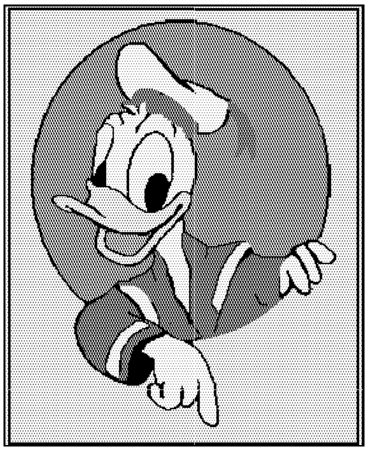
The BM2FONT Manual
(English Version)

Heinrich-Heine-Universität
Düsseldorf
Universitätsrechenzentrum
Friedhelm Sowa

Version 2.0 — January 25, 1993



Integration
of
Graphics
and
Pictures
into TEX–Documents

Contents

List of Figures

1 Introduction

The integration of graphics and halftone pictures into TEX– and LATEX– documents can be done in many different ways. The main disadvantage of most of the methods is that the format of the graphics or picture will be in a device dependent format. This often requires a different format of TEX commands for each printer and specialized driver programs.

It seems to be an anachronism that after more than ten years of TEX's release there is not a common means of inclusion of graphic elements that will work with all printers.

The need for the solution of this problem is increased by the profiliteration of user friendly graphics programs that are available on personal computers and workstations. Also, the improving quality, availability, and decreasing price of optical systems like scanners, cameras, and video scanners have enabled convenient and economical digital sources of photographs and works of art.

This document is intended to introduce the TEX user to a system that should solve this problem, even for the casual user. A logical procedure which solves this problem of integration of graphics into a TEX document is outlined by the following steps:

1. Obtain the desired picture or graphics in hard copy form.
2. Translate that picture into the TEX language so that it will be understandable for all driver programs. Since TEX's basic actions are to place a character (or rule) at a given point, this means translate the picture into a series of characters that TEX can properly place on the page.
3. Issue the commands that place these characters in appropriate positions relative to the others.
4. Use TEX to typeset the document.
5. Use a driver program to convert TEX's output into a hard copy.

The second step in the list is the one which has been missing in a portable and reliable form. Again, since TEX's primary means of putting ink on paper is through the use of fonts, this step is accomplished by converting the graphic into a series of characters in a "special" font that is meaningful for this graphic only.

The tool or program which does this is called BM2FONT. This is a program which is available for IBM PCs and compatibles at this writing. It is intended to be publicly distributed as is the rest of the TEX system. Although it initially runs only on this class of computers, its output has no restrictions on use **except** that each run creates a font for printers of one resolution and one marking technology at that resolution. This philosophy is obtained as the output files of the program are to be understood on any computer, running under any operating system with its 'local' implementation of the TEX system.

We will discuss in some detail that the original sources of these graphics can come from computer generated pictures or pictures entered into a computer by means of a scanner or other form of electronic camera.

Caveat: The font that the picture has been created at works only for that resolution of printer. There is another element that affects the quality of the output picture. Some printers form the printed page by marking the **black** pixels and others form the printed page by erasing the **white** space where pixels are not black. Different fonts should be done for each type printer. The majority of printers that are available are of the *write black* type.

The picture of Donald Duck on the title page was obtained from a public archive of bitmaps. The process of including it in the document was to execute the conversion utility like:

```
bm2font donalt.gif
```

which created several files: `donalt.tex donalta.pk donalta.tfm`

(Notice that the original version of this manual was in German. If it had been in English, the first two files mentioned would have been `donald.gif` and `donald.tex`). The `bm2font` utility has converted a graphics file into two files that describe a font and a TEX source file that uses the font described in the other two files. The user's TEX source must now include two fragements of TEX code which implement the use of the font/graphics. These fragments are shown in the following two lines:

```
\input donalt
```

```
\setbox\donalt
```

The last line is used just like a character or any other box in TEX. We will give more details later.

2 Sources of pictures

The contents of a graphics file can consist of either vector or raster information. Vector information will eventually undergo a metamorphosis in the printing cycle and be changed to raster information since our output devices print rasters that are always 'ink' or 'no ink.' These two basic states of binary systems are common throughout the world of electronic data processing. If the bit is "on", it has ink and if it is "off" it has value "0". On the paper either black or white areas will appear. There is an obvious extension for the handling of colors in that it is layers of these pixels of each color that will be "on" or "off."

The idea of converting vector graphics to raster graphics is not new. It is what the 'printing engine' in todays printers do as their primary function. The basic printers actually copy characters (which are usually bitmaps) to the bitmap for the whole page. The more advanced printers convert vector commands into bitmaps. Most printers also have commands that allow the "filling" of closed areas with ink. Thus, some printers can create characters by drawing the outline with vector commands and "filling." This does not lead to high quality fonts at the usual 300 dot/inch resolution. The use of such filling techniques is seldom helpful in pictures. These have rows of rasters that frequently have little in common with the adjacent rows of pixels. The contribution here is that the conversion happens at a stage that makes the graphics into a font which can be used by TEX in a manner that is reasonably device independent. This enables the basic printer to also do this graphics.

There are many commercial and public domain programs that can be used to create graphics on personal computers and workstations. There are also several *de facto* standards for storing bitmapped graphics. Most of the programs that allow the user to create graphics support more than one of these output formats for the storage of graphics.

We decided that the support of bitmapped graphics should be supported because it is the 'lowest common denominator' of all representations of graphics. The machine readable graphics can be created by programs like MacPaint or by scanning photographs, sketches, and drawings with some kind of a camera device. Vector graphics can also be converted in the same manner or more directly by the use of the vector information and output device resolution.

