

Using HP2XX

A HP-GL Converter

Edition 1.4, for HP2XX version 3.4.4
June 2003

by Martin Kroeker (previously by Heinz W. Werntges)

mk@daveg.com
Using HP2XX, Revision : 1.4
T_EXinfo 2005-03-21.17

Copyright © 1998 - 2003 Martin Kroecker Copyright © 1992 - 1994 Heinz W. Werntges

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the accompanying file named COPYING which contains the “GNU General Public License” is included exactly as in the original, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that the abovementioned file COPYING containing the “GNU General Public License” may be included in a translation approved by the Free Software Foundation instead of in the original English.

Short Contents

1	Introduction	1
2	Basics	3
3	Advanced subjects	9
4	Installation and modification notes	13
A	Known HP-GL commands	17
B	Option summary	21
C	Acknowledgements	25

Table of Contents

1	Introduction	1
1.1	Invoking <code>hp2xx</code>	1
1.2	<code>hp2xx</code> for the impatient	1
2	Basics	3
2.1	Modes of <code>hp2xx</code>	3
2.2	Sizing your output	3
2.3	Pen sizes and colors	3
2.4	Selecting a page	4
2.5	Vector formats	5
2.6	Raster formats	5
2.7	Printer formats	6
2.8	Preview	6
2.9	Misc. options	7
3	Advanced subjects	9
3.1	The coordinate range	9
3.2	Fixed scaling	9
3.3	Scaling to true size	9
3.4	Swapping	10
3.5	Dots and lines	10
3.6	Unsupported formats	10
3.7	T _E X formats	11
4	Installation and modification notes	13
4.1	Installation procedure	13
4.1.1	Installing an executable version	13
4.1.2	Source-level installation	13
4.2	Adding your own formats	14
4.3	Future improvements	14
4.4	Font coding	15
Appendix A	Known HP-GL commands	17
Appendix B	Option summary	21
B.1	General options	21
B.2	Size controls	22
B.3	Raster format controls	23
B.4	PCL specifics	23
B.5	TIFF specifics	23
B.6	TIFF specifics	23
B.7	Margins	24
B.8	Preview (DOS/PC's only)	24
B.9	Help	24

Appendix C	Acknowledgements	25
C.1	Acknowledgement	25
C.2	Copyright notice	26

1 Introduction

The **hp2xx** program is a versatile tool to convert vector-oriented graphics data given in Hewlett-Packard's HP-GL plotter language into a variety of popular both vector- and raster-oriented graphics formats.

The various supported output formats include Encapsulated PostScript (EPS), PCX, IMG, and several formats intended to facilitate the generation of graphics within T_EX documents. In addition, **hp2xx** output is printable on the HP Laserjet/Deskjet printer series and some Epson Stylus printers, and it may be used as a HP-GL previewer on many platforms, e.g. X11, OS/2, MS Windows and plain old DOS (VGA).

hp2xx first converts all HP-GL data into pure vectors and buffers them internally. It then converts these vectors into a specified output format (vector modes), or rasterizes them (raster modes) on an internal bitmap. In raster modes, **hp2xx** then translates the bitmap into the output format.

1.1 Invoking hp2xx

The format of the **hp2xx** command is:

```
hp2xx [options] [input-file/s]
```

It follows the UNIX System V tradition of a filter, i. e., options begin with '-', followed by a single letter and an optional parameter. Options must appear immediately behind the program name and before the input file name(s) (if specified). If no input file is given, **hp2xx** reads from **stdin**. In addition to this traditional option handling, **hp2xx** also supports GNU-style long options and option/non-option permutation (see [Appendix B \[Appendix B\], page 21](#)). However, throughout this manual all examples will only display short options.

hp2xx writes to the output file whose name can be specified by option '-f'. Without option '-f', **hp2xx** generates output file names from the input names and the selected mode (see option '-m'). **hp2xx** writes to **stdout** if you supply a dash as output file name like in '-f-'.

1.2 hp2xx for the impatient

This section is intended to give those of you a quick-start who are quite familiar with traditional UNIX-style programs and with HP-GL and other graphics formats. The following examples will give you a good idea of **hp2xx**'s functionality. See [Appendix B \[Appendix B\], page 21](#), the Option Summary, for further details.

```
hp2xx foo.hp
```

Preview of HP-GL graphics in file 'foo.hp'. The picture will fit into a square of 200 mm width, assuming that your output device (screen) features 75 DPI resolution (default).

```
hp2xx -q -d86 -h160 -w220 foo.hp bar.hp
```

Multiple-file preview. Option '-q' puts **hp2xx** into "quiet" mode. The picture will fit into a rectangle of 220 mm width and 160 mm height, assuming a 86 DPI resolution of the output device (screen).

```
hp2xx -t -c12340567 -p12230412 foo.hp
```

Preview, size according to original HP-GL data (as on a plotter), with different pen colors and sizes. Color and width according to:

Pen #	:	Color code	/	Size (pixel)
1	:	1 (black)	/	1
2	:	2 (red)	/	2
3	:	3 (green)	/	2

```

4   : 4 (blue)      /   3
5   : 0 (background) /   0
6   : 5 (cyan)      /   4
7   : 6 (magenta)   /   1
8   : 7 (yellow)    /   2

```

```
hp2xx -m eps -l a.log -h100 -w150 -p542 foo.hp bar.hp
```

Encapsulated Postscript mode. Files ‘foo.eps’ and ‘bar.eps’ will be created. The diagnostic output will be written into ‘a.log’, so `hp2xx` works quietly. Both EPS pictures will fit into a rectangle of size 150 x 200 mm. The size (width) of pen 1: 0.5 mm, pen 2: 0.4 mm, pen 3: 0.2 mm, pen 4 ... pen 8: 0.1 mm.

```
unix% cat foo.hp | hp2xx -m pcl -o30 -o50 -i -F -f- | lpr -P ljet
```

In this generic UNIX example, `hp2xx` reads HP-GL code from `stdin`, converts it to HP-PCL which is suitable for direct output on any HP Laserjet printer, and pipes the output via `stdout` into the appropriate printer queue. Option ‘-f-’ forces `hp2xx` to write to `stdout` instead of a file, ‘-i’ initializes the printer before the output, ‘-F’ sends a FormFeed at the end of output. There will be (additional) 30 mm left and 50 mm top margins. 75 DPI are assumed per default.

```
hp2xx -m pcx -f foo3.pcx -d300 -h80 -w150 -r90 -P2:4 foo.hp
```

PCX mode. Output goes into file ‘foo3.pcx’. A limiting rectangle of 150 x 80 mm at 300 DPI is assumed. The picture will be rotated by 90 degrees. Only pages 2 to 4 of the multi-page HP-GL source is used (each occurrence of HP-GL code `PG`; increments the internal page counter).

2 Basics

This chapter provides you with almost anything you'll need for successful **hp2xx** applications. You'll probably soon operate **hp2xx** by solely consulting the option summary (see [Appendix B \[Appendix B\], page 21](#)) or just by calling **hp2xx** without any parameters to obtain its built-in option summary.

