UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE FÍSICA
CAIXA POSTAL 20516
01498-970 SÃO PAULO - SP
BRASIL

PUBLICAÇÕES

IFUSP/P-1003

TRANSFERÊNCIA DE DADOS ENTRE MICROCOMPUTADORES APPLE E IBM-PC

W.M. Gonçalves & J.C. Sartorelli
Instituto de Física, Universidade de São Paulo

Agosto/1992

Transferencia de dados entre microcomputadores Apple e IBM-PC

W.M.Gonçalves & J.C.Sartorelli

Instituto de Fisica

Universidade de São Paulo

C.P. 20516

01498 São Paulo-SP

Bitnet: GONCALVES@IF.USP.ANSP.BR

SARTORELLI@IF.USP.ANSP.BR

Resumo

Desenvolvemos um algorítmo para a transferência de arquivos binários e do tipo texto do Apple para o IBM-PC.

A conexão entre os dois microcomputadores é feita através da porta de jogos do Apple e da interface paralela do IBM-PC, com um cabo de 4 vias.

Os programas de controle foram desenvolvidos em BASIC e linguagem de máquina 6502 para o Apple e em Pascal para o IBM-PC.

Abstract

We have developed an algorithm to send binary and text files from the Apple to the IBM-PC microcomputer.

The two microcomputers are connected via the Apple game port and the IBM-PC parallel interface with a four wire cable.

The control programs were written in BASIC and in the 6502 machine language for the Apple, and in Pascal for the PC.

1. Introdução

Os microcomputadores Apple ainda são empregados em muitos laboratórios de pesquisa e didáticos no controle de experimentos e na aquisição de dados [1,2,3,4,5]. Muitos laboratórios dispõem de placas de hardware para Apple, como conversores analógico/digitais e digitais/analógicos, placa de relógio, placas paralelas e seriais, expansões de memória, que ainda podem ser reaproveitadas.

Entretando, no Apple a análise e tratamento de dados podem se tornar trabalhosos e lentos, pois em comparação a um IBM-PC, o Apple não tem maiores recursos de memória, velocidade de processamento, boas saídas gráficas. Além disso, arquivos de dados gravados em disquete pelo Apple não podem ser lidos pela unidade de disco do IBM-PC e vice-versa.

Neste trabalho apresentamos uma solução para a transferência de arquivos do Apple para o PC através de um sistema simples e de baixo custo que usa interfaces já presentes nos dois micros (a porta de jogos do Apple e a interface paralela padrão do IBM-PC), conectadas através de um cabo de 4 vias.

2. Tipos de arquivos

A transmissão de cada byte de arquivos de dados em disco do Apple para um arquivo no IBM-PC é feita bit-a-bit.

Esses bytes de informação devem ser previamente carregados para a memória do Apple e tratados convenientemente de acordo com o tipo de arquivo (binário ou texto) que se deseja transferir [6,7].

Os arquivos binários são carregados inteiramente do disco para uma área de memória ("buffer") através do comando "BLOAD" do DOS do Apple, por exemplo:

```
BLOAD DADOS, A8192
```

onde um arquivo binário de nome "DADOS" é lido para a memória do Apple à partir do endereço 8192. Os arquivos binários são gravados pelo sistema operacional com um cabeçalho que contém o endereço inicial de carga e o comprimento do arquivo. Programas em linguagem de máquina, telas gráficas, ou uma área de memória do micro ("buffer") são normalmente gravados em arquivos binários.

Os arquivos do tipo texto também são utilizados para gravar seqüências de bytes, sendo cada byte correspondente a um caracter alfanumérico ou de controle pela tabela de codigos ASCII. O byte 26 ($1A) é usado para finalizar o arquivo texto, uma vez que este tipo de arquivo não contém cabeçalho com informações sobre o endereço inicial de carga na memória e o seu comprimento. O sistema operacional (DOS 3.3) não dispõe de um comando análogo ao BLOAD para a carga de arquivos texto para um "buffer" na memória. Os arquivos a serem lidos devem ser abertos com a seqüência de comandos:


```
PRINT CHR$(4);"OPEN"; NOME$
PRINT CHR$(4);"READ"; NOME$
```

onde NOME$ é uma variável contendo o nome do arquivo texto. Cada caracter do arquivo é lido com a instrução:

```
GET A$
```

onde A$ é uma variável do tipo "string". O byte correspondente é obtido através da instrução:

```
BYTE=ASC(A$)
```

sendo BYTE uma variável inteira. O arquivo texto é lido completamente repetindo-se o comando GET A$ em um laço de repetição até que o byte de fim de arquivo (=26) seja encontrado.