The different bitmap formats that are supported by BM2FONT will be described. Some details as to their common origins, uses, and limitations will be given. The suffix of the input file **is important** and is used to specify the format of the input bitmap file. If that format requires a header, then "proofing" of the format will be done by scanning the header. Deviations are reported by the BM2FONT. Conversion of the graphics continues, if possible. This causes unexpected results in some cases. The reasons are:

- These standards are often poorly documented.
- These standards are often incomplete and subject to significant variations in their implementations.
- Programs that create such files are like all other programs in the world and are therefore subject to bugs.

We have used the word "standards" rather loosely here. These are certainly not standards (in most or all cases) similar to those promulgated by ANSI, DIN, and/or ISO. These are standards only in that there is some documentation available, more than one vendor's product can output files meeting the standard to some extent, and that it is apparently 'supported' by more than one vendor.

BM2FONT support is based upon the best information available. Errors or warnings about inconsistencies in the input file can appear like: "keep your fingers crossed." It is possible that you may still get a good document, but don't count on it.

The different types of bitmaps that BM2FONT currently handles are:

`PCX`, `GIF`, `BMP`, `IFF` or `LBM`, `TIFF`, `IMG`, `CUT`, and `BitMaps`, a home grown basic version.

2.1 PCX files

ZSOFT developed the PCX–format for dumps of video screens. It now supports up to 256 colors in a picture and is quite efficient in storing a screen bugger. Only 16 colors were supported through versions 3.0. It uses a runlength encoding scheme to achieve the economy of disk storage. This format is also used by some programs which "capture" screens and make files for hardcopies of the screen information.

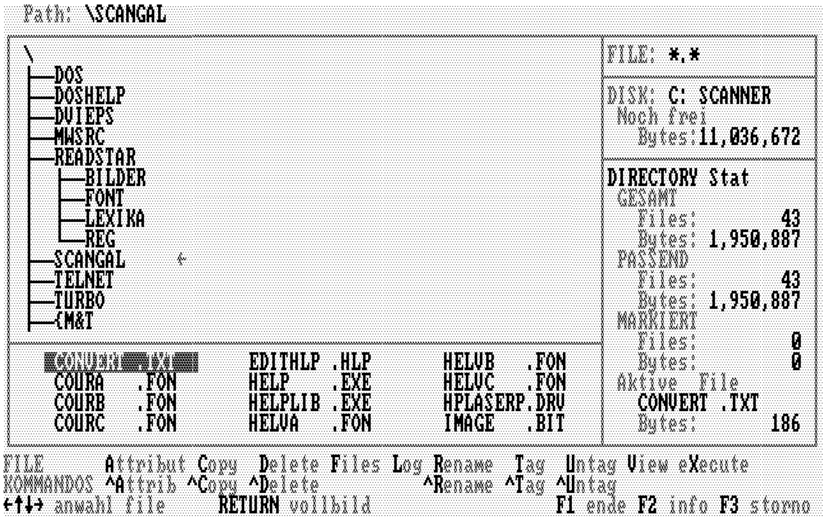


Figure 1: Example of a PCX file containing 16 grey shades, printed with 15 grey shades

Paintbrush, PC Paintbrush+, Deluxe Paint II or Publishers Paintbrush support the PCX format as well as some scanner systems.

2.2 GIF files

CompuServe introduced the Graphics Interchange Format (GIF) in 1987. It is a device independent and mainly used for exchange of graphic files within network systems. The LZW algorithm is used to obtain efficient storage.

BM2FONT does not support all commands that can appear in GIF file. For example, BM2FONT will extract only the first picture from a GIF file that may contain mutliple pictures. This decision is based upon the fact that each picture should be converted to fonts individually. Further, the tools should be kept as simple as possible. Overlays and other special effects should be done by specific software. Animation is not appropriate for paper documents. Taking only the first picture ignores the problem but keeps this tool simple.

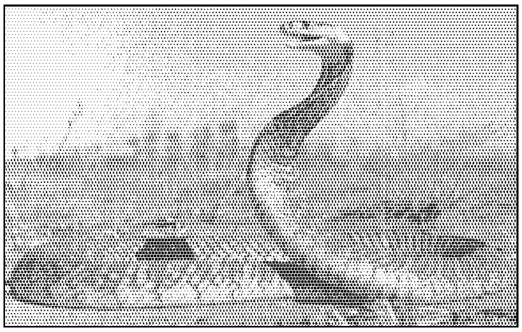


Figure 2: Example of a GIF file with 256 colors, printed with 16 grey shades and separation of raster points

The documentation of the GIF standard has a discussion on the aspect ratio of pixels in the file. This information is not given in most of the cases. As a matter of fact, none of the GIF files that we have seen have such information imbedded. A picture that was created on an EGA video screen would look distorted on paper if this is not included. Thus, BM2FONT looks for the typical horizontal and vertical resolution of an EGA card and stretches the picture to a suitable height. This "default stretching" can disturb the intention of the picture, particularly when printing art graphics. The user can override this by furnishing an explicit aspect ratio.

2.3 BMP files

Windows 3.0 includes tools for creating graphics in the device-independent BitMaP (BMP) format. This format supports uncompressed bitmaps and run length encoding (RLE) compression for samples of 4 and 8 bits. These represent 16 or 256 color maps, respectively.

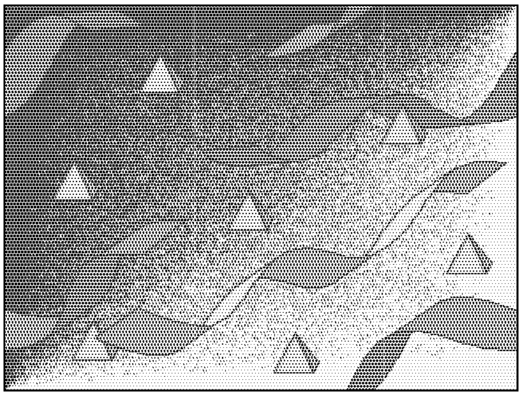


Figure 3: Example of a BMP file with 16 colors, printed with 16 grey shades and separation of raster points

BM2FONT does not support the RLE compression of BMP graphics in the current version. At this time we have not had an opportunity to test decompression on pictures with RLE. Current versions of graphics programs that we have access to using Windows 3.0 apparently do not support RLE. As files including this feature become available, this feature should be easily added.

