Usage notes for DVIWIN 2.9

Hippocrates Sendoukas

December 1, 1994

Introduction

This document summarizes the use of DVIWIN which is a driver for DVI files under MS-Windows 3.1 and Windows NT. Some portions of the code have been adapted from the DVI driver family by Nelson Beebe. This version is optimized for smaller processors and exploits some unique characteristics of the operating environment. Most program functions can be accessed through standard menus or accelerator keys that are shown after each menu entry. The general idea is to build a bitmap with the current page contents and display it on a window. You can move around the document by using the scrollbars, menu commands or shortcut keys. You can also print the file on any printer supported by Windows, provided that the printer driver supports bitmaps (and the printer has memory for a full page of graphics), and your machine has enough memory to construct the bitmap. It also supports TeX \special{} commands for inserting arbitrary graphics in a document. Finally, it has some special facilities to facilitate the integration between an editor, TeX and DVIWIN. The program opens as many font files as possible: please make sure that you set the maximum number of files in your system to at least 50 through the "FILES=" statement in your config.sys file.

Main Menu

File This submenu lets you load a new file, close the current file, print the current file, configure your printer or exit the program. At the end of the submenu, you will also see the last four files that you viewed; you can open any of them simply by clicking on the desired filename; the program also remembers the position within each file, and will place you there automatically. The "Print" entry leads to a dialog box that lets you select the pages to be printed, specify an external dvi driver to be used for printing, and setup the printer (you can also select the active printer if you have installed multiple drivers). You can also select the resolution at which the document is sent to the printer; it defaults to the "Auto" value which is as close as possible to the printer's resolution, but you can easily override it. When exiting the program through the "Exit (Save Parms.)" command, all parameters are saved for future reference; You can use the "Quit" command if you want to exit the program without saving the current parameters.

Page This submenu lets you move around the current page, or to other pages. You can also use the standard cursor keys to duplicate most of these functions. The "Goto" entry lets you specify the page that you want to display; a negative number specifies a page relative to the end of the document. That is, a minus one displays the last page of the document; a minus two shows the page before that, and so on. You can go to the next or previous page of the document by pressing the "PgDn" or "PgUp" keys, by clicking the buttons at the far right side of the main menu, or by clicking the right mouse button on the vertical scrollbar; a click on the upper half of the scrollbar is equivalent to a Page Up command, while a click on the lower half is equivalent to a Page Down command. The next four commands move you to each corner of the page; if you press them once, they move you to the respective margin; a second press moves you to the physical corner of the paper.

Resolution This submenu lets you select the resolution applied to the document. You can

use practically any resolution to accomodate any raster device with a Windows driver. The last two entries in the submenu are initially empty: you can put there any desirable resolution for your screen or printer; this is done through the "Custom Resolutions" entry in the "Options" submenu. There is a keyboard shortcut here not appearing on any menu: the plus key (in the numeric keypad) in conjunction with the control key moves you one step up in the resolution table; the control-minus combination moves you in the opposite direction. This shortcut is available only for the standard resolutions; since the custom resolutions appear at the end of the table, they are not numerically ordered, so the shortcut is disabled to prevent any mishaps. Keep in mind that higher resolutions require more memory. For machines with 4M of RAM, the program works comfortably (ie., without significant swapping) with resolutions up to 400dpi on Letter or A4 paper sizes; you will need more memory for higher resolutions or larger paper sizes.

Zoom The program can also scale the entire page, so you can use the printer fonts even for the screen. This submenu lets you control the zoom factor directly; you can also increase or decrease the zoom factor by pressing the plus or minus keys (in the numeric keypad), or by clicking the first two buttons on the right hand side of the main menu. The zooming capability is convenient and saves disk space, but it is not a panacea; I would not recommend for example to use 600dpi printer fonts for the screen, since you will be wasting lots of memory and the display will be much more sluggish than otherwise (there are simply much more pixels on a 600dpi page). Still, the capability is there, so you can decide for yourself if you want to use it.