No IBM-PC, independentemente do tipo de arquivo Apple, os bytes que vão sendo recebidos através do processo de transmissão são armazenados em um "buffer" na memória do micro. O "buffer" é gravado em disco (rígido ou flexível) ao final da transmissão.

3. Interface

No Apple, utilizamos como interface a porta de jogos[6,8], que é um soquete de 16 pinos montado na placa principal do micro, com 4 saídas TTL (anunciadores de AN0 a AN3) e 3 entradas TTL (PB's de PB0 a PB2).

A cada anunciador estão associados dois endereços de controle ("soft-switches"), um para colocá-lo em estado "ON" e outro em estado "OFF". A "soft-switch" é ativada

através de uma leitura do endereço. Por exemplo, para o AN0, temos:

	Pino	"Soft-switches"
AN0	15	$C059 ("ON")
		$C058 ("OFF")

O estado de uma entrada é determinado pelo estado do bit de sinal (sétimo bit) do conteúdo da sua "soft-switch". Se esse conteúdo for >= 128 (7^{o} bit = 1) o nível de entrada é "1" e se < 128 (7^{o} bit = 0) o nível de entrada é "0". Por exemplo, para ler a entrada PB0 ("soft-switch" $C061=49249), executamos o comando:

A= PEEK(49249)

e verificamos o conteúdo da variável A como descrito acima.

No IBM-PC[9,10] utilizamos a interface paralela (LPT1) que tem várias entradas e saídas TTL, porém com um sistema de controle diferente. Um único endereço ("port") pode controlar várias entradas (ou várias saídas). Por exemplo, o porto de saída da paralela $378:

Bit	7	6	5	4	3	2	1	0
Pin	9	8	7	6	5	4	3	2

onde o estado de cada saída (Pin de 2 a 9) é dado pelo valor dos bits correspondentes (Bit de 0 a 7). Com a instrução (em Pascal) de escrita ao porto, por exemplo,

port[$378]:=$06

colocamos as saídas do pino 5 e 3 em estado "1" e as demais em estado "0". A associação deste fato com a tabela acima é verificado através da representação binária de $06:

$06 = 00001010

Um porto de entrada opera de modo análogo, sendo necessária uma operação de leitura. Por exemplo, o porto de entrada $379:

Bit	7	6	5	4	3	2	1	0
Pin	11	10	12	13	15	-	-	-

é lido com o comando em Pascal:

a:=port[$379]

As conexões entre os dois microcomputadores estão descritas na figura 1.

FIGURA 1

O anunciador AN1 (linha de dados) é usado para informar ao PC os estados dos diferentes bits ("1" ou "0") do byte a ser transmitido. Os bits vão sendo lidos sucessivamente da direita para a esquerda e o byte é remontado no PC.

A sincronização das atividades de envio e recepção é feita pelo anunciador AN0 (linha de "strobe") e pela entrada PB0 (linha de "acknowledge"):

	OFF	ON
AN0	Linha de dados não foi preparada pelo Apple	Linha de dados pronta para ser lida pelo PC.
PB0	PC ainda não completou a leitura da linha de dados.	Operação de leitura da linha de dados foi completada.

4. Rotinas

No Apple,a rotina em linguagem de máquina[11] divide-se em duas subrotinas. A primeira subrotina transmiste um buffer da memória contendo N bytes e é usada para o envio de arquivos binários. Por exemplo:

```
POKE 225,L1: POKE 226,H1: POKE 227,LB: POKE 228,HB
CALL 768
```

onde L1 (LB) é o "low-byte" e H1 (HB) é o "high-byte" do endereço do início (fim) do "buffer". Esta subrotina em linguagem de máquina transmite seqüencialmente todos os bytes do "buffer" após a instrução CALL e só retorna ao BASIC depois de ter transmitido todo o "buffer".