2.1 Modes of hp2xx

The mode switch '**-m string**' tells **hp2xx** about the mode it should use to generate output, i.e., the desired output format. **hp2xx** can run in three different groups of modes: Generating vector graphics, raster graphics, or "preview mode", i.e., displaying the graphics. Preview is the default; see [Appendix B \[Appendix B\], page 21](#) for a list of all modes.

2.2 Sizing your output

NOTE: The basic unit length within **hp2xx** is mm (millimeter). This unit is always assumed except where noted otherwise.

In contrast to a real HP-GL plotter, **hp2xx** lets you decide freely about the size of the picture. While preserving the picture's aspect ratio, **hp2xx** will fit the picture into a window of width w and height h . By default, $w = h = 200$ (mm). Set these basic sizes using options '**-w w**' and '**-h h**'.

Sometimes you'll want to change the aspect factor of a picture, e.g., to spread out a square picture into landscape. Option '**-a af**' is used for this. $af > 1$ increases x/y ratio, $0 < af < 1$ decreases x/y .

Example: Let's assume your picture covers a native coordinate range of 100...900 plotter units in x direction and 200...600 in y direction. Thus, its width is double its height. Using defaults, **hp2xx** will create a picture of size 200 x 100 mm, while options '**-w 100 -h 40**' will lead to a picture of size 80 x 40 mm, and '**-w 100 -h 40 -a 0.5**' results in a 40 x 40 mm picture.

Alternatively, ignore explicit size control and rely on the true HP-GL coordinates (and therefore: sizes) of the given input file(s). Flag '**-t**' inhibits '**-a -h -w**' and lets **hp2xx** use true HP-GL sizes, based on the assumption that 1 HP unit = 1/40 mm.

Some modes of **hp2xx** support page offsets, i.e., left and upper margins added to the picture, probably in addition to some hard margins which cannot be avoided. Currently, these modes are *eps*, *pcl*, and *pre*. The left margin (offset) is modified with option '**-o off_left**', while the upper margin can be controlled via '**-O off_upper**'. *off_left* and *off_upper* are specified in mm.

Option '**-C**' (center the picture) works in combination with options '**-o -O**' by modifying the offsets in such a way that the resulting picture is centered within the frame defined by options '**-w -h**'. Example: Assume that the picture is a square, and '**-w 100 -h 40**' are specified. A 40 mm square with an additional left margin of $(100-40)/2 = 30$ mm will be produced if option '**-C**' is given. If the actual width/height ration of the picture exactly matches the ratio defined by '**-w -h**', option '**-C**' has no effect.

2.3 Pen sizes and colors

Imagine a plotter with a pen carousel, e.g., like the model HP7550A. The carousel carries a (small) number of pens. Their colors and tip thicknesses (sizes) are selected by a human operator, while the plotter only receives commands like "Now use pen number 5". If you don't provide a pen, the plotter will move and "draw" without this pen if its number is selected.

hp2xx emulates a carousel of up to 8 pens of various colors and sizes. By default, all pens are present, have *foreground* color (typically *black*), and their tip thicknesses are one unit (1/10

mm). (NOTE: in versions before 3.4.2, pen widths in raster modes were counted in pixels, so the default in those versions was 1/10 mm for vector modes, 1 pixel for raster modes).

HPGL/2 commands NP,PC,PW may override this for up to 255 freely configurable pens.

Pen colors and sizes are represented by digits to allow for a compact option list. There are 8 colors including *background* (usually white). See [Appendix B \[Appendix B\], page 21](#), the Option Summary, for a list of all colors. E.g., color 3 is green, and color 7 means yellow. Permitted pen sizes are 0 ... 9 units in versions before 3.4.3, or up to 3.5mm (using characters A to Z to denote the range from 1.0 to 3.5) in later versions. WARNING: In versions of **hp2xx** before 3.4.3, all raster mode pen sizes were approximated by double, triple, etc. width, and lines wider than 4 would normally not give useful results. In 3.4.3, the line drawing algorithm has been substantially improved and should render accurately scaled pen widths.

Options ‘**-c c-string**’ and ‘**-s s-string**’ tell **hp2xx** about the pens to be placed in the carousel. *c-string* and *s-string* are strings of 1 to 8 digits, corresponding to special choices of pen 1 to 8. Defaults are *c-string* = *s-string* = ‘11111111’ unless the *hpgl* file contains corresponding PC and/or PW commands. Specifying ‘**-c**’ or ‘**-s**’ overrides the equivalent HPGL/2 command. If you specify less than 8 pens, the remaining pens keep their defaults.

Examples:

```
hp2xx -p13 foo.hp
```

Show a preview of ‘**foo.hp**’, drawing all lines with pen #2 three pixels wide instead of default 1 pixel, which applies to all other pens.

```
hp2xx -c12740 -p12230412 foo.hp
```

Here, pen #5 is “removed”. Pens #1 and #7 keep their default sizes, all others are set to various sizes. Pen #2 is red, #3 is yellow, and #4 is blue, while all other pens keep ‘**foreground**’ color, e.g., black.

If your HP-GL file contains the PC and/or PW commands introduced with HP-GL/2, these are interpreted to allow up to 255 pens. The special case where a pen is redefined to take on different colors is currently only supported in PostScript or PDF output mode. In all other modes, the last PC command encountered determines that pens’ color for the entire plot.

2.4 Selecting a page

There is a HP-GL command named ‘**PG;**’ which amounts to a FormFeed. Thus, there are multi-page HP-GL sources. While **hp2xx** was designed for just one output picture per input file, there is a simple way to cope also with multi-page sources:

hp2xx keeps track of the number of encountered ‘**PG;**’ commands. All code up to the next (if any) ‘**PG;**’ command is considered a single page. Pages are counted, starting at 1. You can ask **hp2xx** to ignore all HP-GL commands other than on page *n* with option ‘**-P n**’, effectively filtering out any one-page graphics. Sometimes, converting a whole page range makes sense, too. Therefore, **hp2xx** also accepts page ranges via ‘**-P n1:n2**’. The default is ‘**-P 0**’ which selects all pages. The output will be split into appropriate files whose name is derived from the base name of the input file by appending the frame number before the filetype extension. In preview mode, each page will display as a separate image - in contrast to versions earlier than 3.4.3, which would render everything in one image.

WARNING: Some HP-GL sources may start with a ‘**PG;**’ so the first page of your graphics may be 2 instead of 1. Look for the number of encountered pages in the diagnostic output if you miss the expected page! If the detected coordinate range shows unreasonable numbers like 1e10, you’ll be probably looking at an empty page.

2.5 Vector formats

All HP-GL graphics are decomposed by **hp2xx** into elementary move and draw commands. Selecting a vector mode essentially defines the conversion rules of such commands into specific formats.

The most popular and versatile vector format currently is Encapsulated PostScript (EPS). (In fact, it is much more than just a vector graphics format, but **hp2xx** uses only EPS's line drawing features.) Many programs allow importation of EPS files, and PostScript gives excellent printing results, so `'-m eps'` is highly recommended. Recently, Adobe's Portable Document Format (PDF) has mostly taken its place for formatted online documentation. For Web-based applications, the Scalable Vector Graphics (SVG) format, a W3C standard, holds some promise as a non-proprietary, XML-based format for vector images, not only on web pages. In the engineering community, the 2D DXF format originally used by AutoCAD has become a common denominator for the exchange of vector data. In the machine-tooling community, a standardized vector format known as G code is widely used (though machine-specific limitations and extensions make it not truly universal). Its implementation in **hp2xx**, selectable via the `'-m nc'` option, should be suitable for hobbyists wishing to do engraving, but it would not be wise to use it in professional machine tooling.