Options This submenu lets you set several options about the operation of the program. All options (as well as window size and position) are saved in a private configuration file (dviwin.ini for the 16-bit version of the program, or dviwin2.ini for the 32-bit version), so you do not need to set them repeatedly.

Trace Fonts: This option controls the tracing of fonts; the program creates a log file where it stores various messages. If this option is selected, the program will print in the log file all font activities. If you encounter any font problems (such as an empty screen), turn on this option *before* opening the dvi file, and look at the log file *after* the loading; the messages in the log file should help you locate the problem. The program keeps up to seventeen fonts open; if more fonts are needed, it closes the least frequently used font. If the program cannot find a font even though that font exists and the path is correct, verify that you have a "FILES=50" (or higher) statement in your config.sys file.

Rulers: If this option is turned on, the cursor turns into a crosshair, two rulers appear, and the current cursor position is reflected on both rulers. This is very useful if you need to align text precisely. If you need more precise information about text positioning, you can press the left mouse button in conjunction with the control key, and the program will report the exact position of the cursor.

Keep Horizontal Position: The next option determines if the program should preserve your horizontal position when you change pages. This is useful if the displayed document is wider than the screen, and you want to align the window in order to see as much text as possible.

Keep Vertical Position: This is similar to the previous option, but it applies to the vertical position.

Warning Beeps: If this option is turned on, the program will beep whenever you try to do something illegal (eg., trying to go past the last page of the document); otherwise, it will be quiet.

Save parms. by default: This option controls the meaning of the default exit (double click on the top-left window corner); if it is turned on, the program will save the parameters; otherwise, it is equivalent to the "Quit" command.

Black and White: When the zoom factor is greater than one, the program needs to interpolate between the foreground and background colors. It can interpolate between any arbitrary colors, but you will need a 256-color (or better) video mode for optimal results. The color requests under a 16-color mode are ignored, because all palette entries are reserved by Windows; if however we use black letters on a white background, we can still use a four-level grayscale since the standard palette contains two shades of gray. This option enables you to do this, even if you have selected different colors from the Control Panel.

Reset all fonts: This command instructs dviwin to flush the font cache and read all PK and FLI files again. It is useful if you generate fonts automatically and want to avoid the PK format completely.

Default Margins: This dialog lets you set the default initial position when changing pages; TeX documents have both margins set to one inch; therefore, if you set the margins to one inch, you will maximize the initial viewing area. You can always move around the page (using the scrollbars or the cursor keys), but it is better to begin from a convenient location.

Apply Margins: You may find it inconvenient to specify the numerical values of the margins. If you select this command, the program will set the default margins to the current ones. In this way, all you need to do is position the document properly and select this command.

Font Directory: This option specifies the structure of the font directory. We normally create a base directory (eg., c:\fonts) with a subdirectory for each resolution. Some programs require the name of each subdirectory to be just the resolution number, while others require different names. A simple way to resolve the problem is to specify a directory *template* in order to find the fonts. The template consists of the base directory followed by a subdirectory containing the special characters "$r" which will be replaced by the required horizontal resolution when the program is looking for a particular font. If for example your base directory is "c:\fonts" and the subdirectories are simply the resolution numbers, you should set the font directory to "c:\fonts\$r". If on the other hand the subdirectories are of the form "100dpi", "121dpi", etc., you should set the font directory to "c:\fonts\$rdpi". This method should help resolve conflicts among various DVI drivers. Whenever you change the font directory, the program tries to ensure that you have specified the $r component in the template. If the specification is incorrect, the program will not be able to find any PK fonts, so nothing will be displayed if you rely on such fonts. Similarly, you will get a warning if you specify $r more than once. The special sequence $(xxxx) or ${xxxx} will be substituted by the contents of the environment variable "xxxx".

