários ".tr600pc" e ".termalfa".

Estes utilitários e a biblioteca são gerados pelo programa "criaterminal" e devem ser copiado para os diretórios "/usr/bin" e "/usr/lib" pelo programa "copiaterminal", em modo super-usuário.

3.2.1 ".termalfa"

Este utilitário é gerado a partir da compilação de "termalfa.c" e tem por objetivo enviar ao PC a sequência de caracteres de controle que põe o terminal no modo alfanumérico. Sua execução é necessária quando os utilitários de transferência de arquivos ou programas que usam a biblioteca gráfica são cancelados (ver seção 2.2.2).

3.2.2 ".tr600pc"

Utilitário responsável pela transferência de arquivos do ED680 para o PC. É gerado a partir de dois módulos:

a) com600pc.c: contém rotinas básicas para transferência de mensagens de um equipamento a outro (ver seção 3.2.4);

b) tr600pc: contém rotinas para transferência do arquivo propriamente dito. Este módulo é bastante simples e basicamente implementa o protocolo descrito em 3.3.

Os dois módulos incluem em seu código o arquivo "comedefs.h", que contém a definição dos caracteres de controle, códigos de erro retornados pelas funções, comandos enviados ao PÇ mensagens de erro e macros para envio/recepção de caracteres.

3.2.3 ".re600pc"

Utilitário que transfere arquivos do PC para o ED680. É composto por dois módulos:

a) com600pc.c: contém rotinas básicas para transferência de mensagens de um equipamento a outro (ver seção 3.2.4);

b) re600pc: contém rotinas para transferência do arquivo propriamente dito. Basicamente implementa o protocolo descrito em 3.3.

Os dois módulos incluem em seu código o arquivo "comedefs.h" (ver seção 3.2.2).

3.2.4 com600pc.c

a) "inic_com": programa a tty para funcionamento sem interpretação de caracteres de controle e envia sequência para início do modo "transferência de arquivos" (ESC + "T"). Os parâmetros correntes da tty são salvos para serem restaurados no fim da execução (ver item b). A tty é reprogramada da seguinte forma:

- desabilita processamento de caracteres especiais de saída.
- deixa bit 7 dos caracteres recebidos.
- desabilita protocolo XON-XOFF na entrada
- habilita envio de XOFF para controlar fluxo do PC para ED680.
- desabilita interpretação de caracteres especiais na entrada.
- desabilita eco automático.
- desliga controle de página.
- indica que a leitura via "read" deve retornar mesmo que não existam caracteres no buffer de entrada.

OBS: os valores usados na programação são descritos em /EDI 87/, no item tty(4);

b) "fim_com": envia sequência de controle que retorna o terminal ao modo alfanumérico e restaura a programação original da tty;

c) "trans_msg" , "rec_msg": transmite/recebe uma mensagem de acordo com o protocolo descrito em 3.3.1;

d) "erro": rotina utilitária que exibe mensagem de erros no formato padrão destes programas.

3.2.5 "libgraf.a"

A biblioteca gráfica é gerada a partir da compilação do arquivo "graf600pc.c", através dos seguintes procedimentos executados pelo arquivo "criaterminal":

```
$ cc -c graf600pc.c
```

```
$ ar r libgraf.a graf600pc.o
```

"graf600pc.c" inclui o arquivo "grafdefs.h", que contém definições dos comandos gráficos e macros para transmissão de caracteres ao PC.

O módulo "graf600pc.c" contém todas as funções gráficas, que basicamente enviam ao PC seu código e os parâmetros recebidos. A seguir, apresentamos todas as funções e a codificação dos parâmetros de cada uma delas:

a) "inic_graf": muda o terminal para o modo gráfico, enviando a sequência de caracteres de controle (ESC + "G"). Reprograma a tty conforme descrito em 3.2.4;

b) "fim_graf": muda o terminal para o modo alfanumérico e restaura a programação original da tty;

c) "limpa_tela_graf": apenas o código da função é enviado para o PC, pois não tem parâmetros;

d) "seta_modo_acesso": envia o código da rotina e um byte contendo o parâmetro;

e) "set_t_lin" e "seta_fator_escrita": enviam código da função e depois dois bytes contendo o parâmetros (byte mais significativo primeiro);

f) "seta_padrao_8x8": envia código da função e oito bytes com o padrão;

g) "des_texto": envia código da função, dois bytes para coordenada x, dois para y, um byte para cor, um para tamanho do string e o texto a ser desenhado (sem null);

h) "des_retangulo", "enche_retangulo" e "des_linha": enviam código da função, dois bytes para cada coordenada (x1, y1, x2, y2) e um byte para cor;

i) "apaga_retangulo" : envia código da função e dois bytes para cada coordenada (x1, y1, x2, y2).