Unix users may also want to use either **Gnuplot** or especially **XFig** to annotate their plots. The native formats of these programs are directly supported via the `'-m gpt'` and `'-m fig'` options.

The Enhanced Metafile Format was originally tied to the Microsoft Windows environment, but it has recently found more wide-spread use as one of the few vector formats supported by StarOffice and OpenOffice.

Currently, all other supported vector formats represent various compromises to persuade **T_EX** or **L_AT_EX** into the generation of graphics. See [Section 3.7 \[T_EX formats\]](#), page 11, if you are specially interested in **T_EX**.

2.6 Raster formats

Raster graphics are probably the most widely used graphics by now. Many publishing programs accept rasterized graphics. It's likely you'll use **hp2xx** primarily in some raster mode.

In addition to vector modes, all raster modes need the desired resolution of an assumed underlying pixel grid to plot on, i. e., the number of pixels per unit length within that grid. A traditional measure is the number of "dots per inch" (DPI). **hp2xx** makes an exception from its usual unit length **mm** and lets you specify the traditional DPI values. Option `'-d num'` affects both x and y direction, *num* being the DPI value (an integer). If `'-D num_y'` is also specified, *num_y* will override the *num* DPI value, but only for the y direction.

Some programs were found, which generate HPGL output with too tight clipping bounds, which lead, for example, to some parts of text characters clipped off. Use option `'-e num'` to add some extra amount of space to clip areas to work around such problems. For example, `'-e 40'` will add 40 extra plotter units to every side of the clipping box, which corresponds to 1 **mm** in true size.

There are plenty of raster formats on various platforms, much more than **hp2xx** will ever handle. The supported raster formats IMG, JPG, PBM/PPM, PCL, PCX, PNG and TIFF were chosen for their widespread use, their simplicity, for actual demand, and for accessibility of specifications. If your desired format is not supported, look for a converter. E.g., the Portable Bitmap (PBM) project and more recently, the ImageMagick package offer quite a variety of such converters.

Please note that the preview mode (which does not create any output file) is a special raster mode. Instead of going into some output file, the internal bitmap is transferred into display memory. Therefore, the above considerations apply also to preview mode.

2.7 Printer formats

Currently, there is only one well-tested printer format (not counting ‘*eps*’, which is printable on PostScript printers): ‘*pcl*’, which stands for HP PCL Level 3. Essentially it is a raster format, but it comes with a few restrictions and additional options which correspond to printer properties. It prints on HP Laserjet and HP Deskjet series printers and compatibles.

The restriction concerns the resolution (DPI) during rasterization. Due to printer hardware limitations, only ‘*-d 75*’, ‘*-d 100*’, ‘*-d 150*’, and ‘*-d 300*’ are recommended (*hp2xx* will emit a warning for other values, but will try to create the desired output); option ‘*-D*’ must not be used. Some recent printer models do support 600, 720 or even higher resolutions, but even for those the requirements both in terms of cpu load and file size may be prohibitive at least on older systems.

There are two flags which may be useful if the output goes directly to a printer: Option ‘*-i*’ initializes the printer and tries to select the required paper format before the output starts, and ‘*-F*’ sends a FormFeed (ASCII 12) after the output.

For the HP Deskjet printer series, there is support of some “special” commands; ‘*-S 1*’ activates these. There is a limited support of color modes available, too: For the DJ500C and newer models, *hp2xx* can generate both CMY-based and CMYK-based color output (if colors are used: see option ‘*-c*’). Supply option ‘*-S 3*’ for CMY color mode, and option ‘*-S 4*’ for CMYK color mode (for the DJ550C). The Deskjet modes automatically invoke TIFF compression (mode 2), while pure PCL Level 3 does not know about compression. NOTE: Some recent low-end Designjet plotters (if ordered without the HPGL processor board) and probably also some models of Deskjet use a sparsely documented variant called PCL3GUI which appears to differ at least in the initialization sequence used. Currently *hp2xx* will not work well – maybe even not at all – with such printers.

A driver for the Esc/P2 raster language used by the Epson Stylus series of inkjet printers has been added in *hp2xx-3.4.0*. This should work at least for the small-format printers at 720 dpi.

2.8 Preview

Preview mode is *hp2xx*’s default. Its use prior to all other conversions is recommended since it offers a good impression of your final results. Functionally it is also a raster mode.

Depending on your hardware and operating system platform, *hp2xx* uses one of a variety of preview modules. On GUIs, a window containing the graphics will pop up, while on other systems the whole screen may be used for preview. You can control the position of a preview window via options ‘*-o*’ and ‘*-O*’ in a natural way. In full-screen previews, unused spaces are padded to the right and bottom with background color.

Since there is no way for *hp2xx* to predict the actual size and resolution of your preview device, e.g., screen, you may have to gauge *hp2xx*’s preview mode (using options ‘*-whdD*’). For example, if your device effectively works at 86 DPI and offers an active area of 24 by 18 cm,

```
hp2xx -d86 -w240 -h180 foo.hp
```

will make maximum use of your screen area and give you correct sizes. Since a single gauge will do for all future calls, you’ll probably want to create some one-line batch file for invoking *hp2xx* in preview mode, correctly gauged for your screen.

Depending on page offsets and the selected sizes and resolutions, a preview may not fit on your screen. In that case, some preview modules simply clip the picture; others give a warning but let you continue (DOS), and others simply terminate.

The X11 viewer allows to pan the visible area using mouse button one (which usually corresponds to the left button), and supports one level of zoom around the current cursor position, selectable with button two (middle button, or right button on two-button mice).

DOS users: Most VGA cards offer high-resolution modes (SVGAs). Unfortunately, there is no software standard for these modes. **hp2xx** lets you utilize these modes anyway with just a little help from you. Tell **hp2xx** the so-called mode byte of your favorite hi-res mode via option `-V num`. Since **hp2xx** issues only standard BIOS calls for mode switching, setting of color look-up table entries, and pixel drawing, chances are good that your VGA card's hi-res modes will work!

WARNING: You can damage your hardware by specifying inappropriate VGA modes! Generally you'll need a monitor which can sync on the horizontal frequency of the selected VGA hi-res mode, e.g., a multi-scan monitor. In case of doubt, switch off your monitor immediately!!

2.9 Misc. options

hp2xx features an on-line options summary. Invoking **hp2xx** with option `-H`, or with any illegal option or without any parameter, will display about 2 pages of text. (Note: I'd have preferred option `-h` for on-line help, but this option is needed by the indispensable *height* parameter.)

The list of supported output files (displayed for the `-m` parameter) shown in the help text always corresponds to the selection actually compiled into that particular executable copy of **hp2xx** (which may vary according to licensing considerations, local availability of third-party libraries, or preferences of whoever built it).

