53 Printer Programming

"Blessed are the pure in heart."

As everyone knows, Printers help us to produce hard copies. The quality of the printer is referred by the term 'resolution'. Dots per inch (dpi) is the unit of resolution.

53.1 Types of Printers

Nowadays we've got Dot Matrix, Inkjet & Laser Printers. Other old printers like Line, Drum etc, are now obsolete.

53.1.1 Dot Matrix Printers

Dot matrix printers use round-headed pins arranged in a rectangular pattern (like matrix). These pins strike against the inked ribbon and form various characters and patterns. The number of pins determines the print quality, which is usually either 9 or 24.

53.1.2 Inkjet Printers

Spraying inks over the paper forms the characters. The ink particles are ionized and the magnetized plates let the ink to form typical pattern on the paper.

53.1.3 Laser Printers

Laser Printers work just like copier machines. That is, they form the electrostatic image of the entire images on a photosensitive drum with the help of laser beam. Then toner (toner is an ultra fine colored powder) is applied to the drum and so it adheres to the sensitized areas corresponding to the character and other patterns. Now the drum spins over the paper, transfers the toner to paper from drum and the paper gets printed.

53.2 Printer Languages

People thought that it is necessary to have a 'language' to control printers.

53.2.1 Page Description Language

Page Description Language (PDL) is used to communicate usually with page printers. Inkjet and Laser printers are referred as *page printers*, because they manipulate the entire page in memory (Dot Matrix printers manipulate character by character). It is the duty of the internal firmware found on the printer to convert PDL codes to specified pattern of dots. We've got two PDLs namely Printer Control Language (PDL) and PostScript.

53.2.1.1 Printer Control Language

Printer Control Language (PCL) is developed by Hewlett Packard in 1984 to be used in HP LaserJet printers as a PDL. PCL uses control codes (like escape codes). The recent version of PCL is 6.

53.2.1.2 PostScript

PostScript was developed by John Warnock of Adobe as PDL to be used in laser printers. PostScript is referred as a standard programming language. It is also referred as object oriented language, because it sends images to the printer as *geometrical objects* rather than *bitmaps*. This technique is also referred as *vector graphic*, instead of *bitmap graphic*. Recent version of PostScript is 3.

53.2.2 Escape Codes

Dot matrix printers mostly use escape codes. Almost all Laser and Inkjet printers support PDLs. But some printers (Dot Matrix) don't support PDL and they use Escape Codes. The printer commands are sent as a combination of Escape Sequences. For example, to set the line spacing to 1/8 inch, the respective command is ESC '0'. Likewise we've got so many commands or *Escape Sequences*. Escape codes are non-standard as each printer vendor use different sets of Escape Codes.

53.3 Printing non-printable characters

In this section, I am going to explain how to print non-printable characters on Epson 9 pin Dot Matrix Printer. This can be achieved by creating PCL or PostScript file. But for that, you have to know the *file format* of them and it is a tedious job. It means that you have to develop your own software that 'creates' PCL or PostScript! The easy way is to use *Escape Codes*.

Note

Since the Escape Codes are mostly 'printer' and 'vendor' specific, the Escape Sequences I have used here will mostly work **only** on Epson 9 pin Dot Matrix Printers.

53.3.1 Epson Extended Character Set

Ordinary Epson character set doesn't have non-printable characters. But Epson Extended character set contains all printable characters and 'few' non-printable characters: single box characters, heart, diamond, club, spade, plus/minus sign, and division sign. But this extended character set uses different values to represent such an extended character.

Character	ASCII value	Epson Extended Character Set Value		Character	ASCII value	Epson Extended Character Set Value
┌	218	135		─	196	133
┐	191	136		│	179	134
└	192	137		♠	6	145
┘	217	138		♥	3	146
├	195	132		♦	4	147
┤	180	131		♣	5	148
┬	194	130		÷	246	158
┴	193	129		±	241	159
┼	197	128				

To set the printer to Epson Extended Character Set, we have to send `ESC 'm' 4`. For that we can use the `biosprint( )` function. As this mode uses 'character set', it will be faster than graphics mode.

53.3.2 Graphics Mode

Graphics mode is the slowest one. To set the printer to graphics mode, we have to send: `ESC '*' n1 n2 n3`.

where n1 is the resolution (n1 = 4 means 80 dpi),
n2 = number of bytes to print % 256,
n3 = number of bytes to print / 256.

Pin	Pin No.	Command to be sent
●	7	128 (2^7)
●	6	64 (2^6)
●	5	32 (2^5)
●	4	16 (2^4)
●	3	8 (2^3)
●	2	4 (2^2)
●	1	2 (2^1)
●	0	1 (2^0)

Let's see how to program the pins of printer head. To activate the bottommost pin 0 we have to send 1 as a command, to activate pin 1 we have to send 2 as a command…

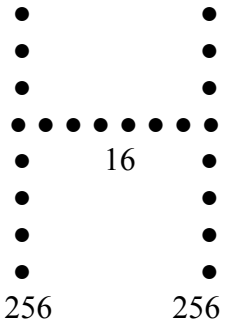
So to activate pins 0, 1 and 7 at a given time, we have to send 1+2+128 = 131 to the printer. Before that, it is necessary to set the printer to graphics mode with the command:
`ESC '*' 4 8%256 8/256` (or `ESC '*' 4 8 0`).