2.4 IFF or LBM files

The Interchange File Format standard was defined by the Electronic Arts corporation. The format is mainly used on Amiga computers. It has been introduced to the PC world by the graphics software DeLuxe Paint and DeLuxe Paint II.

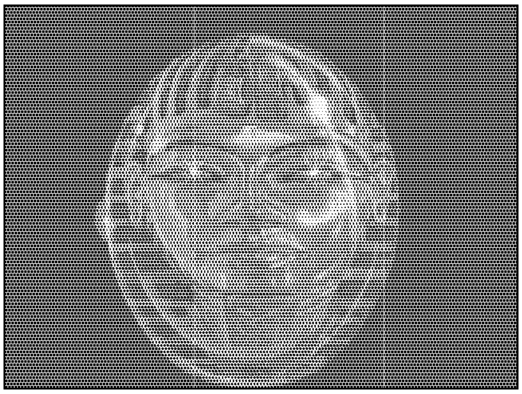


Figure 4: Example of an `IFF` or `LBM` file with 256 colors, printed with 16 grey shades and separation of raster points

`IFF` or `LBM` files are usually labelled by the extension `.lbm` in the MS-DOS environment, but BM2FONT also allows a file with extension `.iff` to be accepted as an `IFF` or `LBM` file.

2.5 TIFF files

The Tagged Image File Format of the Aldus company is one of the most commonly used standards for storing and interchange of graphics. Its support of black and white graphics as halftones and colors is probably one of the chief reasons for its popularity.

The following picture was originally a color photograph and is used to illustrate the quality of representing color pictures as halftones:

Figure 5: Example of a `TIFF` file with 16 grey shades, printed with 16 grey shades and separation of raster points

The most important item in this digitizing process is the number of colors available in the scanned image. It is even more important than the resolution.

2.6 IMG files

The GEM IMaGe file format (IMG) is yet another format. This format encodes the contents of the screen buffer by different methods of data compression. These include: runlength encoding, bitstreams, patterns, and repeated sequences The following example is made with the SIRGraph software, which is available in several different environments.

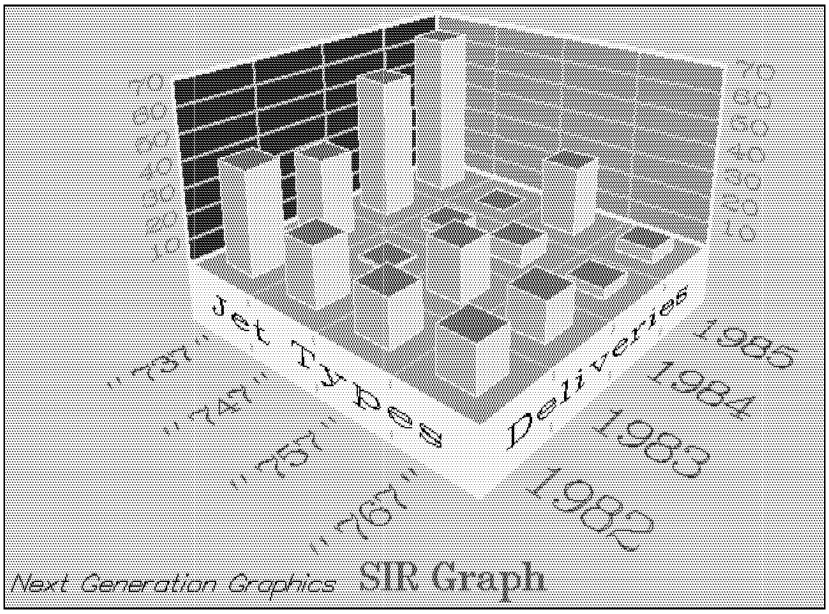


Figure 6: Example of an IMG file with 16 colors, printed with 16 grey shades, no error distribution used

The plan is to have BM2FONT run on several different platforms which include workstations, a mainframes, supercomputers, along with the current PC environment.

2.7 CUT files

The CUT format is used for storing images, among others, by the video system ImagePro (Dr. Halo). It uses a runlength encoding format. It may be necessary to reduce the height of the captured image, depending upon the hardware. BM2FONT uses a default x/y–ratio of the halftone pixels of 5/7. Of course, this can be specified on the command line if another parameter value is desired.

Figure 7: Example of CUT files with 256 grey shades, printed with 16 resp. 36 grey shades

2.8 BitMaps

The incorporation of bitmapped graphics should not be limited to these formats. There are many more and there is no reason to expect that other formats will not be developed. Also, if the available conversion programs can't produce a supported format, it may be able to produce a fairly "pure" bitmap that would not even have a header. BM2FONT does accomodate this but it may require the user to create a special decoding program.

BM2FONT is not able to process a pure bitmap file correctly without any accompanying information only by scanning the filename. The filename only says, that there is coming a bitmap, because the extension didn't match one of the mentioned formats like `.gif` or `.tif`. By using the parameter `l` like for example `-l320` BM2FONT assumes, that there comes a bitmap with a width of 320 bytes.

The following picture was generated with `bm2font venus.bit -l320 -gy -x4 -u2 -b2`. Originally it has been the file `venus.gif`, but the right border of the picture was not scanned as it should be. An extraction of the image was done which removed the damaged part of the picture. Then it was processed by BM2FONT.