During operation, **hp2xx** outputs various information about the current HP-GL file and about **hp2xx**'s actions. As usual, all this goes to `stderr`. You can re-direct these diagnostics into a file even without any help from a UNIX shell by specifying a log file using option `-l logfile`, or you may switch off diagnostics completely with option `-q` ('quiet' mode). NOTE: Using both options as in `-q -l logfile` is of no use as it will result in an empty *logfile*.

Finally, there is a simple way to rotate whole pictures: Option `-r angle` rotates the picture counter-clockwise by the supplied angle (given in degrees). E.g.,

```
hp2xx -r90 foo.hp
```

will show the picture rotated by 90 degrees, letting vectors originally pointing left-to-right now point bottom-to-top. This may be handy e.g. for printing in landscape format. NOTE: The limiting rectangle supplied by `-hw` is not affected by `-r`, so in order to obtain e.g. a full-page landscape picture on an A4 page, issue a command similar to:

```
hp2xx -m pcl -d 150 -r90 -h270 -w160 landscape.hp
```


3 Advanced subjects

3.1 The coordinate range

The natural unit of length in HP-GL is $1/40$ mm = 0.025 mm, so a typical A4 page covers roughly 11000 x 7500 natural units. Typically, coordinates in HP-GL commands will be found in the range 0 ... 12000. **hp2xx** will tell you the maximum and minimum coordinates (“picture limits”) it finds in your HP-GL picture for both x and y direction. These values usually roughly cover this range. Even if your HP-GL source plots in user-specific coordinates (realized via HP-GL command ‘SC;’ (SCale)), this remains true, since **hp2xx** internally transforms all points back to natural coordinates. Whenever the above range is grossly violated, you may suspect corrupted data, because no real plotter would be able to plot such a file.

If you ever discover a picture limit equalling plus or minus 10^{10} , your HP-GL probably didn’t draw anything. Initially, **hp2xx**’s internal picture limits are set to impossibly large (or small) values, i. e., $\pm 10^{10}$, but the first plot command will set them to values found therein, and successive plots push the limits outward. Example: *xmax* starts at -10^{10} , the first plot command may change it to 2536, the next to 3470, the next 20 command fall short, etc. Eventually, *xmax* assumes the largest value and stays there. Knowledge about these details may sometimes be crucial (see [Section 3.3 \[Scaling to true size\]](#), page 9).

hp2xx uses the picture limits internally for scaling and fitting the data into the supplied limiting rectangle (see [Section 2.2 \[Sizing your output\]](#), page 3). You can also affect the picture limits yourself for special effects (see [Section 3.2 \[Fixed scaling\]](#), page 9).

3.2 Fixed scaling

As noted earlier, **hp2xx** does not draw to scale, but rather it fits a picture into a given limiting window. While this is very handy in most applications, it may be undesirable when a series of pictures must be drawn to the same scale. Unless all pictures possess the same picture limits (modulo offsets), e.g., because all of them are surrounded by some fixed frame, **hp2xx** would scale them all up differently to fit each of them tightly into the limiting window.

There are two simple cures: First, make use of the true size option ‘-t’. If the original HP-GL sizes do not fit, adjust picture limits to guarantee a constant scaling: Make a preview of all pictures and note the coordinate ranges **hp2xx** reports. Then, determine picture limits which cover all of these individual limits. Finally, run **hp2xx** to create your desired outputs using options ‘-xXyY’ to tell **hp2xx** about the picture limits it should use. If the pictures do not share common offsets, you may have to correct for offsets manually. Use the preview mode for testing. You’ll get the same scale as long as the limiting window and (*xmax* - *xmin*) and (*ymax* - *ymin*) remain constant for all pictures.

WARNING: **hp2xx** does not clip lines (unless told to by the IW directive of HPGL/2, of course). If the picture limits which you manually can pre-set via options ‘-xXyY’ are chosen too narrow, they will be pushed outside just as described in the last section, resulting in a different scale. Check the coordinate ranges **hp2xx** reports. The should match the values supplied by options ‘-xXyY’!

3.3 Scaling to true size

The very first (i.e. ancient) releases of **hp2xx** (binaries) did not offer option ‘-t’, which does everything you’ll need for producing output with exactly the sizes shown on a real plotter. The following paragraph shows how to manually emulate the working of this option. Though outdated, I left it in the manual as background material:

Sometimes you might want to create pictures sized exactly as if they were drawn on a real plotter. There is a little trick which allows you to do so using **hp2xx**: As notes above, the natural

unit of length in HP-GL is 0.025 mm. Therefore, you can calculate the true picture size from the picture limits reported by **hp2xx**. Transform these data into mm and simply specify the limiting window accordingly! Example: '**hp2xx truesize.hp**' reports the following coordinate ranges: $xmin = 250$, $xmax = 5250$, $ymin = 100$, $ymax = 3100$. Thus, the picture is $(xmax - xmin) * 0.025$ mm = 125 mm wide and $(ymax - ymin) * 0.025$ mm = 75 mm high, and '**hp2xx -w125 -h75 truesize.hp**' will draw it in true size.

3.4 Swapping

hp2xx allocates memory for an internal bitmap dynamically. Large pictures, high resolution, and use of colors may combine to let your computer run out of memory (especially on non-swapping operating systems like DOS).

In this case, **hp2xx** swaps the bitmap to disk, slowing down considerably. Redirecting swapping to a fast disk, preferably a RAM disk, might speed up things. You can replace the default swap file '**hp2xx.swp**' using '**-s 'swapfile'**'. NOTE: If for some reason **hp2xx** is aborted during swapping, you might have to delete the swap file manually.

3.5 Dots and lines

Here are some basics about the generation of dots and lines within **hp2xx**. I mention them, because there is something left to be improved here...

Some HP-GL codes cause **hp2xx** to generate points rather than lines of length zero. There is a subtle difference between both. Depending on the current output format, special code for points will be generated, and occasionally, a point will look different from a zero-length line. Use '**-m epic**' for such an example.

Line thicknesses can vary. Especially for thick lines, the matter of line caps (how lines are ended, e.g. with a round cap) becomes relevant. (According to the HPGL/2 references, these issues are only relevant for lines wider than 3.5 mm). **hp2xx** does not yet offer complete support for these features - currently your best bet is to use '**-m eps**' or '**-m pdf**', as **hp2xx** will simply emit the corresponding linestyle commands of those formats instead of having to calculate all the details itself. While most line end cap types are supported even in the raster modes (since version 3.4.3), this applies especially to the mitering options, i.e. the way the junction is drawn. To force a certain type of line caps, you can also use '**-m eps**', edit the resulting Encapsulated PostScript file, look for a line with **setlinecap** in it (near line 45), and select the line cap of your choice by modifying the PostScript command **setlinecap** accordingly. You can also use Metafont (via '**-m mf**') and replace the picked pen "pencircle" by some other type. However, both methods are far from convenient.

The internal rasterization done by versions of **hp2xx** before 3.4.3 was a simple process based on positioning squares (roughly corresponding to the linewidth) by the Bresenham algorithm. Since 3.4.3, a modified algorithm for drawing wide lines - based on an IBM Technical Bulletin written by Alan Murphy - is employed, which also supports true endcap drawing. The current code still does not support the various miter options, e.g. drawing out the endpoints of two lines so that they meet in an exact triangle or a blunt wedge. Unfortunately the current layout makes it hard to implement such a feature, as the internal representation of the command sequence is just another sequential list of lines without any indication of intersections.