Note

At a given time you can program up to 8 pins only. So if you sent 256, all pins will be activated. You cannot program the 9th pin (i.e., pin 8).

For example, to print the character 'H' in graphics mode, the command to be sent to the printer will be:

`ESC '*' 4 8 0` (then pin values) `256 16 16 16 16 16 16 256`



53.3.3 Font Map

```
10000001
10000001
10000001
11111111
10000001
10000001
10000001
10000001
```

Font map of 'H' that will be in ROM

Now we have learned about graphics mode. In the previous section, we have manually found out the 'pin values'. Manually finding out pin values is a tedious job and it is tough too. Fortunately in ROM, we've got 'font map' for each characters. So it is wise to use font map, which is already available in ROM to generate *pin values*.

Interrupt 10h, AX = 1130h, BX = 0300h returns pointer to 8x8 font.
Interrupt 10h, AX = 1130h, BX = 0200h returns pointer to 8x14 font.

I prefer 8x8 font, because it reduces the programming effort and speeds up printing.

Note

If you prefer 8x14 font, you have to print the part of the font (with height 8) in one line and then you have to print the remaining part of the font (with height 6) in another line.

The returned pointer by int 10h will point to the font map of first character of the ASCII set (i.e., NULL or ASCII-0). The font map of the letter 'H' will be at the offset 'H' (ASCII-72). Similarly font map of every letter of the ASCII set (including non-printable characters) will be at the offset of its ASCII value. So with the help of the pointer and a simple program, we can find out the *pin values* easily.

53.3.4 Optimization Tip

We must understand that graphics mode is the slowest one. Printing with *Epson Extended Character Set* is faster than graphics mode. So it is wiser to use Epson extended character set's all available characters. For all other non-printable characters use *graphics mode*.