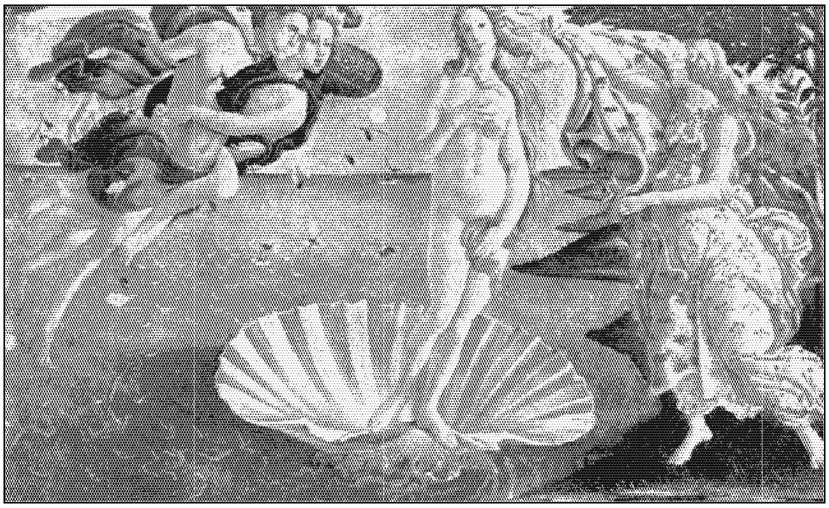


Figure 8: Example of a `BitMap` with 16 grey shades, printed with 14 grey shades

Each time a pure bitmap is processed, several parameters should always be included:

(`-gy`) indicates that halftone pixels are present,

(`-x4`) the width of the color sample (in bits),

(`-u2`) the size of the raster (`-u2`), and

(`-b2`) the grade of lightening.

3 The method of integration

The inclusion of graphics is accomplished by writing a text file (of TEX commands) and two files that contain font information that TEX and its variants understand. The TEX commands make explicit use of the font files. Further, the font may have several "characters" which represent parts of the picture. The TEX macros will assemble these characters to make the image.

BM2FONT wrote the following text file for the example of Figure 2. First, the new picture fonts are defined, then we get the width of the picture and put the parts of the picture into a box. A macro is defined to output this box, like a TEX table, paragraph, or other box. TEX gets the information about the dimensions of the characters, which were used to typeset the picture from the `tfm` file, the driver programs read the packed (`pk`) files files to print the document, including the picture.

```
\newbox\cobrabox
\newdimen\cobraw
\font\cobraa=cobraa at 72.27truept
\setbox\cobrabox=\vbox{\hbox{%
\cobraa\char0\char1\char2}}
\cobraw=\wd\cobrabox
\setbox\cobrabox=\hbox{\vbox{\hsize=\cobraw
\parskip=0pt\offinterlineskip\parindent0pt
\hbox{\cobraa\char0\char1\char2}
\hbox{\cobraa\char3\char4\char5}}}
\ifx\parbox\undefined
```

```
      \def\setcobra{\box\cobrabox}

\else

      \def\setcobra{\parbox{\wd\cobrabox}{\box\cobrabox}}

\fi

```

The maximum size of the characters which are 'tiled' together is 0.5 inch. These characters should be placed immediately adjacent to each other and there should also not be any interline space between rows of these tiled characters.

The driver program that convert's TEX's output (a `.dvi` file) into a printer image may have some slight problems that keep the 'tiles' in proper alignment. In the low resolution printers (600 dots/inch and less), the drivers should generally allow a slight accumulation of spacing errors in long words due to the fact that the exact width of characters are not whole multiples of the width of pixels. This accumulation is normally bounded by a `maxdrift` value. If it is not zero, then these tiles can overlap or have a gap between them.

If so, look around, there are several good drivers that will not have this problem. A single picture is converted into several characters because many printers have a small limit to the size of characters they can handle. The driver family from Nelson Beebe now handles this properly as does the `dvips` driver from Tom Rokicki. The size of this **M** is approximately the same as the individual tiles or parts of the picture that are created by BM2FONT.

Within a TEX installation there may be one or more directories from which TEX and the driver obtain font files. The output of BM2FONT must be placed in one of those directories or an environment variable must be set to let these programs know where these font files are placed. BM2FONT uses an environment variable in the PC environment to place the generated files in a proper directory. The environment variables are named `TEXFONTS`, `TEXINPUTS` and `DIRPXL`. The following example is based upon the default resolution of 300dpi:

Environment-Variable	File written by BM2FONT
`texinputs=\tex\inputs`	`\tex\inputs\cobra.tex`
`texfonts=\tex\fonts\tfm`	`\tex\fonts\tfm\acobra.tfm`
	`\tex\fonts\tfm\bcobra.tfm`
`dirpxl=\tex\fonts\pxl`	`\tex\fonts\pxl300\acobra.pk`
	`\tex\fonts\pxl300\bcobra.pk`

If these environment variables are not defined, BM2FONT outputs the font and TeX files in the current directory.