3.6 Unsupported formats

This is just a brief note, not a real manual entry – sorry.

PIC	ATARI format, e.g. for the text processor Signum. Try to replace by IMG.
PAC	ATARI format, e.g. for the CAD program STAD

DJ_GR	DOS previewer, based on DJ Delorie's gcc port and extender go32. Works fine, but will be replaced by DOS/OS2 EMX version. Abandoned by new maintainer due to lack of platform - volunteers welcome.
OS2	Full-screen OS/2 2.x and DOS previewer. HWW: I don't yet have the right development system, so this code is still missing. MK: Any volunteers ? Is this still needed at all, now that several years have passed since HWW wrote this comment ? OS/2 users should be able to use the 'PM' mode, which is kept alive by Kazutaka Nishiyama.

3.7 T_EX formats

This section is a bit historic now, as one can easily embed 'eps' graphics in LaT_EX and preview the result with xdvi et al. nowadays. I have left it in for the sake of completeness, and for special cases where one might still prefer plain T_EX or one of the classical import methods.

T_EX was designed for typesetting, not for handling graphics. Putting graphics directly into T_EX therefore is always somewhat clumsy. hp2xx offers four different compromises to do that, and much better, though more indirect ways.

'-m mf' generates Metafont source code. Run **Metafont** and **gftopk**, and you'll end up with a special **pk** font containing the single letter Z which represents your picture. Placing this Z somewhere in your document using standard T_EX commands draws your picture there.

If you want to avoid fiddling with additional programs and fonts, if you work with LaT_EX, and if you do not need high-quality plots, the macros within **epic.sty** may help you. '-m tex' causes hp2xx to generate appropriate T_EX source code which you can '\input{' into LaT_EX sources.

For emT_EX users, there are yet another two way: '-m em' creates T_EX code containing many commands like '\special{em:...}' for line drawing. The line drawing task will therefore be handled not by T_EX itself but by the emT_EX drivers which can handle arbitrary line slopes etc. Similarly, '-m cad' produces code based on the same principle, but compatible with program 'T_EXcad.exe', which is distributed as a part of emT_EX, and which offers editing and drawing features for the desired HP-GL figure(s).

Please note that all methods for generation of graphics *within* T_EX are compromises which usually work only for simple graphics. You'll probably prefer using external methods like including EPS vector graphics files with Tom Rokicki's **dvips** driver, or PCX files via the emT_EX drivers, or you'll generate special fonts with convenient programs like F. Sowa's **bm2font**. hp2xx can help you in all of these cases. The following table shows the pros and cons of the various approaches (all are based on PD software):

Internal methods (all allowing DVI previewing of graphs):

via Metafont

- + : Machine-independent; fully compatible with T_EX
- : Slow; capacity problems with Metafont / gftopk / some DVI drivers
if used with large and/or complex graphics

via **epic.sty**

- + : Machine-independent; single-step, native LaT_EX approach; PD software
- : Slow; requires LaT_EX; low-quality lines; just one line thickness;
complex graphs may exceed T_EX capacity

via emT_EX's \special{em:...}

- + : No T_EX capacity problem; good line quality; single-step procedure;

- rasterization on demand, giving optimal resolution
- : Slows down drivers; driver capacity may be exceeded; emTeX required

External methods:

via PCX file inclusion:

- +: Easy and fast; DVI preview of graphics
- : Requires emTeX drivers (only available on DOS and OS/2)

via special fonts:

- +: Easy, fast, and trouble-free font generation via `bm2font`;
DVI preview of graphics (!); portable
- : Many files for fonts etc.; confusing for novices

via EPS:

- +: High-quality results; easy; no burden for TeX or drivers
- : No DVI preview (unless ghostscript is available); PostScript printer (or, e.g., GhostScript) required;
PostScript previewing is slower than pure DVI previewing.

4 Installation and modification notes

4.1 Installation procedure

Please note: The following description is very brief and assumes that you are familiar with installation of PD software in general.

4.1.1 Installing an executable version

This is simple! If you find a collection of pre-compiled versions of **hp2xx**, obtain the file **‘read.me’** and read it to find out the name of the file which fits to your system. Obtain it, rename it to something like **‘hp2xx’** or **‘hp2xx.exe’**, and place it somewhere on your search path – that’s it. Linux users will usually be able to find appropriate RPM, DEB or similar binary packages for their distribution on the CDs and websites of the various distributors. FreeBSD users should check the **ports** collection for packages, and users of commercial unices should check the freeware archives provided by their vendors. Finally, users of that other operating system may find semi-current versions through the gnuwin32 project page on sourceforge.net.

However, since the source release of **hp2xx** under the GNU public license, the usual mode of installation has become configuring and building **hp2xx** on your own system. This is detailed in the next section.

4.1.2 Source-level installation

NOTE: I am tempted to switch to GNU autoconf , i.e. have a configure script generate the necessary Makefile automatically. This will probably not be welcomed by those not on Unix-like platforms, however. Currently, installation depends too much on manual work yet. Here is a description how to proceed:

After unbundling all sources, go to subdirectory **‘./makes’**. Select a makefile most closely resembling your system’s needs from the samples given, copy it to **‘./sources/makefile’**, adapt it manually (if necessary), and run **make all**. (The source package as distributed through ftp.gnu.org usually contains a file **‘sources/Makefile’**. This is what i use myself on Linux systems, so you could try using that one first, if you are trying to build **hp2xx** on this or a similar platform. If everything is set correctly, this results directly in a valid executable file which you may install at any convenient place on your search path.

There are two types of makefile adaptation: First, let’s assume there is a makefile template available for your system. You then have the option to add a few non-default modes. (These are typically platform-specific modes, or modes requiring third-party libraries like PNG, JPEG, TIFF or PDF that are not usually part of a default system installation - particularly on commercial operating systems). Do so by un-commenting the appropriate lines near the beginning of the makefile, and by commenting out the corresponding standard lines.

The second type of course applies to systems with special needs which are not yet covered by any makefile template. Currently, you are on your own when it comes to supplying alternate paths, renaming or adding system libraries and alike. Most probably you might have to tell the makefile where to look for the X11 stuff, and where to find the png library, if at all.

Note: Don’t feel alarmed if your makefile seems to neglect many source files. Any single installation will make use of only one previewer (two on SUNs with activated SunView support), and there are platform-dependent sources for some output formats which are not always used.

Users wishing to compile **hp2xx** on MS Windows systems will find a step-by-step procedure in the file **‘hp2xx_nt’** in the documentation directory.

4.2 Adding your own formats

First, study [Chapter 1 \[Introduction\]](#), [page 1](#) for the outline of the modular structure and general operation of `hp2xx`.

Let's assume you want to support TIFF format. The probably easiest way of adding new formats is by modifying copies of existing files. Since TIFF is a raster format, a good starting point would be `to_pcx.c`. (Files `to_mf.c` or `to_eps.c` should be considered in case of a vector format, and `to_vga.c` or `to_x11.c` in case of a new previewer.) Copy it to a file `to_tiff.c` and edit the latter. The old code is pretty much self-explanatory. Essentially, the output file is opened, initializations are performed, and the internal bitmap is converted into the target format (here, TIFF) scanline-by-scanline. There is just one routine called from other modules (originally named `PicBuf_to_PCX`. Rename it to e.g. `PicBuf_to_TIFF`) and adapt the conversion code.