A segunda subrotina é usada para a transmissão de um byte a cada instrução CALL:

```
POKE 229,BYTE
CALL 859
```

onde BYTE é a variável inteira que contém o valor do byte a ser transmitido. Com esta subrotina são enviadas "strings", tais como o nome do arquivo a ser transferido, comprimento do arquivo, ou caracteres individuais, como caracteres de controle (tipo de arquivo, caracter de fim de arquivo, etc). É também com esta subrotina que as strings A$ lidas de arquivos do tipo texto são transmitidas.

Na listagem 1 temos o programa AP-PC.BAS para o Apple. As instruções e argumentos da linguagem de máquina estão codificados em comandos DATA (linhas de 50 a 80,

listagem 1) e são introduzidos através de instruções POKE (linhas 90 e 100, listagem 1) pelo próprio programa BASIC, evitando-se o uso de um programa montador para a linguagem de máquina.

Na listagem 2 temos o programa AP-PC.PAS escrito em Turbo-Pascal para o PC, que deve ser compilado para o disco, gerando a versão auto-executável AP-PC.EXE.

5. Execução

Deve ser executado em primeiro lugar o programa do Apple, e em seguida o do PC (os arquivos no PC serão gravados no mesmo disco de chamada do programa). O processo é então comandado inteiramente do Apple, sendo as informações de controle transmitidas ao PC.

No Apple, as seguintes mensagens auto-explicativas são apresentadas sucessivamente:

```
-->        USA O DAVID-DOS?  (S/N)
```

O sistema operacional DAVID-DOS é uma modificação do sistema operacional de disco DOS 3.3, cerca de 3 vezes mais rápido, e dispõe de comandos de carga de arquivos texto com apenas um comando TLOAD, de funcionamento análogo ao BLOAD para arquivos binários. Com o DAVID-DOS, arquivos texto são transmitidos usando-se a subrotina de transmissão de "buffers", com grande aumento da velocidade de

transmissão em bytes/segundo (bps), conforme podemos verificar na tabela 1.

Tabela 1.

--> NOME DO ARQUIVO (?=CAT) >

Deve ser fornecido o nome do arquivo no Apple. A entrada do caracter "?" fornece o catálogo do disco do Apple.

Caso o nome do arquivo no Apple não seja um nome válido para um arquivo no PC, ou seja, um nome com até 8 caracteres alfanuméricos, excetuando-se o ponto ("."), é apresentada a mensagem:

--> NOME NO IBM-PC >

devendo ser fornecido um nome nas especificações acima.

No PC, os arquivos texto receberão a extensão ".TXT" e os binários a extensão ".BIN".

--> TIPO: T)EXTO B)INARIO (?=CAT) >

Deve ser fornecido o tipo do arquivo (T ou B). Esta informação pode ser encontrada no catálogo do disco. Os arquivos texto são marcados com "T" e os binários com "B".

À medida que é realizada a transmissão, o número de bytes transferidos é apresentado na tela do PC.

--> OUTRO ARQUIVO (S/N) ? >

Novos arquivos podem ser transferidos (opção "S") ou a sessão é encerrada nos dois micros (opção "N").

6. Comentários e conclusão

No caso de arquivos gerados em CP/M ou ProDos, use os respectivos programas de conversão para o DOS 3.3.

Arquivos gerados com processadores de texto, com formatação e acentuação feitas com caracteres de controle não são necessariamente interpretados no PC na sua forma original.

Arquivos binários transmitidos ao PC, porém gerados por processadores de texto, devem ser reprocessados com o programa "BIN2TXT.PAS", listagem 3.

No caso de travamento, ou de interrupção dos programas devido a algum erro, os dois micros devem ser reinicializados.

O tamanho de arquivos binários é limitado a 30 Kbytes para o DOS 3.3 e a 40 Kbytes para o David-DOS, aproximadamente.

Se o PC tiver mais de uma interface paralela, a correta implementação deve ser feita testando-se cada uma das interfaces.