4 Variations by changing parameter values

We should seldom process pictures pictures with BM2FONT's default parameters. This is especially true if the pictures are photographs taken under varying conditions. The user should exercise some 'artistic license' in the choice of parameters to achieve the best effects. The list of the default parameters is given by BM2FONT when invoked without any parameters on the command line.

```
C:\>bm2font

This is BitMapTOfont, version 2.0 of january 93

Converting Bitmap Files to TeX-Fonts

 usage is BM2FONT filename and parameters

 -f<name of picture for TeX>     (std filename)

 -h<horizontal resolution>       (pixel/inch, std 300)

 -v<vertical resolution>         (pixel/inch, std 300)

 -l<length of mapline>           (in bytes, only pure bitmaps)

 -a<show pictures on screen>     (y or n, std n)

 -p<height of character 1 inch>  (y or n, std y)

 -e<stretch EGA pictures>        (y or n, std y)

 -i<inversion of pixels>         (y or n, std n)

 -g<greypixels in bitmap>        (y or n, std n)

 -w<let white be light grey>     (y or n, std y)

 -d<distribute errors>           (y or n, std y)

```

```
-s<separation of grey dots>     (y or n, std n)
-r<repeat each grey pixel>      (y or n, std n)
-u<pixels for grey rectangle>   (less 8)
-c<vert. pixels for rectangle>  (less 8)
-x<bits per sample>             (0 < x < 9)
-b<reduce halftone colors>      (f.e. by 1, less u*c*4, std 0)
-t<gradation value>             (in %, std 70)
-z<area of gradation>           (in %, std 70)
-m<width of picture on paper>   (in mm)
-n<height of picture on paper>  (in mm)
```

We **emphasize again** that the font files that BM2FONT creates from a picture should be for a specific device. Each type of printer is a different marking engine and has different characteristics. When we create fonts for TEX using METAFONT, we specifically state which printer the font is intended for. A popular list of modes for METAFONT has 15 different 300 dpi modes.

TEX is device independent, but the driver can't be! This is even more true with halftone pictures than it is with fonts. When previewing the document on a resolution of 100 dots per inch only the layout of the page can be interesting, not the picture itself. One cannot be confident of the appearance of the picture on a 300 or 1200 dpi device based upon the previewing of the 'font' on a screen's very low resolution. The user should be prepared to make several runs with different parameters to find the 'best settings' for each picture or set of pictures.

4.1 The name of the output files: Parameter -f

The default procedure is to use the part of the input file name before the extension as the root of the names of the output files. In a previous example, the input came from `cobra.gif` and the output files all had '`cobra`' in them. The parameter `-fsnake` would cause the string 'cobra' to be replaces with 'snake' in the names of the output files, both the `.tex` and font files.

4.2 Resolution: Parameters `-h` and `-v`

The default assumption in BM2FONT is a resolution of 300 dpi in both the horizontal and vertical directions. A pixel may have different positioning resolutions. If the pixels were placed at 300 dpi intervals horizontally and 240 dpi intervals vertically, then the parameters `-h300 -v240` would be used. Some of the more commonly used dot matrix printers have such an anomoly even though the pixels are essentially round (or square).

This information is used in calculating the `tfm`-dimensions which TEX will use. Thus, we can influence the relative size of the output font. The physical size of the picture from the tiled characters is also affected by several parameters, especially the raster size. The raster size and the printer's resolution must both be considered to get good results with halftones.

4.3 Usage of bitmaps: Parameter `-l`n, `-g`c and `-x`c

These parameters were introduced in the discussion of Figure 8. The width of the picture is declared as `-l`n bytes. This implies that the conversion that has written the bitmap has padded the rows of pixels to a multiple of 8 if the natural bitmap was not.

The parameter `-g`c is used to indicate that `y`*es* there are grey values of the pixels because a black and white picture is expected. Halftone pictures also need the specification of the color sample's width, which is expected to be

8 bits. An 8 bit width gives a possible 256 grey shades. A picture with 16 grey shades have a width of 4 bits. In the latter case, BM2FONT is to be called with parameter setting `-x4`.

4.4 Graphics adapter: Parameter `-a`

BM2FONT is written in a manner that the picture can be shown on the screen with most of the common video adapters. It is possible that there will be some problems and inconsistencies with different VGA cards. For this reason, this previewing or accompanying output to the screen is supressed, by default. This viewing is activated by the parameter `-ay`. In this mode, BM2FONT waits after the picture is displayed on the screen for a key to be pressed, before continuing. The user may enter the `q` key which will cause the program to stop and not delete temporary files which have the extension of `.tmp`.

4.5 Scaling: Parameter `-m` and `-n`

The user is sometimes surprised by the size of the picture that results from BM2FONT's processing. Scaling is possible by the use of these parameters. The parameters `-m` and `-n` are followed by numeric values which set the width and height of the resulting picture (in the font representation) in units of millimeters. Thus, multiple pixels may be collapsed into one and the decision of how this is done is based upon the size of the original picture in pixels, the resolution of the printer, and these parameters.

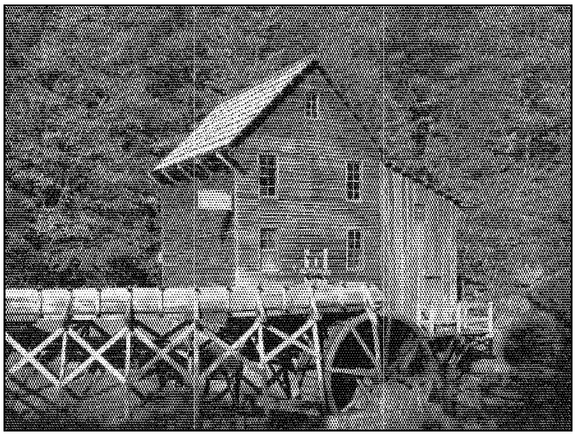


Figure 9: Example of a scaled picture with a width of 75mm, 15 greyshades

The image above was genereated with parameter `-m75`. It was scaled from 1024x768 pixels to 443x332 pixel. The scaling process is done by repeating or deleting pixels in the simplest way and has **significant** influence on the quality of a picture. In many cases, like photographs, the change of quality can be ignored. But in case of black and white pictures the problems should not be ignored, for example:

The picture was reduced to a width of 40mm

Two color graphics should be scaled to the desired size before using BM2FONT. This will prevent effects like staircasing or lines with different widths. Text within the original graphic may or may not be readable after the scaling process.