Once you've done that, the rest (integration of the new format into the package) is easy: First, edit `hp2xx.h` and add a prototype line for `PicBuf_to_TIFF` in analogy to e.g., `PicBuf_to_PCX`. Edit the `makefile`'s and add `to_tiff.c` to the list of sources and e.g. `to_tiff.o` to the list of objects. Now you are ready for compilation tests (but not for linking yet).

Then, change the main file `hp2xx.c` at various places: Near the beginning of the file, add `'XX_TIFF, '` to the `hp2xx_mode` typedef, and a line like `'XX_TIFF, "tiff", '` to the `ModeList` struct below. Please note the alphabetical order of these lists. Never put anything behind the termination code `XX_TERM`! At the end of the file, add a `case` statement to the `switch` list in analogy to e.g. the `PCX` entry.

You may also want to add a line to the on-line help to announce the new format, and change the release number and date. Look for functions `Send_ID` and `usage_msg` at the first quarter of file `hp2xx.c`!

Now a `make all` will produce code containing the new format. If your format turns out to work nicely and seems to be of general interest, please consider contributing it to the `hp2xx` project.

4.3 Future improvements

The following table lists miscellaneous desirable features for future releases (check the `'TODO'` file in the source package for current status) :

- Other, more rarely used HP-GL commands
- PCL: Better data compression for DJ500, DJ500C, DJ550;
- Loadable fonts, e.g. Hershey fonts, or: more built-in fonts

4.4 Font coding

This section is intended for those few users who might care to improve the built-in character set of `hp2xx`.

HP-GL plotters feature built-in fonts with both fixed and variable-width characters. There are commands for font selection and quick switching between two pre-selected fonts, and there is also a way for users to download own character definitions.

`hp2xx` currently features just a few fixed-width character sets.

If you plan to modify these characters set or to add more, you need an understanding of how characters are drawn by `hp2xx`. The source file `'charset.h'` contains a comment explaining this procedure. Below you find a (modified) copy of this:

This file defines a standard character set by elementary
"draw" & "move" commands. The format is a very compact one from
the old days where every byte was still appreciated.

A font or character set is an array of strings. Each character is
addressed by its ASCII code.

A character is a (NULL-terminated) string of bytes. Each byte
codes for a draw or move action according to the code below:

```

Bit:  7  6  5  4  3  2  1  0
      p  x  x  x  y  y  y  y

```

p: Plot flag. If set, "draw to" new point, else "move to" it.
xxx: 3-bit unsigned integer (0...7). X coordinate of new point.
yyyy: 4-bit unsigned integer (0..15). Y coordinate of new point.

The baseline is $y = 4$ instead of $y = 0$, so characters with parts below it can be drawn properly. Function "code_to_ucoord" transforms these coordinates into actual user coordinates.

Example: code for character 'L': `"\032\224\324"` translates to:
`moveto(1,10); drawto(1,4); drawto(5,4);`

From the example you can conclude that the font below essentially is defined on a 5x7 grid:

	0	1	2	3	4	5	6	7	
15	-	-	-	-	-	-	-	-	- : unused
14	-	-	-	-	-	-	-	-	# : always used
13	-	-	-	-	-	-	-	-	o : sometimes used
12	-	-	-	-	-	-	-	-	
11	-	-	-	-	-	-	-	-	
10	o	#	#	#	#	#	-	-	
9	o	#	#	#	#	#	-	-	
8	o	#	#	#	#	#	-	-	
7	o	#	#	#	#	#	-	-	
6	o	#	#	#	#	#	-	-	
5	o	#	#	#	#	#	-	-	
4	o	#	#	#	#	#	-	-	
3	o	o	o	o	o	o	-	-	
2	o	o	o	o	o	o	-	-	
1	o	o	o	o	o	o	-	-	
0	o	o	o	o	o	o	-	-	

Appendix A Known HP-GL commands

hp2xx originally emulated a subset of the Hewlett-Packard 7550A plotter. The following manual was used as reference for command definitions: [1] *HP 7550A Interfacing and Programming Manual*. This has gradually been expanded to include almost all of what is now known as HPGL/1, and most of the commands from HPGL/2.

Not all commands are supported. Among the non-supported commands are those which do not really apply to a software emulator, like:

- commands affecting the communication between plotter and host computer,
- commands for changing the behaviour of a real plotter, like plotting speed etc.,
- commands for the control of plotter memory allocation,
- commands causing various plotter display outputs.

Other non-supported commands would be desirable, but were left out due to their inherent complexity (or just because nobody had an immediate need for them yet).

Programmers intending to add more HP-GL features should ideally take care to implement the less-than-obvious side effects of existing commands on the new features, too (and vice versa). E. g., line types (LT;) affect most but not all drawing commands: While the ER; command (edge rectangle relative) uses the current line type, its counterpart EA; (edge rectangle absolute) always draws solid lines. However, both PR; and PA; use the current line type! In addition, new features may need initializations by the already supported codes IN; or DF;, so these may have to be expanded.

The remainder of this section lists all HP-GL commands given on pages 1-2 to 1-4 of [1] and marks them as either

- (.) not applicable,
- (-) ignored,
- (*) partly supported, or
- (+) supported.

The label “supported” is used when I think the command is fully supported in the context of the already implemented commands. In general, you should have absolutely no problem with this class of commands.

Though there still are unsupported commands, this does not mean that you might have trouble using hp2xx. Nowadays, most HP-GL files are machine-generated, e.g. by CAD or DTP programs. These tend to make use of just a subset of HP-GL. To my experience, chances are high that hp2xx will give you the picture you want! (Areas where hp2xx is definitely lacking are font support and polygon fills, but even there you should get a good approximation of what a plotter would produce).

HP-GL s n Description & Remarks		
Cmd	i	
===== ==== =====		
AA	+	Arc Absolute
AC	+	Anchor corner
AD	-	Alternate font Definition
AF	+	Advance Full page [same as PG]
AH	+	Advance Half page [same as PG]
AP	.	Automatic pen operations
AR	+	Arc Relative
AS	.	Acceleration select
AT	+	Arc through three points
----- ---- -----		
BF	-	Buffer Plot [maybe not a valid hpgl command]

BL	+		Buffer Label
BP	+		Begin Plot
BR	+		Bezier curve, Relative
BZ	+		Bezier curve, Absolute
----- --- -----			
CA	+		Designate alternate character set
CC	-		Character Chord angle
CF	-		Character Fill mode
CI	+		Circle
CM		.	Character selection mode
CO	+		File comment
CP	+		Character plot
CR	+		Color Range
CS	+		Designate standard character set
CT	+		Chord tolerance
CV		.	Curved line generator
----- --- -----			
DC		.	Digitize clear
DF	+		Default
DI	+		Absolute direction
DL	-		Define downloadable character
DP		.	Digitize point
DR	+		Relative direction
DS	-		Designate character into slot
DT	+		Define label terminator
DV	+		text Direction Vertical
----- --- -----			
EA	+		Edge rectangle absolute
EC		.	Enable paper Cutter
EP	+		Edge polygon
ER	+		Edge rectangle relative
ES	+		Extra space
EW	+		Edge wedge
----- --- -----			
FI	-		pcl Font ID
FN	-		pcl secondary Font Number
FP	+		Fill polygon
FR	-		FFrame advance
FS		.	Force select
FT	+		Fill type
----- --- -----			
GC		.	Group count
GM		.	Graphics memory
----- --- -----			
IM	-		Input error reporting mask
IN	+		Initialize
IP	+		Input P1 and P2
IR	+		Input Relative P1 and P2
IV	-		Invoke character slot
IW	+		Input window
----- --- -----			
KY		.	Define key