3.3 Protocolo para transferência de arquivos

A comunicação entre o PC e o ED680 para transferência de arquivos é feita em dois níveis:

a) nível de protocolo: responsável pela transferência de mensagens de um equipamento a outro;

b) nível de aplicativo: transfere arquivos de um equipamento a outro.

3.3.1 Nível de protocolo

As rotinas que implementam este nível da comunicação estão contidas no módulo "com600pc.c" e têm por finalidade garantir que uma mensagem seja corretamente transferida de um equipamento a outro.

O formato de uma mensagem é o seguinte:

STX
tamanho do texto (1 byte)
caracteres do texto (em 8 bits)
ETX
caractere para verificação, calculado pela soma de todos os demais (de STX a EXT)

Existe ainda uma mensagem de formato especial (apenas EOT) que indica o fim da transmissão de um conjunto de mensagens.

O receptor de uma mensagem sempre deve responder com "ACK" quando a recepção tiver sido feita com sucesso (todos os caracteres foram recebidos e o byte de controle foi OK).

OBSERVAÇÕES:

a) cuidados especiais são necessários para o cálculo do caractere de controle: este deve ser armazenado em 16 bits, mas apenas o byte inferior deve ser transmitido;

b) não há detecção de erros de "timeout" no PC (apenas no ED680). Este tipo de erro é detectado quando um caractere é esperado por tempo demasiadamente longo.

3.3.2 Nível de aplicativo

Neste nível são tomadas providências para garantir que o arquivo seja corretamente transmitido de um equipamento a outro.

a) protocolo de transmissão (do ED680 para PC) - contido no arquivo "tr600pc.c":

- envia mensagem contendo
  - comando (1B) : "transmite ASCII" ou

"transmite binario"
nome do arquivo destino no PC
(tamanho string = tam mensagem-1)

- espera mensagem do PC indicando se o arquivo destino foi aberto com sucesso ou não. Essa mensagem tem apenas 1 byte:

"1" = abertura OK
diferente de "1" = abertura sem sucesso

- fica em loop até encontrar o fim do arquivo, fazendo:
  - lê um bloco de 255 bytes do arquivo fonte
  - envia mensagem com o bloco lido
  - espera confirmação de gravação no PC, através de mensagem de 1 byte. A ausência de resposta indica erro na gravação.

- envia EOT (fim do arquivo)

- espera confirmação do fechamento OK do arquivo destino, através de uma mesagem de 1 byte. A ausência de mensagem indica erro no fechamento do arquivo.

b) protocolo de recepção (do PC para o ED680) - contido no arquivo "re600pc.c":

- envia mensagem contendo
  comando (1B): "recebe ASCII" ou
  "recebe binario"
  nome do arquivo destino no PC
  (tamanho string = tam mensagem-1)

- espera mensagem do PC indicando se o arquivo fonte foi aberto com sucesso ou não. Esta mensagem tem apenas 1 byte:

"1" = abertura OK
diferente de "1" = abertura sem sucesso

- fica em loop até receber EOT, fazendo:
  - espera mensagem contendo um bloco do arquivo (até 255 bytes)
  - grava o bloco no arquivo destino
  - envia mensagem de um byte ao PC indicando se a gravação foi feita com sucesso ou não. Se houver erro, o aplicativo aborta, retornando o terminal ao modo alfanumérico.

- fecha o arquivo destino

- envia mensagem de um byte ao PC indicando se o fechamento do arquivo foi feito com sucesso ou não. Em caso de erro, aborta e muda o terminal para modo alfanumérico.

BIBLIOGRAFIA

/BIG 86/ BIGGERSTAFF T. Systems software tools. Englewood-Cliffs, Prentice-Hall, 1986.

/EDI 87/ EDISA. Manual de referência do EDIX5. Gravatai, 1987.

/JAC 88/ JACOBI R. PGE - Pacote gráfico do EMHIR. Porto Alegre, PGCC da UFRGS, 1988. (a ser publicado)

/HUN 85/ HUNT W. J. TTY : a terminal emulation program. In: _____. The C toolbox. Reading, Addison-Wesley, 1985. p.327-64.