

SIGNS.DOC
Version 5.0

by Robert W. Bloom
26 Sep 89

Release Notice:

Signs and all associated files (Signs, MkFntNdx, font files, etc.) are freely released to the public for non-profit use. If anybody's going to profit, it should be me! Anywhat, this is my contribution for all the great programs I've received from the pipeline. Thanks guys!

Revision History:

Signs:

- 1.x - March 86, first versions, menu driven. GSigns created.
- 2.x - May 86, all screen oriented.
- 3.x - Jun 86, bit mapped font files, released w/MakeFont Version 2.0. New features: inverse video, user changeable single block character, inter-character spacing, block_char used for fill in inverse video.
- 4.x - Oct 86, added bit-mapped output, Library released with 3 font files (font1, font2, gothfont), MS-DOS version released and patched so that 'Gothfont' fit.
- 4.2 - Aug 89, patched to compile with TurboPascal v5.0.
- 5.0 - Sep 89, Redesign of font file interface to use HP LaserJet font files, elimination of MakeFont and Generic versions for CP/M, addition of a font index file generator, expanded Epson printer support. Converted old font files into HP LaserJet-compatible font files. TurboPascal v5.0 specific.

Older versions are still workable, but why bother. Though I still have my CP/M machine, it doesn't get turned on much.

Background:

I was never happy with the sign- and banner-making programs available for CP/M. Sure there were a number of good ones, specially 'GOTHIC,' but none have enough variability to suit me. By variability, I mean the capability to change sizes, fonts,

and paper positioning to create whatever I wanted. What I really wanted was a 'Print Shop' for CP/M-80. Alas, it's not available and it looks as if no one is developing any new stuff for my (old) favorite OS.

I thought the problem was solved when I ran across a set of two programs by Ken Crook in Pascal. I was just learning Pascal, so I thought this would be a good way to get some fun education. Unfortunately, the program was not written very well - although some good options were present, I couldn't even compile the programs (called 'sign' and 'banner') as they wouldn't fit in memory. (CP/M-80, TurboPascal v3.01, 63k, error messages off.) The main problem was that all of the character fonts were held in memory at one time. Even eliminating all the symbols and keeping just numbers and upper case letters yielded a com file of 36k. (Besides, input was taken from a file not the console, which was a pain.)

Features:

So I decided to start over and write my own. Major features of my program are:

- Fonts are held in an external HP LaserJet-compatible 'soft' font file which is read by random access methods. It reads only the required characters into memory when they are needed. An index to the font file is required (contains character descriptor data and pointers into the HP file) which is created by the separate program MkFntNdx.

- Signs and Banners are made by changing an option. They may be output to the console, a file, or lst: device (printer). Lots of other options are available. (see below)

- Any HP LaserJet-compatible 'soft' portrait font may be used. Because of the very-high resolution 300dpi of the LaserJet, many soft fonts will be 'too big' for screen output. I.e., each dot is turned into a character position limiting screen output to 80 'dots' wide. To remedy this, two font sets are supplied: one with characters that are up to 12 units high counting a two block descender and up to 10 blocks wide; and one which is four times wider and twice as high. Both are proportionally spaced. The smaller of the two is readable only

with a magnifying glass if actually downloaded to LaserJet - but it works fine with Signs.

- Font editors are *not* supplied, editing an HP font file is a very tricky business. 'AlterFnt' and 'QFont' are two shareware-available editors available. Many soft fonts are also available. I designed one of supplied fonts, a friend (RK Sparks) did the other.

- Four printers are supported: IDS, Epson, HP and 'dumb'. Dumb printers can't output in graphics dot-mode. There are four graphic output densities for the HP and Epson.

- If printing, one can change the cpi and lpi from the menu which automatically adjusts the printer. Input lines are checked to see if they will fit. Normal (80) and wide (132) terminals, as well as small (8") and large (14") printers are correctly handled. The user can even request output in "reverse video" (White letters, black background instead of black letters on a white background.)

- The letters of the output sign/banner can be made of the letter itself, any single character, a 'overstrike' block to make a nice black solid block or individual dots using the graphics mode of the printer. (Naturally the graphics dot-mode will create much smaller letters and is strongly printer dependent.