----- --- -----		
LA	*	Line Attributes
LB	+	Label
LO	+	Label origin
LT	+	Line type
----- --- -----		
MC	-	Merge Control
MG	+	Message [same as WD]
MT	.	Media Type
----- --- -----		
NP	+	Number of Pens
NR	.	Not ready (unload page and go offline)
----- --- -----		
OA	.	Output actual position and pen status
OC	.	Output commanded position and pen status
OD	.	Output digitized point and pen status
OE	-	Output error
OF	-	Output factors
OG	.	Output group count
OH	-	Output hard-clip limits
OI	.	Output identification
OK	.	Output key
OL	-	Output label length
OO	.	Output options
OP	+	Output P1 and P2
OS	-	Output status
OT	.	Output carousel type
OW	+	Output window
----- --- -----		
PA	+	Plot absolute
PB	+	Print buffered label
PC	+	Pen Color
PD	+	Pen down
PE	+	Polyline Encoded
PG	+	Page feed
PM	+	Polygon mode
PR	+	Plot relative
PS	+	Plot Size
PT	+	Pen thickness
PU	+	Pen up
PW	+	Pen Width
----- --- -----		
QL	.	Quality Level
----- --- -----		
RA	+	Fill rectangle absolute
RF	-	Raster Fill pattern
RO	+	Rotate coordinate system
RP	-	Replot
RR	+	Fill rectangle relative
RT	+	Relative arc through Three points
----- --- -----		
SA	+	Select alternate character set

SB	-	Scalable or Bitmap font selection
SC	+	Scale
SD	-	Standard font attribute Definition
SI	+	Absolute character size
SL	+	Character slant
SM	+	Symbol mode
SP	+	Select pen
SR	+	Relative character size
SS	+	Select standard character set
ST	.	Sort vectors
SV	-	Screened Vectors
----- ----- -----		
TD	-	Transparent Data
TL	+	Tick length
TR	-	Transparency mode
----- ----- -----		
UC	+	User-defined character
UF	-	User-defined fill type
UL	+	User-defined line type
----- ----- -----		
VS	.	Velocity select
----- ----- -----		
WD	+	Write to display
WG	+	Fill wedge
WU	+	pen Width Unit
----- ----- -----		
XT	+	X-Tick
----- ----- -----		
YT	+	Y-Tick

Appendix B Option summary

In the following, options are grouped into subjects and are listed alphabetically within each subject. Both long options and short (one-letter) options are listed, where short options appear in parentheses. Except for the +DPI option, there is a one-to-one correspondence between long and short options. You may use either long or short options. Mixing long and short options is acceptable.

Option parameter names suggest the expected data type, e. g., ‘`--rotate (-r) float`’ means that option ‘`--rotate`’ or its corresponding short form ‘`-r`’ expect a parameter of type ‘float’.

B.1 General options

‘`--pencolors (-c) string`’

Pen color(s), a string of 1 to 8 digits. Valid digits: 0...7 (0=Background or off, 1=Foreground, 2=Red, 3=Green, 4=Blue, 5=Cyan, 6=Magenta, 7=Yellow). Default: ‘11111111’

‘`--outfile (-f) string`’

Name of output file. If omitted, `hp2xx` generates the name from the input file name and the current mode string. ‘`-f-`’ causes `hp2xx` to write to `stdout`. Default: none.

‘`--logfile (-l) string`’

Name of log file. If given, diagnostics go into this file, else to `stderr`. Remark: ‘`-q`’ inhibits all diagnostics!

‘`--mode (-m) string`’

Mode string. Valid: *string* =

- ‘`cad`’ (T_EXcad-compatible line generation using `\special{em:...}`),
- ‘`dxl`’ (Autocad Drawing Exchange Format),
- ‘`em`’ (more efficient line drawing with E. Mattes’s T_EX `\special{em:...}`),
- ‘`eps`’ (Encapsulated PostScript),
- ‘`gpt`’ (Gnuplot input file),
- ‘`hpgl`’ (simplified HP-GL, useful for imports),
- ‘`img`’ (e.g., GEM’s Image format),
- ‘`jpg`’ (JPEG image format),
- ‘`mf`’ (Metafont source),
- ‘`nc`’ (G code for CNC engraving tools)
- ‘`pbm`’ (Portable Bitmap (monochrome) or Portable Pixmap (color plots)),
- ‘`pcl`’ (HP-PCL Level 3),
- ‘`pcx`’ (Paintbrush format),
- ‘`pdf`’ (Adobe Portable Document Format),
- ‘`png`’ (Portable Network Graphics format),
- ‘`pre`’ (Preview mode; no output!),
- ‘`rgip`’ (Uniplex RGIP vector format),
- ‘`svg`’ (Scalable Vector Graphics, for XML web pages)
- ‘`tex`’ (line drawing with T_EX / `epic` macros)
- ‘`fig`’ (for Brian Smith’s XFIG vector drawing program)
- ‘`tiff`’ (TIFF tagged image file format)
- ‘`esc2`’ (Epson Stylus printers)

Occasionally available (currently unsupported) modes:

- ‘`ilbm`’ (e.g., for AMIGA: ILBM/IFF format),
- ‘`pac`’ (e.g., for ATARI/STAD),

‘pic’ (e.g., for ATARI/Signum).

Default: ‘pre’.

‘--nofill (-n)’

Ignore all polygon fill commands in the HPGL file(s). This may sometimes be necessary to improve readability of images at small scales (e.g. thumbnails in an overview listing), or to work around problems in the polygon fill code.

‘--pensize (-p) *string*’

Pensize(s), a string of 1 to 8 digits. Valid digits: 0...9 (unit = 1/10 mm) for vector modes, 0...4 (unit = pixel) for raster modes. Default: ‘11111111’

‘--pages (-P) *int*[:*int*]

Select HP-GL page *int* or a page range. Valid: *int* integer and ≥ 0 . Default: *int* = 0 (all pages).

‘--quiet (-q)’

Quiet mode (no diagnostic output).

‘--rotation (-r) *float*’

Rotation angle [deg]. E.g., ‘-r90’ gives landscape. Default: 0.0

‘--swapfile (-s) *string*’

Name of swap file. Default: *string* = ‘hp2xx.swp’.

‘--mapzero (-M) *int*’

Remap commands affecting pen number zero to any other (preferably unused) pen (mainly to avoid changing the background color of raster format files).