The program **HP2XX** converts HPGL-files (and some others) into the **PCX** format. This is typical of the programs that should be used before BM2FONT. The user can then determine the size of the graphic.

4.6 Width of the raster: Parameter `-u` and `-c`

Halftone pictures cannot be made by converting one pixel from the original picture into one pixel on the printer. One pixel in the original picture may be converted to, say, four pixels on the printer. Assume these four pixels form a square. Then, all four of these printer's pixels are white when the pixel in the original picture is white and all four are black when the original pixel is black. All the stages in between will reflect the halftones or levels of grey that give a picture texture. This method is called dithering or rasterization. The quality of a halftone picture on paper has a direct dependence upon the resolution of the output device. Higher resolutions allow more pixels to be in a 'grey pixel' giving more 'grey levels.'

The next level of description will be based upon BM2FONT's default output device resolution of 300 dpi. Of course the horizontal and vertical resolution can be set with the `-h`n and `-v`n parameters, respectively. The user sets the size of the grey pixel with the with the `-u`n and `-c`n parameters. These values refer to the number of device pixels (wide and high, respectively) in one-fourth of each grey pixel. Is `-u3` is set, then the grey pixel will be a square of six of the device's pixels in both directions. The height parameter defaulted to the horizontal value since it was not set. This 6×6 pixel will have a size of approximately 0.5 mm for the 300 dpi device.

The following dither matrix is used as a basis. The numerals within the matrix represent the grey values, for which the pixels must be blackened at the corresponding positions within the chosen grid. Original grey values are transformed to the available halftones.

	Column						
Row	1	2	3	4	5	6	7
1	01	03	06	10	18	26	38
2	02	04	07	12	19	28	40
3	05	08	09	14	21	30	42
4	11	13	15	16	23	32	44
5	17	20	22	24	25	34	46
6	27	29	31	33	35	36	48
7	37	39	41	43	45	47	49

A grey dot is built up by 4 neighboring original colored pixels. This means, that every intensity level has 4 different occurences, following called patterns. The amount of available grey levels G so is $G = 4uc$, where u is the width and c is the height of the dot raster in pixels. When using the above mentioned parameter setting we will have 36 levels of grey intensities and each level has 4 rasters, that are used by BM2FONT to compose the picture.

Grey level	Patterns			
	1	2	3	4
1	●○○ ○○○ ○○○			
2	●○○ ○○○ ○○○	○○○ ○○○ ○○●		
3	●○○ ○○○ ○○○	○○○ ○○○ ○○●	○○○ ○○○ ●○○	
4	●○○ ○○○ ○○○	○○○ ○○○ ○○●	○○○ ○○○ ●○○	○○● ○○○ ○○○
5	●○○ ●○○ ○○○	○○○ ○○○ ○○●	○○○ ○○○ ●○○	○○● ○○○ ○○○
6	●○○ ●○○ ○○○	○○○ ○○● ○○●	○○○ ○○○ ●○○	○○● ○○○ ○○○

⋮

Grey level	Patterns			
	1	2	3	4
20	●●○ ●●○ ●○○	○○● ○●● ○●●	●○○ ●●○ ●●○	○●● ○●● ○○●
21	●●● ●●○ ●○○	○○● ○●● ○●●	●○○ ●●○ ●●○	○●● ○●● ○○●
22	●●● ●●○ ●○○	○○● ○●● ●●●	●○○ ●●○ ●●○	○●● ○●● ○○●
⋮	⋮	⋮	⋮	⋮
34	●●● ●●● ●●●	●●● ●●● ●●●	●●○ ●●● ●●●	●●● ●●● ○●●
35	●●● ●●● ●●●	●●● ●●● ●●●	●●● ●●● ●●●	●●● ●●● ○●●
36	●●● ●●● ●●●	●●● ●●● ●●●	●●● ●●● ●●●	●●● ●●● ●●●

Depending on the actual position in a grey pixel, as it is generated for the output device, the corresponding raster of the grey value is chosen. During rotation around the center of the grey pixel the four segments are used in the order 2, 3, 1 and 4.

For each row of the picture the patterns r_i are used as the following table shows:

r_1	r_4	r_1	r_4	r_1	r_4	...
r_2	r_3	r_2	r_3	r_2	r_3	...
r_4	r_1	r_4	r_1	r_4	r_1	...
r_3	r_2	r_3	r_2	r_3	r_2	...
r_1	r_4	r_1	r_4	r_1	r_4	...
r_1	r_4	r_1	r_4	r_1	r_4	...

4.7 Gradation: Parameters `-t` and `-z`

It is necessary to compensate for the fact that most output devices will make dark grey too dark. This is due to the fact that 'write black' marking engines write overlapping pixels that have a larger diameter than their spacing. BM2FONT uses *gradation* to correct these 'tones.'

The parameters **`-t`** and **`-z`** are used to communicate the values of parameters which specify the range and amount of of lightening. These parameters are percentages and their use is illustrated in the heuristic:

$$y = \begin{cases} \frac{1}{2x_0^\alpha} x^{1+\alpha} + \frac{1}{2} x_0^\alpha x^{1-\alpha} & 0 \leq x < x_0 \\ x & x \geq x_0 \end{cases}$$

where $\alpha = \frac{t}{100}$, $x_0 = \frac{z}{100}$

This gives a more intensive lightening near $x = 0$, which is 'black.' The lightening disappears for $x \geq x_o$. The default value of 70 for both **`t`** and **`z`** causes a slightly lightened picture, that has effects within 70% of the grey scale.

```
bm2font parrot.gif -t70 -z70
```