When dealing with devices having unequal horizontal and vertical resolutions, you will probably want to be more explicit in the resolution specification. If for example you want to use 24-pin dot matrix printer at both 360x180dpi and 360x360dpi, Dviwin needs to differentiate between the two sets of fonts. For this

reason, the program also understands the special sequences $x and $y which will be substituted by the requested horizontal and vertical resolutions respectively ($x is just a synonym for $r). If you put for example the PK files at the directories c:\tex\fonts\360-180, c:\tex\fonts\360-360, etc. and you specify the font template as "c:\tex\fonts\$x-$y", dviwin will be able to use the correct fonts for both printer modes. Note that such a scheme may also be necessary if you want to use a 24-pin printer at 360x180dpi mode as well as a laser printer at 300dpi, because the laser fonts at magstep1 have the same horizontal resolution as the 24-pin fonts at magstep0.

The specification of the base directory is different from the first version of the program (2.0); if upgrading from that version, make sure that you add the suffix \ $r to your base directory.

The program can also use font libraries (fli files) produced by emTeX. For each font that it opens, it will look first in all font libraries in the base directory and if it does not find the required font, it will search for PK files using the normal template. *Note however that the font libraries distributed with emTeX consist of printer fonts only; to display your documents, you will either need to add screen fonts to these libraries, or use printer fonts in conjunction with zooming.*

There is just one complication with FLI files; each font library contains a directory of all fonts in the library as well as their horizontal resolution; this works well in most cases, but dviwin needs to know the vertical resolution of each font when you use devices with unequal horizontal and vertical resolutions. There is a simple solution to this problem: the FLI format contains a comment field which is normally unused in the fonts distributed with emTeX. If we put in the comment field the aspect ratio of the device (defined as vertical resolution divided by the horizontal resolution), then dviwin can figure out the vertical resolution for each font in the library. You don't need to get any new libraries; you only need to add the appropriate comment by the command: 'fontlib -x"NNN" XXXX' where NNN is the aspect ratio (in the form of a simple real number) and XXXX is the FLI file. You can also put anything else you like in the comment field; dviwin ignores the rest; just make sure that you put the aspect ratio in the beginning.

In some cases you may want to keep fonts under different base directories; this can be useful if you are on a network; in that case you may want to have often used fonts on a local hard disk for faster access, while keeping rarely used fonts on a network drive to conserve some disk space. This can be easily done by specifying multiple templates separated by semicolons (this is identical to the format of the PATH statement under DOS). If for example you keep the heavily used fonts under the "c:\fonts" directory and you have some rarely used fonts under the "m:\tex\pkfonts", you can specify the template as "c:\fonts\$r;m:\tex\ pkfonts\$r".

The program first searches for a font in all font libraries; then it looks for PK files. If it still cannot find the font, it will try neighboring resolutions. It will also try to find the file "dviwin.sub" in the font base directory; that file lets you specify which font to open if it cannot find the desired font; the format of this file is quite simple. A sample line might be:

cmbx10 58 -> cmbx6 100

This line instructs the program to use the font cmbx6.pk at 100 dpi if it cannot find the font cmbx10.pk at 58 dpi. You can also substitute an entire font family with another. If for example you do not want to store the cmdunh10 family, you can specify:

cmdunh10 -> cmr10

This instructs the driver to use the font cmr10 (at the appropriate resolution) if it cannot find the font cmdunh10. There is also a new substitution format dealing with devices with non-square pixels. If for example you use a 9-pin printer at 120x144dpi mode but do not have the cmu10 font at the appropriate resolution, you could specify:

cmu10 120 144 -> cmu10 121 121

which instructs dviwin to use the 121x121dpi font instead of the one requested by TeX. I would not recommend any such substitutions though: this method is provided only for the sake of completeness.

If the program cannot find a requested font either directly or through substitutions, it searches the substitution file for an entry of the form:

default -> xxxx

This entry instructs dviwin to use the font xxxx if it cannot find any other font. The general idea is to produce some (not necessarily correct) output if a font is missing. The sample substitution file uses cmr10 as the default font, but you can specify any font that you like.

You can also use comments in the font substitution file: everything after a percent mark (%) is ignored. Finally note that font substitutions may let you view a document even when you don't have the appropriate fonts, but the results will not be visually pleasing. It is a good idea to check the log file for any font substitutions if you ever see anything wrong about the fonts in your document.