53.3.5 Program

The following is the code to print non-printable characters on Epson 9 pin Dot Matrix Printers.

```
/*-----------------------------------------
      PR - To print non-printable characters
   File name: Pr.c
 *-----
 */

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

#define PRINTER_WRITE   ( 0 )
#define PRINTER_STATUS  ( 2 )
#define ESC             ( 27 )
#define LPT1            ( 0 )

#define FNTHEIGHT ( 8 )

void Send2LPT1( int num, ... );
void SetLineSpacingTo1by8( void );
void SetPrinter2GraphicsMode( void );
void PrintWithEpsonCharSet( unsigned char ch );

/*-------------------------------------------------------
      Send2LPT1 - Send 'num' characters to LPT1       */

void Send2LPT1( int num, ... )
{
   va_list argptr;
   int i;
   va_start( argptr, num );
   for ( i=0 ; i<num ; ++i )
   biosprint( PRINTER_WRITE, va_arg( argptr, int ), LPT1 );
   va_end( argptr );
} /*--Send2LPT1( )-------------*/

/*-------------------------------------------------------------------
      SetLineSpacingTo1by8 - Sets line spacing to 1/8 inch        */
```

```
void SetLineSpacingTo1by8( void )
{
   Send2LPT1( 2, ESC, '0' );
} /*--SetLineSpacingTo1by8( )------*/
/*-------------------------------------------------------------
      SetPrinter2GraphicsMode - Initializes printer to graphics mode */

void SetPrinter2GraphicsMode( void )
{
   Send2LPT1( 5, ESC, '*', 4, 8%256, 8/256 );   /* 80 dpi quality */
} /*--SetPrinter2GraphicsMode( )-------*/

/*-------------------------------------------------------------
      PrintWithEpsonCharSet - Initializes printer to Epson Extended
        Printer Character Set and print a single character 'ch'.
        Epson Character Set contains all printable characters, single
        line box characters and few other ASCII characters.
        (It is faster than Graphics mode.)                  */

void PrintWithEpsonCharSet( unsigned char ch )
{
   Send2LPT1( 4, ESC, 'm', 4, ch );
} /*--PrintWithEpsonCharSet( )-------*/

int main( int argc, char *argv[] )
{
   FILE *fp;
   struct REGPACK regs;
   unsigned char ch;
   char far *font8x8, far *ptr;
   unsigned int segment, offset;
   int fntval, mask, powof2, status;
   register int i, j;

   /* call the bios interrupt to get the address of the desired font */
   regs.r_ax = 0x1130;
   regs.r_bx = 0x0300;
   intr( 0x10, &regs );
 /*make a far pointer font8x8 point to info returned by the bios call*/
   offset = regs.r_bp;
   segment = regs.r_es;
   font8x8 = (char far*) MK_FP( segment, offset );
    /*---Check For any Errors-----> */
   if ( argc < 2 )
    {
      cprintf(
    " Syntax: PR filename [ -bb | -nbb ]                          \a\r\n"
```

```
 " -bb     Box Better. Box characters will appear better. But \r\n"
 "         characters of adjacent lines may touch each other. \r\n"
 "         (default)                                          \r\n"

 " -nbb    No Box Better. Characters of adjacent lines won't  \r\n"
 "         touch each other.                                  "
  );
   exit( 1 );
 }
status = biosprint( PRINTER_STATUS, 0, LPT1 );
if ( status & 0x01 )
   {
    cprintf( " Fatal Error: Printer time out \a\r\n" );
    exit( 1 );
   }
if ( status & 0x08 )
  {
   cprintf( " Fatal Error: I/O error \a\r\n" );
   exit( 1 );
  }
if ( (fp = fopen( argv[1], "rb"))==NULL )
  {
   perror( " Fatal Error\a" );
   exit( 1 );
  }
/*-- <---Error Checked...OK!  --*/

/*  if switch is not equal to "-nbb", then do default ie, "-bb"  */
if ( strcmpi( argv[2], "-nbb" )!=0 )
         SetLineSpacingTo1by8( );

while ( !feof( fp ) )
 {
   fread( &ch, 1, 1, fp );
   if ( ch=='\r' || ch=='\n' || ch=='\a'|| ch=='\t'|| ch=='\v'
                       || ch=='\f'|| ch=='\b'|| ch==0
                       || ch==255 || (ch>=' '&&ch<='~') )
         PrintWithEpsonCharSet( ch );
     else
       {
          switch( ch )
           {
               /*  Box Characters adjust  */
               /*  upper left corner  */
               case  218:  /* '┌' */
                         PrintWithEpsonCharSet( 135 );
                         break;
```

```
/*   Upper right corner  */
case  191:  /* '┐' */
          PrintWithEpsonCharSet( 136 );
          break;
/*   Lower left corner  */
case  192:  /* '└' */
          PrintWithEpsonCharSet( 137 );
          break;
/*   Lower right corner  */
case  217:  /* '┘' */
          PrintWithEpsonCharSet( 138 );
          break;
/*   Middle left corner  */
case  195:  /* '├' */
          PrintWithEpsonCharSet( 132 );
          break;
/*   Middle right corner  */
case  180:  /* '┤' */
          PrintWithEpsonCharSet( 131 );
          break;
/*   Middle top corner  */
case  194:  /* '┬' */
          PrintWithEpsonCharSet( 130 );
          break;
/*   Middle bottom corner  */
case  193:  /* '┴' */
          PrintWithEpsonCharSet( 129 );
          break;
/*   Center cross  */
case  197:  /* '┼' */
          PrintWithEpsonCharSet( 128 );
          break;
/*   Horizontal  */
case  196:  /* '─' */
          PrintWithEpsonCharSet( 133 );
          break;
/*   Vertical  */
case  179:  /* '│' */
          PrintWithEpsonCharSet( 134 );
          break;
/*  Other ASCII Characters adjust  */
case  6:          /* spade */
          PrintWithEpsonCharSet( 145 );
          break;
case  3:    /* heart */
          PrintWithEpsonCharSet( 146 );
          break;
```

```
                    case  4:    /* diamond */
                                PrintWithEpsonCharSet( 147 );
                                break;

                    case  5:    /* club */
                                PrintWithEpsonCharSet( 148 );
                                break;
                    case  246:  /* 'ў' */
                                PrintWithEpsonCharSet( 158 );
                                break;
                    case  241:  /* 'ë' */
                                PrintWithEpsonCharSet( 159 );
                                break;
                    default:
                          mask = 128;
                          SetPrinter2GraphicsMode( );
                          for ( i=0; i<8; ++i )
                           {
                              /* make ptr point the start of the letter in
                                 the rom font each character is FNTHEIGHT
                                 bytes with each bit in a byte being a
                                 pixel on/off for that scan line of the
                                 charater
                              */
                              ptr = font8x8 + ( ch * FNTHEIGHT );
                              fntval = 0;
                              powof2 = 128;
                              for ( j=0 ; j<8 ; ++j )
                                {
                                fntval += (*ptr&mask) ? powof2 : 0;
                                ++ptr;  /* ptr points to the next scanline
                                                of the current character */
                                powof2 >>= 1;
                                }
                              biosprint( PRINTER_WRITE, fntval, LPT1 );
                              mask >>= 1;  /*  or dividing by 2  */
                           }
               }
          }
     }
   fclose( fp );
   return(0);
} /*--main( )------------*/
```

Suggested Projects

1. As far as I know, there is no function library for printing purposes. So develop your own PRINTER.LIB. The library should contain similar functions like `Set2GraphicsMode()`, `SetLineSpacingTo1by8()`, etc. It is very easy to do so! No programming skill is necessary! The only thing you need is Escape Codes.
2. Write a program that prints the given text in Braille characters. (Hint: You may need to alter your dot-matrix printer. That is, you have to remove the ribbons, replace the existing soft pins with hard pins. For programming, use graphics mode)