B.2 Size controls

‘--aspectfactor (-a) *float*’

Aspect factor. Valid: *float* > 0.0 . Use *float* > 1.0 for landscape and *float* < 1.0 for portrait deformations. Default: *float* = 1.0

‘--height (-h) *float*’

(Upper limit of) height of picture in mm. Default: *float*=200.0

‘--width (-w) *float*’

(Upper limit of) width of picture in mm. Default: *float*=200.0

‘--x0 (-x) *float*’

Pre-set left limit of HP-GL coordinate range to *float* (rarely used).

‘--x1 (-X) *float*’

Pre-set right limit of HP-GL coordinate range to *float* (rarely used).

‘--y0 (-y) *float*’

Pre-set lower limit of HP-GL coordinate range to *float* (rarely used).

‘--y1 (-Y) *float*’

Pre-set upper limit of HP-GL coordinate range to *float* (rarely used).

‘--zengange (-z *float*’

This option is only relevant for CNC G-code generation (‘-m nc’, where it sets the working depth of the tool.

‘--zretract (-Z *float*’

This option is only relevant for CNC G-code generation (‘-m nc’, where it sets the retraction level of the tool.

`--truesize (-t)`

Ignore options `'-a -h -w'` (aspect factor, height, width). Size information will come from the HP-GL intrinsic data. WARNING: Avoid using option `'-r'` (rotate) as it works on top of HP-GL and thus will distort the detected HP-GL sizes.

B.3 Raster format controls

`--DPI (-d) int`

Set x resolution to *int* dots per inch (DPI). If not overridden by `'-D'`, sets also y resolution to *int* DPI. Valid: *int* integer and > 0. Default: *int* = 75.

`--DPI_x (-d) int`

Same as `'--DPI'`

`--DPI_y (-D) int`

Set y resolution to *int* DPI. *int* integer and > 0. Default: *int* = 75.

`--extraclip (-e) int`

Set extra clipping space to *int* plotter units. Default: *int* = 0.

B.4 PCL specifics

`--PCL_formfeed (-F)`

Send a FormFeed after graphics data. Default: No FormFeed.

`--PCL_init (-i)`

Pre-initialize printer. Default: No pre-init

`--PCL_Deskjet (-S) int`

Use (Deskjet) Special commands. *int* = 0 deactivates this option, *int* = 1 enables b/w mode, *int* = 3 is intended for DJ500C (CMY) color support, *int* = 4 supports DJ550C (CMYK mode).

`--DPI_x (-d) int`

Set x resolution (see above): Valid here: *int* = 75, 100, 150, 300

`--DPI_y (-D) int`

Set y resolution (see above). Invalid here!

B.5 TIFF specifics

`-S int` Select TIFF compression algorithm. *int* = 0 or 1 no compression, *int* = 2 run length encoding (RLE), *int* = 3 Group 3 FAX, *int* = 4 Group 4 FAX; *int* = 5 LZW (normally disabled in `to_tiff.c` due to Unisys patent), *int* = 6 old-style JPEG, *int* = 7 JPEG, *int* = 8 Deflate.

B.6 TIFF specifics

`-S int` Select desired translation of HPGL pen attributes to DXF Group 62 line parameters. *int* = 0 ignore attributes (use color=1, width=0.1 throughout), *int* = 2 use pen number for color, use actual width, *int* = 3 use width-dependent color (<0.2 == 1, <0.3 == 2 etc.), *int* = 4 use width-dependent color with upper limit of 4 beyond 0.4.

B.7 Margins

(Apply to modes ‘eps’, ‘pcl’, ‘pre’ ONLY)

‘--center (-C)’

Center the picture within the frame defined by options ‘-w -h’ by adding to the left or upper margin. Null effect if the width/height ratio of the picture matches that of options ‘-w -h’.

‘--no_ps (-N)’

Ignore papersize definition given by a PS command in the HPGL file, and calculate the actual image size. This may be necessary to avoid unnecessary margins (with programs that write a standard header regardless of the true size of the drawing).

‘--xoffset (-o) *float*’

X offset of picture (left margin) in mm. Valid: *float* >= 0.0, default: *float*=0.0

‘--yoffset (-O) *float*’

Y offset of picture (upper margin) in mm. Valid: *float* >= 0.0, default: *float*=0.0

B.8 Preview (DOS/PC’s only)

‘--VGAmodebyte (-V) *int*’

VGA mode byte (decimal). Default: *int* = 18. WARNING: Setting inappropriate VGA modes may damage your hardware, especially your monitor!

B.9 Help

‘--help (-H)’

(or calling hp2xx without any arguments) Show on-line help.

Appendix C Acknowledgements

C.1 Acknowledgement

Since `hp2xx` first became publicly available (in its early days, as binaries on several platforms), many people contributed to this project by supplying first HWW and now MK with valuable suggestions, code patches and reports. Many thanks to all of them! It is a pleasure to especially thank the following people for their outstanding contributions:

HWW's Versions up to 3.2.0 (mid-1994):

Nelson Beebe

Help with the new generic makefile (easier configuration); clean selection mechanism for previewer suggested; suggestions for an improved X.11 previewer

Elisabeth Dregger-Cappel

Network and host resources for original `hp2xx` distribution

Joern Eggers

New ATARI format "cs" for CS-TeX; bug fixes for arcs / circles

Roland Emmerich

DOS betatests and suggestions; `showit`

R. Frahm PCX color improvements

Jonathan M. Gillian

DOS betatests and suggestions

Gilles Gravier

RGIP converter

Thomas Hiller

Code for EA support

Claus H. Langhans

AMIGA portation; pbm, ilbm formats

Lawrence Lowe

Many helpful suggestions and improvements, e.g. for pcl format

Ian MacPhedran

Color / binary format for pbm/ppm

Norbert Meyer

ATARI portation; img, pic, pac formats; first ATARI previewer

Michael Schmitz

Many VAX & MACH tests

Michael Schoene

X11 stuff; many tests

Andreas Schwab

Improved ATARI previewer

Friedhelm Sowa

Many DOS tests and suggestions for cooperation of `hp2xx` with \TeX figure generation

Gerhard Steger

Many VAX tests; VAX scripts; access to MicroVAX platforms

Horst Szillat

OS/2 support, OS/2 full-screen previewer & help

Alois Treindl

Code for UC support

Versions since 3.3.0 (mid-1999):

Emmanuel Bigler

xfig and gnuplot ascii support

Ian MacPhedran

XFig output

Michael Schmitz

PNG output

Rolf Schreck

PS/RO fixes

Eugene Doudine

PE command support

Georgy Salnikov

Improvements for character commands (LB,DI,DR)

Michele Liberi

TIFF output

Lars Erikson

EPS plotsize bug fixes

Gerhard Buergmann

bug fixes for plot scaling and PE support

James Shaw

polygon mode testing and examples, WinNT building instructions

Andrew Bird

enhanced PW support raster-mode linewidth rendering TIFF enhancements

Bengt-Arne Fjellner

EMF generation

Michael Rooke

CNC G-code generation

Georg Viehsoever

DXF enhancements and bugfixes

C.2 Copyright notice

Copyright (c) 1998 - 2003 Martin Kroecker

Copyright (c) 1991 - 1994 Heinz W. Werntges

All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the abovementioned author(s).

THIS SOFTWARE IS PROVIDED “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