The specification of the font substitution file is *different* from the first release of the program (2.0): in that version, the font substitution file was called "texfonts.sub" and the resolution was specified in TeX units for compatibility with the Beebe drivers. The problem is that the TeX units are somewhat confusing since we normally deal with dpi. To avoid any further confusion, I renamed the file to "dviwin.sub" so that there are no clashes with any other programs expecting to use the file "texfonts.sub" with the old conventions. Inside this distribution you will find a sample substitution file with comments showing the equivalent substitutions under the old conventions. I apologize for this incompatibility, but the other method was simply too confusing. The sample substitution file also shows an easy method to handle fonts with long filenames (eg., lcircle10 and lcirclew10 which are longer than the 8-character limit under DOS).

The program does *not* support GF or PXL fonts; if you have such fonts, you can use some standard TeX utilities (gftopk or pxtopk) to convert them to the PK format. The program supports up to 256 characters per font. You can get a decent set of PK fonts for a variety of screen resolutions from the SIMTEL archives (files "dvivga2.zip" through "dvivga8.zip" in the directory pd1:<msdos.tex>). Printer fonts for several printers can be found under the directory tex-archive/systems/msdos/emtex at the CTAN hosts (ftp.shsu.edu, ftp.tex.ac.uk or ftp.dante.de).

If you prefer the zooming approach, you will only need the printer fonts. Otherwise, you will also need the base resolution and up to four magsteps for each device that you want to use; if for example you use a 640x480 mode, you will probably want to set the base resolution to 100dpi. Therefore, you will need the fonts at 100dpi, 110dpi (magstephalf), 121dpi (magstep1), 145dpi (magstep2), 174dpi (magstep3) and 208dpi (magstep4). If you use an 800x600 mode you will probably set the base resolution to 121dpi so you will need the fonts at 121, 132, 145, 174, 208 and 250dpi. If you use a 1024x768 mode, you

will probably set the base resolution at 145dpi, so you will need the appropriate fonts (according to the above pattern).

If you want to use a 9-pin dot matrix printer, you should be aware that most Windows drivers do not support the 240x216dpi mode of those printers; since the standard emTeX font libraries use that resolution, you will not be able to use them. I have generated a set of FLI files at the 240x144dpi mode (which is supported by the Windows drivers) with all the fonts for plain TeX, LaTeX (but not SliTeX) and some utility fonts used by Knuth (logo* and manfnt). You can find these files in the directory tex-archive/systems/msdos/emtex/fx_med_fonts at the CTAN hosts (ftp.shsu.edu, ftp.tex.ac.uk or ftp.dante.de). I have also added the aspect ratio in the comment field of each library, so dviwin will be able to use them without any problems.

Log Viewer: This option lets you specify the program that will be used to view the log file when you use the "Log" command from the main menu. You can specify any browser or editor that you like; you can also use a small browser that I wrote (wbr.exe); it is quite convenient, can use any font installed in your system, can search for regular expressions and can handle multiple files using the MDI standard. The browser is executed asynchronously (that is, you can switch between the browser and the DVI driver at will). The program also supports a simple substitution pattern for this command: the sequence "$l" will be substituted by the log file's name, while the sequence "$b" will be substituted by the name of the dvi file without the dvi extension. This simplifies the integration of dviwin with any browser and lets you looks at several related files quickly. If for example you always want to look at the dvi log file (.lg) as well as the TeX log file (.log), you can specify the Log Viewer command as:

wbr -l $l $b.log

The sequence $(xxxx) or ${xxxx} will be substituted by the contents of the environment variable "xxxx". There will also be cases where the TeX log file (.log) is unavailable, and the browser will complain; you can easily avoid such complaints by using the "-w" flag in the above command. Note that the name of the NT version of the browser is "wbr2.exe"

Custom Resolutions: The table in the "Resolution" submenu lets you use most common resolutions, but it cannot account for every possible device in the market. Therefore, the program lets you specify up to two custom resolutions for your particular devices; you can specify the horizontal as well as the vertical resolution of the device, so you are not constrained to devices with square pixels. Any positive numbers are acceptable, but make sure that you have the appropriate fonts for these resolutions. You can delete a custom resolution by specifying a zero value for the horizontal resolution.