I think you'll find that the program is self-explanatory. Once selected, a option does not change until manually changed. This allows changing just one option quickly. Some options eliminate others.

Files:

READ.ME - introduction to signs.

RWBFONT.FNT, RKSFONT.FNT - HP LaserJet compatible soft font files

RWBFONT.FNX, RKSFONT.FNX - index to above created by MkFntNdx and used by Signs.

MkFntNdx.PAS - (Make Font Index) source code for font index generator - asks for input filenames.

SIGNS.PAS - source code for sign generator. Also

calls the include files: CONST.PAS, ASK.PAS,
DISP.PAS, PRT.PAS, UTIL.PAS
SIGNS.IN - sample file input, contains all printable
characters.
SIGNS.OUT - sample output file from signs.in
SIGNS.DOC - this file.
SIGNS.EXE, MKFNTNDX.EXE - executables
PRINTER2.PAS, PRINTER2.TPU - TurboPascal v5 fix for
LST output

Compiling:

TurboPascal v 5.0 was used to compile the program,
using the CRT and PRINTER2 'use' units. A
installed Ansi driver is required for the
low/high video calls.

I found a bug in TurboPascal v5.0: the default
'printer' unit in TURBO.TPL will not output a
CHR(26) which is needed when one outputs
graphics. Borland provided the 'printer2' public
domain unit which fixes the problem. This
problem was not in TurboPascal v3.

Running:

As a minimum, one needs to have the following files in
the default directory: Signs.exe, one HP
LaserJet-compatible soft font file, and a index
to the font file (created by mkfntndx.exe.)

Notes:

- some options are disabled with other options. For
instance, if one selects console output, you
don't see any printer options. Or if you enter a
given left margin to use, the output centering
option is disabled. (To center the output after
zeroing a previously-given left margin, one has
to manually enter 'y' to centering as a non-zero
left margin sets centering to 'no'.) Non-
applicable menu items are not displayed.
- beware of hardware differences in printers. If in

doubt, make the printer type 'dumb' and avoid bit-mapped graphics output.

- Outputting to a file automatically sets the line width to the maximum. You probably will want to reduce it to something reasonable - especially if you ask for auto-centering.

Options:

Upon entry to the program one is placed directly in "change parameters" mode. A options menu is shown: (The actual menu will have inappropriate entries missing, they will appear as they become pertinent.)

Signs Version: v5.0, 10 Sep 89
Mode: **Change Parm**s

```

----- Options and I/O Parameters
-----
T-      Sign type -> Sign          I-      Input
      Device -> File
B- Block/Letter type -> Letters    R-      FileName to
      Read -> Signs.in
F-      Font File -> Font.usp      N-      Number of
      Copies -> 1
W- Width Multiplier -> 1           O-      Output
      device -> Printer
H- Height Multiplier -> 1          S-      Device
      Size -> Normal
V- Inverse Video -> Off, Normal    Y-      Printer
      Type -> Epson
A- Auto-Centering -> Yes           P-      Pitch
      chars/inch -> Squeezed [17]
M- Given left margin -> none defined L-
      Lines/Inch -> Twelve
G- Given Width -> not given        C-      Color of
      Print -> Black
Q- Quit and Return back to OS      D-      Graphics
      Density -> Single
X,<cr>- eXit Change Parm           E-      rEcord Output
      In -> Signs.out
-----

```

```

-----
Width available -> 136           Font width: 10
Height: 12
                                Enter option letter
-----
-----

```

On the options menu one can change:

- Sign type: Signs are horizontally printed across page, banners are vertically printed down page
- Block type: The character that makes up the signs can either be the letter printed, solid block, or a single 'dot' (bits). Blocks may be made two ways, by over striking several characters or by sending a printer a single user-defined character. (The default single block character is 177 which is defined as a block in the IBM-PC extended character set. Your printer will probably print something else. Mileage will vary.) If one selects single block, it will ask what character to send. Enter the decimal number of the character. For bit-mapping, each 'block' becomes a single dot in the graphics mode on a printer. When entering bit mode, the width of the output line is automatically set to whatever is appropriate for the printer defined.
- Font file: One can change the name of the font file to be used at any time. If the filename entered it not found, an error message and '????' will be shown for the filename. If found, the size of the font will be shown. An filename extension of .FNT is assumed if not entered. Program will complain if the file is not found or a associated index with a .FNT extension.
- Mult w,Mult h: The output may be doubled, tripled, etc. in both width and height. Enter your choice, the program will simply not print any characters that overflow the output line.
- Inverse video: Does what you think: spaces are printed as the block character, and the character

is printed as a space. Not available with 'bit-blocks.' The single-strike block character (default 177) will be printed between characters.