O sistema de transmissão de dados apresentado, simples e de baixo custo, cuja técnica também pode ser empregada entre outros tipos de microcomputadores, mostrou-se plenamente satisfatório.

TABELAS

	Arq. Texto (bps)	Arq. Binários (bps)
DOS 3.3	41	493
David-DOS	840	887

Tabela 1: Comparação entre as velocidades de transferência (carga + transmissão) em bps de arquivos texto e binários dos sistemas operacionais de disco DOS 3.3 e David-DOS.

DESCRIÇÃO DAS FIGURAS

Figura 1: Conexões entre a porta de jogos do Apple e a interface paralela do IBM-PC para a transmissão de dados.

Referências

1. Laboratório de Estrutura da Matéria, IFUSP, São Paulo.

2. G.H.Guedes, Dissertação de Mestrado, IFUSP, São Paulo, 1990.

3. M.T.A.Orlandi, Dissertação de Mestrado, IPEN, São Paulo, 1991.

4. M.L.Siqueira, R.E.Rapp & F.A.B.Chaves, "On line data acquisition in heat capacity measurements", Rev. Fis. Apl. Instr. **5**, n. 4, 401 (1990).

5. L.Mussio, J.Castiglioni & W.Diano, "Automatización de un equipo termogravimétrico", **2**, n. 2, 118 (1987).

6. "Apple IIe - Reference Manual", Apple Computer, Inc., California, EUA, 1982.

7. R.M.Watanabe, "Guia do Programador DOS", Ed. Aleph, São Paulo, 1986.

8. J.W.Coffron, "The Apple Connection", Sybex Inc., Berkeley, EUA, 1982.

9. "Technical Reference of the IBM-PC", International Business Corporation, 1983.