Font Cache: When closing a file or loading another document, the program does not discard the old fonts; it keeps them in a cache, so the next time that it needs the same font, it can take it from there without accessing the disc; this can improve the speed significantly, but it takes some memory. For this reason, you can control the size of the cache which can range from zero up to fifty; a value of 20 should be enough for most documents. When setting the size of the cache, you can also see the number of cached fonts; this information should help you determine an appropriate size for the cache.

Scroll Factors: The program lets you select the amount by which the window will scroll when you click on the scrollbar or press the arrow keys. The first scroll factor applies to the left mouse button on the scrollbar (or the normal arrow

keys). The second scroll factor applies to the left mouse button on the small arrows at the end of the scrollbars (or the shift-arrow keys). These factors are expressed in percentages of the window dimensions. For example, a value of 80% for the first scroll factor means that when you click on the vertical scrollbar or press the up or down arrow, the window will scroll by 80% of its height (ie., there will be a 20% overlap). If you click on the horizontal scrollbar, the window will scroll by 80% of its width.

Magnifying Glass: The program can also magnify a small area of the page for easier inspection by clicking the middle mouse button (or the left+right buttons). When the zoom factor is greater than one, the program simply displays the "unzoomed" bitmap; when however the zoom factor is equal to one, the magnification is done through simple bitmap expansion for performance purposes. The magnifying glass can be dismissed by pressing any key or clicking any mouse button on its surface.

This dialog lets you specify the size of the magnifying glass. The dialog also contains an option called "Dynamic update"; if you turn it on, then the contents of the magnifying glass are updated whenever you move it. If this option is turned off, the magnifying glass contents are not updated when you move it (they are updated when you click the left mouse button). This behavior is useful if you want to compare the close-up view to the normal view (otherwise, the normal view would be obscured by the magnifying glass).

Page Layout: This is another dialog where you can select the page size; it defaults to the standard "Letter" size (8.5 x 11 inches), but you can use most popular paper sizes including the metric ones (eg. A4). You will also see a checkbox called "Landscape". When this box is checked, dviwin uses a page whose size os the transpose of the one selected from the list (if for example you select the Letter size from the list and check the box, dviwin will use a 11x8.5in page). Note that this checkbox simply increases the available page sizes; it does NOT control the actual orientation of the printed output. If you want to actually print in landscape, you need to specify this in the Printer Setup dialog from the File submenu. You can also specify the measurement system used to report dimensions: the "English" option reports dimensions in inches. The "Metric" option uses millimeters. The "Automatic" option selects inches or millimeters depending on the page size.

Color Interpolation: This entry lets you fine-tune the interpolation used by the zooming routines. The adjustment is similar to gamma correction, and you will find that it has a significant impact on the readability of the text. The displayed text is updated as you are adjusting the value; this should facilitate the selection of a suitable value for your system.

Missing fonts:This entry lets you specify the action that dviwin will take if a font is missing. There are three options: the first one is to take no action. The second option is to execute a command to generate the required font; this command is usually a program or a batch file that calls Metafont to generate the font. After the command is executed, dviwin tries again to locate the required font; if the command was successful, dviwin will find and use the font. This option is very helpful if we do not want to keep most standard TeX fonts on disk; the basic idea is to start with a small set of fonts and generate only the ones that we need as we encounter them. The advantage of the method is that the used fonts are only a small fraction of all the TeX fonts, so we save some disk space; the disadvantage is that Metafont is rather slow, so we may have to wait quite a while