- Centering: If no given offset, answer if output be centered on page or flush to left margin.
- Given Margin: Enter how many characters should be sent before printing signs or banner. This has the effect of a left margin for signs or moving banner 'left' or 'right' on the output device. (By playing with the given offset and reverse formfeed, one can get multiple lines in Banner format.)
- Given width: If not zero, the program will use this figure for the maximum number of characters per output line. Also centering, if on, will be based on this width if not zero. Automatically set to Max_Length upon entering bit-mapped mode (block_type = bit).
- Quit: one can quit from the options menu with a 'Q'. One might need this if you forget the name of the font file to use and signs won't let you exit until you give it a good name.
- eXit: Upon entry of a 'X' or <cr>, one is placed in "input text" mode.
- Input device: One can enter the output lines one at a time from the keyboard or take them in a bunch from a designated text file. The default filename is SIGNS.IN. If input from a file, one can specify how many copies you want. A formfeed is sent between copies if output to a printer.
- File to Read: Name of the file to take input from, multiple lines are output separately.
- Number: If reading from a file, how many copies to output.
- Output device: Specify output should be sent to the

console screen, a text file, or the lst: device. If recording output in a file, the default name is SIGNS.OUT. If printing, one can: enter what pitch should be used (10 pica, 12 elite, 17 squeezed or 20 tiny [squeezed gives 132 characters per 8" line]); how many lines per inch should be used (6, 8, 10, or 12); and color (black, blue, green or red).

- Device size: If printing, indicate whether printer is 8" wide or 14" wide. If displaying on screen, indicate whether terminal is 80 characters wide or 132. Output width for file output is whatever was previously set with a maximum of Max_Length. Specify a given width if you want centering to a different width.
- Printer: Either Epson, IDS, HP or 'dumb'. Dumb printer are not sent any control codes and are not capable of dot-graphics.
- Pitch, Lines: Set to whatever you want for best appearance. Not available for dumb printers. Dot-graphics reset this parameter to whatever is appropriate for graphics.
- Color: The IDS printers support color.
- Density: Epson and HP printers have different graphic output densities. Characters will change size and shape as this is varied.
- Output file: Name of file to store output in.

Note that the size of the font in use is displayed after the font file has been opened, as well as the calculated available output space.

The input text mode has a menu of control characters:

```
-----
-----
^P: Change parameters      ^F, ^L: Send Formfeed to LST
  ^X: Restart Input
^C, ^D: Done, quit to OS   ^R, ^T: Send Reverse Formfeed
```


`^H`: Backspace
`<ret>`: Process Input `^A`: Alternate Input

- send a formfeed to the printer ('^F' or '^L'),
- send a reverse formfeed to printer to return to TOF ('^T'),
- goto the options menu and change the program parameters ('^P'),
- exit program ('^C' or '^D'),
- backspace ('^H' or),
- ask to redisplay the menu again ('^R'),
- delete entered line and start over ('^X'), or
- enter alternate input routine to enter non-ASCII characters ('^A')
- anything else is entered into the line to be printed.

(You CAN mix input and control characters, i.e. enter the input line, strike a ^P to change the parameters, and return to input line without losing anything.)

The alternate input routine (^A) prompts you to enter a DECIMAL code of a character. These can be put to good use if the font file has defined more than the standard 96 ASCII characters. Entries greater than 255 are rejected, others are appended to the input string. The decimal value is echoed as the character is appended. Enter a <cr> to return to normal input.

Hardware:

There is hardware-specific code in the source in several procedures: (I've tried to but them all in the prt.pas include file.)

Procedure `set_up_prt` contains all the character sequences necessary to set and reset the printers in the various modes (10, 12, 17, 20 pitch.) The code included in the distribution copy is for the

IDS MicroPrism (or Prism), Epson, and HP LaserJet. You will have to change it for any other printer model. Note: the IDS printers can't do 20 pitch.