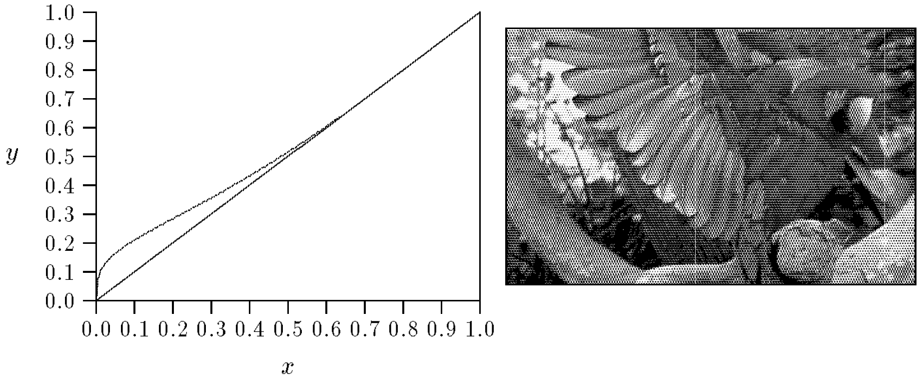


Figure 10: Example of gradation with default parameter values; generation of the picture using 16 grey levels

In this plot, remember that the larger values of x and y are 'lighter' values or further from 'black.'

A more intense lightening over 90% of the grey scale causes a visible difference in the curve describing the heuristic and in the picture, which is the desire.

```
bm2font parrot.gif -t80 -z90
```

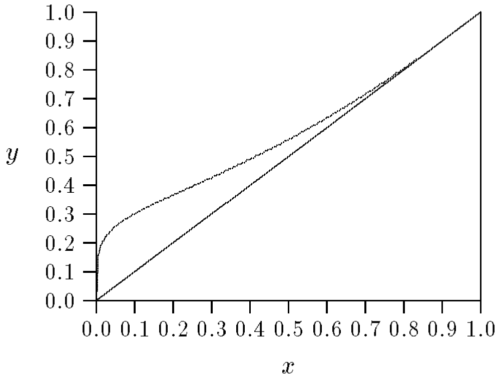


Figure 11: Example of gradation with changing of lightening and range of effect; generation of the picture using 16 grey levels

If lightening is done in the processes before delivering a bitmap to BM2FONT, then the user should furnish the parameter `-t0`. Also, if a picture is furnished with a small number of grey levels, say 8 or 16, gradation should probably be avoided.

4.8 Global lightening: Parameters `-b` and `-s`

The `-b` parameter is similar to the `-z` parameter in that it specifies the range of lightening. It does this in a quite different manner. The `-z` specified the percentage of the darker range of the grey scale that is to be lightened according to the formula. The use of `-b3` in a process with 36 grey levels will cause the output grey scales to be distributed over the lower 33. Thus, the three darkest will not be used.

The lightening of a picture by using this parameter setting can cause some information to be lost. If a picture is "too dark" then it's better to change contrast or brightness when capturing the picture with an optical system like a scanner or a camera. If this cannot be done because the original is not available, then the user should experiment with these parameters.

Most digitized pictures do not use the whole available color scale. BM2FONT tries to expand the calculated intensity levels to all of the available grey scales, because we usually have fewer scales available on the output side. This can cause harsh contrasts in the dithered picture. If the program notices this fact it will give a message. In this case a lightening with `-b` may give a better quality of the picture.

The `-sy` parameter is another parameter which can be used to give some lightening. In this case, it is used to 'separate' 'grey pixels' by rows and columns of white pixels. This is needed if the printer does not give sufficient separation. Use of this parameter will cause the picture to be enlarged.

4.9 Error distribution: Parameter `-d`

We have already mentioned that the quality of a picture often decreases due to the fact that color pictures often have more intensity levels than the number of output grey levels.

The Floyd/Steinberg algorithm is an effective method to solve this problem by distributing the rounding errors to the neighboring pixels cumulatively. The basic idea of this solution is incorporated into BM2FONT by distributing the different levels of intensity onto different patterns that have the same grey level. The loss of information in this rounding process may be seen as hard changes of grey intensities. BM2FONT considers the mean value of neighbouring pixels with different colors. This produces a different scaling into the available grey scale. This usually produces a distinctly better quality of the image. Here is an example, in which about 4% of the grey values have been altered by error distribution.

Image was done with error distribution

Same picture without error distribution

BM2FONT's default is is process all bitmaps with this error distribution. The parameter `-dn` turns this off.

4.10 Other matters: Parameter `-i`, `-w` and `-r`

The color black is normally represented by zero in most palettes. The `-iy` parameter inverts the scale, just in case a picture has appeared in this format.

Most halftone's are more pleasing if truly white areas are shown as a light grey. This is the default for BM2FONT and can be supressed by use of the parameter `-wn`.

The normal mode of BM2FONT includes the representation of each pixel by four 'grey pixels.' This might still produce too small an output on high resolution output devices, especially when working with relatively small input images. The parameter `-ry` will increase the picture's size by repeating each grey pixel horizontally and vertically without changing the raster size.

4.11 Suppressing the screen correction: Parameter `-e`

It is expected that many of the bitmaps that will be included in documents will be created, proofed, and stored on workstations. These workstations (particularly PCs) have a wide range of aspect ratios that the user may not have considered when creating the bitmap. For example, a graphic created using an EGA screen with (640×350) pixels will not appear the same on a VGA screen with (640×480) pixels. Further, the physical dimensions of two screens with the same pixel configurations may be different.

The same problem will occur when the image is transcribed to a hardcopy. BM2FONT can use the `-en` parameter to cancel the default adjustment of the pixel aspect ratio. The `-u` and `-c` parameters were mentioned earlier and can be used to achieve the same effects.