10. J.P.Santos & E.Raymundi Jr., "Programando em Assembler 8086/8088", Ed. McGraw-Hill, São Paulo, 1989.

11. B.W.Schoen, "Assembly 6502", Ed. Aleph, São Paulo, 1985.

Listagem 1 - Programa AP-PC.BAS para o Apple

```
10  REM   Program AP-PC.BAS
20  ONERR  GOTO 520
30  POKE 34,0: HOME : PRINT : PRINT "Transferencia de
    arquivos Apple / IBM-PC"
40  PRINT "W.M.Goncalves & J.C.Sartorelli - IFUSP": PRINT :
    POKE 34,5: PRINT
50  DATA   76,43,3,165,225,24,105,1,133,225,165,226,105,
    0,133,226,165,227 ,56,229,225,165,228,229,226,144,1,
    96,104,104,96,70,229
60  DATA   144,4,173,91,192,96,173,90,192,96,234,173,89,
    192,160,0,177,225,133,229,162,8,32,31,3,173,88,192,
    173,97
70  DATA   192,48,251,173,89,192,173,97,192,16,251,202,
    208,234,32,3,3,76,47,3
80  DATA   234,162,8,32,31,3,173,88,192,173,97,192,48,251,
    173,89,192,173,97,192,16,251,202,208,234,173,89,192,96
90  RESTORE : FOR I = 768 TO 850: READ J: POKE I,J: NEXT
100 FOR I = 859 TO 887: READ J: POKE I,J: NEXT :
    SL =- 16295: PRINT : PRINT
110 PRINT "RODE O PROGRAMA AP-PC NO IBM-PC > "; CHR$ (7):
    A =  PEEK (SL): GET A$: PRINT
120 INPUT "USA O DAVID-DOS? (S/N) > ";DO$: PRINT
130 INPUT "NOME DO ARQUIVO (? = CAT.) > ";N$
140 IF N$ = "?" THEN  PRINT  CHR$ (4)"CATALOG": PRINT :
    GOTO 130
150 N =  LEN (N$): IF N > 8 THEN  GOTO 190
160 FOR I = 1 TO N: IF  MID$ (N$,I,1) = "." THEN  GOTO 190
170 NEXT
180 NP$ = N$: GOTO 230
190 INPUT "NOME NO IBM-PC > ";NP$
200 NP =  LEN (NP$): IF NP > 8 THEN  GOTO 190
210 FOR I = 1 TO NP: IF  MID$ (NP$,I,1) = "." THEN  GOTO
    190
220 NEXT
230 A$ = NP$: GOSUB 430
240 INPUT "TIPO (T)exto / (B)inario (? = CAT.) > ";T$
250 IF T$ = "?" THEN  PRINT  CHR$ (4)"CATALOG": PRINT :
    GOTO 240
260 IF (T$ <  > "T") AND (T$ <  > "B") THEN  GOTO 240
270 A$ = T$: GOSUB 400
280 IF (T$ = "T") AND (DO$ = "S") THEN
    PRINT CHR$(4);"TLOAD ";N$;",A$2000": GOTO 310
290 IF (T$ = "T") AND (DO$ <  > "S") THEN  GOSUB 450:
    GOTO 380
300 IF T$ = "B" THEN  PRINT  CHR$ (4);"BLOAD ";N$;",A$2000"
310 IF DO$ = "S" THEN L =  PEEK (48992) + 256 *
    PEEK (48993): GOTO 330
320 L =  PEEK (43616) + 256 *  PEEK (43617)
330 EC = 8192
340 A$ =  STR$ (L): GOSUB 430
```

```
350 HB =  INT ((EC + L) / 256):LB = EC + L - 256 * HB
360 POKE 225,0: POKE 226,32: POKE 227,LB: POKE 228,HB
370 CALL 768
380 PRINT : INPUT "OUTRO ARQUIVO (S/N)? ";A$: GOSUB 410:
    IF (A$ = "S") OR (A$ = "s") THEN PRINT:PRINT:GOTO 130
390 END
400 REM  SEND CHAR
410 A =  PEEK (SL):A =  ASC (A$): POKE 229,A: CALL 859:
    A =  PEEK (SL): RETURN
420 REM  SEND STRING
430 LL =  LEN (A$): IF LL = 0 THEN  RETURN
435 A =  PEEK (SL)
440 FOR I = 1 TO LL:A =  PEEK (SL):A =  ASC (MID$(A$,I,1)):
    POKE 229,A: CALL 859: NEXT :A$ =  CHR$ (13): GOSUB 410:
    RETURN
450 REM  SEND TEXTO
455 ONERR  GOTO 520
460 PRINT  CHR$ (4)"OPEN ";N$
470 PRINT  CHR$ (4)"READ ";N$
480 A$ = "0": GOSUB 420
490 A$ = "G": GOSUB 400
500 GET A$: GOSUB 410: GOTO 500
520 POKE 216,0
530 A$ =  CHR$ (26): GOSUB 410
540 PRINT : PRINT  CHR$ (4)"CLOSE"
550 GOTO 380
```