Procedure `calc_width` contains the actual pitches for the various conditions. Squeezed print on the IDS is 16.5 pitch, Squeezed print on the Epson is 17 pitch. Tiny (20 pitch) is available on the Epson, not on the IDS.

Procedure `out_char` outputs 'Block_Char', (defined as #127 in the CONST block as the default.) This printers a single pass block for the IDS printer. Apparently the standard mode on the Epson printer doesn't have a one-pass block character (only in 'IBM-mode'.) I advise not to use block mode with the Epson, use overstrike mode to print blocks or define the single pass 'block' character as a '*' or 'X'.

Procedure `main` outputs strings to the printer for formfeed (^L - that's pretty standard) and reverse formfeed (<esc>H0\$<esc>G0\$ - that's not standard).

Procedure `out` outputs one graphics line. The graphics line is varying lengths, depending on the mode and is contained in the char array `gout_1` and `gout_2`. Triple and Quad density for the Epson interleaves the tow graphics lines.

To change number or characters for creating the overstrike block, adjust the `os_char` string to the appropriate characters. The length of `os_char` determines the number of strikes (currently the maximum length is 5).

Notes on Programming Techniques:

I have a tendency to write my Pascal procedures and functions too long according to classical thought. (So much for classical thought.)

However, the long procedures all contain just one idea (or menu) and couldn't be easily and clearly broken. Main, out_sign, out_banner, and check_sign are also too long according by my Pascal Perfesser. (What does he know, anyhow? - how to spell professor for one!) Anywhat, I'll somebody else figure a good way to break it up.

I've even used a couple of GOTO's (heresy!) where it was easier to use one than to figure out how to loop it. There's too many loops as it is. Purists will complain that there are too many global variables and I should have passed the record variable by pointer. Having learned by style from FORTRAN, I'm a long ways from being a purist.

Speaking of style, my style of programming is different from most, I do a lot of dBase II programming so have picked up its style and translated it to Pascal. In particular, BEGIN/END pairs are **not** necessarily matched up in indenting, but flow control statements more than two statements apart **are** and a terminator is used. If there's a natural 'END' there, fine; otherwise a {end} is used. Try it, I like it.

One can change my defaults by changing the initialized 'constants' at the top of the file (actually in the include file const.pas). Warning: if you make input from or output to a file the default, you will somehow have to open the file in procedure main - I opened the files when the user exits from ask_parameters.

Most procedures are in 'natural', i.e. FORTRAN, order. (Another throwback!) Main comes first, followed by major subroutines, closed out with utilities. This has the effect of causing all procedures and functions to be FORWARD declared. (Or almost all - main isn't as it's first anyway.) The real program at the end simply announces itself, calls main, and announces termination. Pascal can be readable! The disp_? and ask_? procedures are

not in natural order and are not forward declared. They are randomly accessed anyhow.

Another warning: the program may hang if the printer is the defined output and is off-line when you exit. This happens because a reset string is sent to it even if no output is done. (assuming the dumb printer switch is off.) This may be a concern if the printer is the default output and one exits without any input.

A carriage return to the entry prompt without any input characters will ask to exit. Input a space<cr> if you want a blank line to be output.

While outputting to a file, the file will remain open until program is exited or the output direction is changed. This way, multiple output lines may be stored in one file. Remember that when outputting to a file that the width is set whatever the previous maximum was. The program can handle up to 3000-some character wide output. Auto-centering may add more or not enough spaces than you want, so set a smaller/larger width with the given-width setting.

Bugs:

I did not figure out how to handle negative left-offsets in the font files when outputting banners. A negative left-offset means that the cursor should move LEFT that number of dots - for a banner that means 'taking back' lines that have already been output.

The algorithm that calculates the actual number of character across needed for a sign is not correct for all cases - negative left-offsets throw it off. The effect is when outputting in inverse video, the border lines are not always the right length.

Have fun. If anyone adds new and exciting options, I would be interested in seeing it. I 'think' all the bugs are out, but one never knows ...

Robert W. Bloom 46 Broadleaf Drive
Newark, DE 19702