4.12 Size of the graphics characters: Parameter `-p`

When choosing `-pn` BM2FONTwrites characters with a width and height up to half an inch into the font. The default limit of width and height is one inch. When using smaller characters there are four times as many characters created. As the maximum number of characters in one font still is 128 — because of old driver programs and implementations of TEX with version number 2.x — it may be that there are generated more fonts than really necessary. Nevertheless it may be usefull to choose smaller characters when producing halftone pictures for low resolution devices because of the encoding method used in `pk` files.

Another reason for using smaller characters with `-pn` may be an available output device with a restiction concerning the size of downloadable characters.

Here the hint, that BM2FONTgenerates the pk files up to a size of 64KB. This limit is still necessary to avoid problems on some sort of machines, running an operating systems, which does not know how to manage available RAM.

Earlier versions of BM2FONT*either create fonts in pk or pixel format depending on that parameter*

5 An example for using BM2FONT

A example showing the effects of different values for some of the parameters is considered a necessary part of the manual. The values of the parameters used in creating the font files used in this manual were set for a typical 300 dpi, write black printer. We make no claim that they are the best settings for this printer and certainly we don't make the claim for your printer.

We strongly recommend that the user experiment with the conversion for their own installation. We would be pleased to learn of generalities that you observe in your use of the system.

The following picture is of a well known building and this picture will be shown using some different parameters.

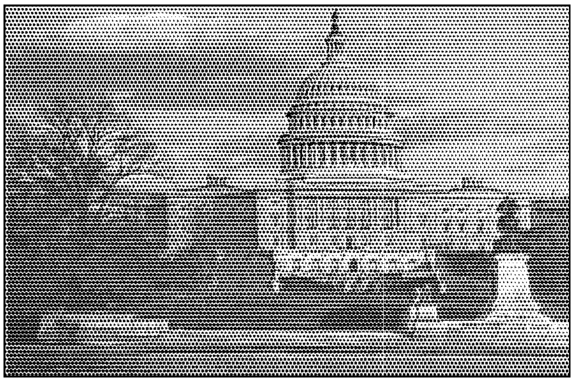


Figure 12: Picture of the U.S. Capitol, printed with 36 grey shades, gradation increased with effect on the whole grey scale

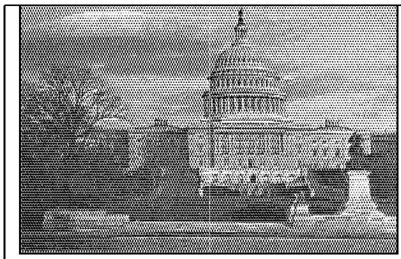
This picture was selected because it includes some specific problems concerning transformation to a halftone image. This choice will allow the user to see the noticeable results of parameter choices without requiring a magnifying glass. It is a nice coincidence that it is a symbol of the country were TeX was created.

These additional copies of this picture of the U.S. Capitol were printed using

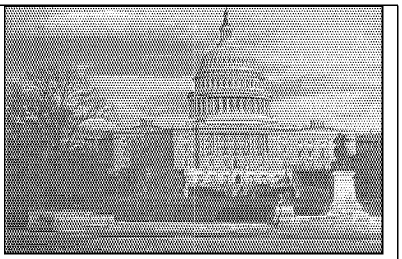
16 grey levels. The parameters that were used are given under each figure. The halftones were lightened on each of the pictures except for the last one.

The effects of using error distribution are not included for reasons of economy of space and the size of the distribution of this code and document. We recommend that the user do this as a test to gain familiarity with the system. The command line invoking BM2FONT is shown below and should be understood to add the individual parts shown below each figure:

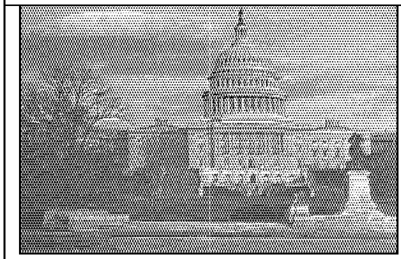
```
bm2font capitol.295 -l295 -gy -x8
```



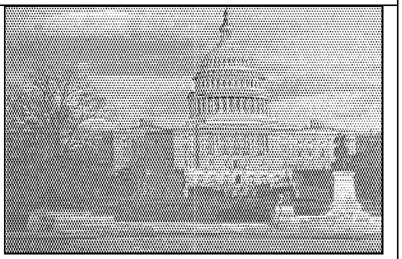
default



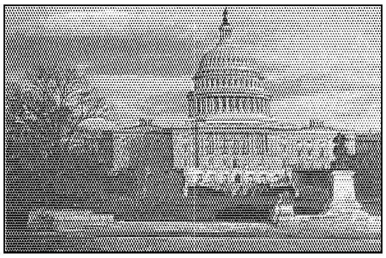
`-b3`



`-t80 -z100`



`-t80 -z100 -b3`



`-b3 -t0 -wn`

6 Hardware and Software

BM2FONT runs on PCs and compatibles under the MS–DOS operating system. We know of no exceptions and the generated output is compatible to all versions of TEX. The VGA, EGA and Hercules graphics adapters are supported. In case of problems concerning graphics adapters, the option `-ay` should not be used. In case of permanent errors please contact:

Heinrich-Heine-Universität Düsseldorf
Universitätsrechenzentrum
Friedhelm Sowa
Universitätsstraße 1
4000 Düsseldorf 1
Federal Republic of Germany
Telephone: 0211 / 311-3913
Telefax: 0211 / 311-2539
Email: tex@ze8.convex.rz.uni-duesseldorf.de or
sowa@convex.rz.uni-duesseldorf.de

The English version of this manual is maintained by:

Bart Childs Department of Computer Science Texas A&M University College Station, TX 77843-3112 Telephone: 409-845-5470 Telefax: 409-847-8578
Email: bart@cs.tamu.edu or bart@tamzeus.bitnet

We would both like to know of efforts to port this code to different platforms. Further, we would like to know of the TEX archives where this is made available to the public.