Listagem 2 - Programa AP-PC.PAS para o IBM-PC

```
uses crt;
const
     PORTIN =$379;
     PORTOUT=$378;
     BUFSIZE=32000;
type
    buffer = array [1..BUFSIZE] of byte;
    stnome = string[12];
    starray = string[255];
{-------------------------------------------------------------}
function read_ch(echo:boolean;mask:boolean):byte;
{
      Le 1 byte.
}
var
   i,j:integer;
   a,b:byte;
begin
     port[PORTOUT]:=1;
     a:=0;
     for i:=0 to 7 do begin
          while( (port[PORTIN] and $80) = 0) do;
          port[PORTOUT]:=0;
          while( (port[PORTIN] and $80) <> 0) do;
          a:=a shr 1;
          b:=port[PORTIN] and $10;
          if (b<>0) then a:= a + $80;
          port[PORTOUT]:=1;
     end;
     if(mask)then a:=a and $7F;
     if(echo) then write(chr(a));
     read_ch:=a;
end;
{-------------------------------------------------------------}
function read_st(echo:boolean):starray;
{
      Le 1 string .
}
var
   i,j:integer;
   a,b:byte;
   stbuffer:starray;
begin
     stbuffer:=''; j:=0;
     repeat
          inc(j);
          a:=read_ch(echo,true);
          if (a<>13) then stbuffer:=stbuffer+chr(a);
     until (keypressed or (a=13) or (j=255));
     if echo then writeln('');
     read_st:=stbuffer;
end;
{-------------------------------------------------------------}
procedure read_bin(size:integer;nome:stnome;tipo:byte);
{
     Le um buffer
}
var
   i,j,k:integer;
   b:buffer;
   a,bb,c:byte;
   fp:file of byte;
   tipotexto:boolean;
begin
     if(chr(tipo)='T') then tipotexto:=true else
          tipotexto:=false;
     if tipotexto then nome:=nome+'.txt' else
          nome:=nome+'.bin';
     assign(fp,nome);
     rewrite(fp);

     j:=0;k:=0;c:=10;
     repeat
          inc(j); inc(k);
          if tipotexto then b[j]:=read_ch(false,true)
               else b[j]:=read_ch(false,false);
          if((k mod 64)=0) then
```

```
                write('Bytes transferidos   : ',k,^m);
          if((j mod BUFSIZE)=0) then begin
                for i:=1 to j do begin
                      write(fp,b[i]);
                      if((b[i]=13) and tipotexto)
                          then write(fp,c);
                end;
                j:=0;
          end;
      until ((keypressed) or (k=size) );
      writeln('Bytes transferidos   : ',k); writeln;
      if (j<>0) then begin
          for i:=1 to j do begin
              write(fp,b[i]);
              if((b[i]=13) and tipotexto) then write(fp,c);
          end;
      end;
      c:=26; if tipotexto then write(fp,c);
      close(fp);
end;
{-------------------------------------------------------------}
procedure read_str(nome:stnome);
{
     Leitura de arquivos texto (DOS 3.3)
}
var
   k:integer;
   ch,lf:byte;
   fp:file of byte;
begin
     lf:=10;
     nome:=nome+'.txt';
     assign(fp,nome);
     rewrite(fp);
     k:=0;
     repeat
          inc(k);
          ch:=read_ch(false,true);
          write(fp,ch);
          if(ch=13) then write(fp,lf);
          if((k mod 10)=0) then
               write('Bytes transferidos   : ',k,^m);
     until ((keypressed) or (ch=26));
     writeln('Bytes transferidos   : ',k-1); writeln;
     close(fp);
end;
{-------------------------------------------------------------}
procedure main;
var
   st,nome:string[12];
   ch,tipo:byte;
   compr,code,n:integer;
   b:buffer;
begin
```



```
     clrscr;
     writeln('Transferencia de arquivos Apple -> IBM-PC');
     writeln('W.M.Gonçalves & J.C.Sartorelli - IFUSP');
     writeln;
     repeat
          write('Nome do arquivo      : ');
          nome:=read_st(true);
          write('Tipo do arquivo      : ');
          tipo:=read_ch(true,true);
          if (chr(tipo)='T')then
              writeln('- Arquivo texto')
          else if(chr(tipo)='B') then
              writeln('- Arquivo binario');
          write('Comprimento (bytes)  : ');
          st:=read_st(true); val(st,compr,code);
          ch:=read_ch(false,true);
          if compr<>0 then read_bin(compr,nome,tipo)
                  else read_str(nome);
          writeln;write('Outro arquivo (S/N)? ');
          ch:=read_ch(true,true);
          writeln;writeln;
     until ((ch<>ord('S')) and (ch<>ord('s')));
     port[PORTOUT]:=0;
end;
{-------------------------------------------------------------}
BEGIN
     main;
END.
```

Listagem 3 - Programa BIN2TXT.PAS

```
uses crt;
var
   arq1,arq2:file of byte;
   b,lf:byte;
   nome:string[8];
   nome1,nome2:string[12];
   ch:char;
BEGIN
     lf:=10;
     repeat
          writeln;
          write('Nome do arquivo binario: '); readln(nome);
          nome1:=nome+'.bin';
          nome2:=nome+'.txt';
          assign(arq1,nome1);
          assign(arq2,nome2);
          reset(arq1); rewrite(arq2);
          while (not(eof(arq1))) do
          begin
               read(arq1,b);
               b:=b and $7f;
               write(arq2,b);
```

```
            if(b=13) then write(arq2,lf);
        end;
        close(arq1); close(arq2);
        write('Outro (s/n) ? '); ch:=readkey;
    until((ch<>'s') and (ch<>'S'));
END.
```