if there are many missing fonts in a dvi file. The dviwin distribution contains the sample files "genpk.pif" and "genpk.bat" which demonstrate how you can automatically generate the missing fonts (the corresponding file under Win-OS/2 is "genpk2.cmd"). If you decide to use this option, you will first need to setup Metafont (ie., set the correct environment variables and generate the plain base including the mode information for various devices) and gftopk, and modify the sample file "genpk.bat" (or "genpk2.cmd") to store the generated fonts at the appropriate location, so that dviwin can find them. Some user customization will yield the best results and that's why genpk.bat (or genpk2.cmd) are batch files instead of compiled programs. The program switches to the directory specified by the TEMP environment variable right before executing the specified command (you can override this from the batch file). There is one thing that you should be aware of: after dviwin executes the desired command, it searches again for the font in the PK format only, but it does not re-read any FLI files; if your batch files create or update FLI files, use the "Reset all fonts" command from the "Options" submenu to force dviwin to re-read the FLI files.

The above option may be too slow on some machines,and it may not work very well under certain circumstances (eg., when running the 16-bit version of dviwin under NT, or the 32-bit version under normal Windows); in those cases, dviwin does not wait for Metafont to complete, and will spawn multiple copies of Metafont if there are more than one missing fonts. The third option tries to circumvent these problems; you specify a command to be executed as well as the name of a batch file; then dviwin just appends the command to the batch file; in this way, you can collect several font generation requests and run the batch file overnight. The last entry field under this method accepts a command to be executed after all the missing font requests for a given document have been collected. If this field is empty, then you will run the batch file manually at your convenience. If the field is non-empty, then dviwin will execute the specified command, and you can still preview the document during the font generation. Here are the appropriate commands for various environments:

Environment	**Command**
Windows 3.x	genall.pif $(TEMP)\missing.bat
Windows NT	genall.pif $(TEMP)\missing.bat
Win-OS/2	genall2.cmd $(TEMP)\missing.cmd

Under Win-OS/2, you also need to set the "File" field to "$(TEMP)\missing.cmd" instead of "$(TEMP)\missing.bat".

After these batch files generate all the fonts, you will want to use the "Reset all fonts" command from the "Options" submenu to read the correct fonts. This can also be done automatically under certain circumstances: the command line switch "-r" instructs dviwin to reset the fonts. Therefore, the batch file can issue the command "dviwin -r" when all fonts have been generated (in theory at least). In practice, this will not work under normal Windows, because DOS batch files cannot invoke Windows programs; it will work however if you rewrite the batch file in one of the Windows scripting languages. NT and OS/2 have no such restrictions, so genall.bat and genall2.cmd do the right thing (you may need to run dviwin in seamless mode under OS/2).

The command to be executed in the last two options is actually a template where the following special sequences will be substituted by the appropriate values:

$f Font name (truncated to 8 characters)

$r Horizontal resolution
$x Horizontal resolution (same as $r)
$y Vertical resolution
$X Horizontal *base* resolution
$Y Vertical *base* resolution
$m Magstep
$(xxx) Contents of environment variable "xxx"
${xxx} Same as $(xxx)
$d Letter of disk drive in the directory containing the dvi file
$p Path (without the drive) of the directory containing the dvi file
$e Extension of Metafont output file (truncated to 3 characters)

When generating fonts synchronously (ie., when using genpk.bat instead of genall.bat) you can interrupt the font generation process by pressing Control-Break. Dviwin will then set the font generation method to "Take no action", so it will not attempt to generate any other fonts; you will still need to wait a bit for Metafont to finish the current font.

Inverse Search: This dialog lets you specify the commands issued by DVIWIN in the course of an inverse search. When you double-click the left mouse button on the DVI file surface, DVIWIN tries to infer the appropriate position in the source file (the presence of src specials is required), and then issues the appropriate command or DDE message. You can specify a command to be issued, whether the editor supports DDE, and the DDE service, topic and messages if applicable. If the editor does not support DDE, then the specified command will be issued. If on the other hand the editor supports DDE, dviwin will try to send a DDE message using the specified service, topic and first message. If no DDE server responds to that message, then dviwin issues the specified command (presumably to start the DDE server), and then sends another DDE message using the specified service, topic and second message (or first message again if the second one is empty). This is the most flexible mechanism supported by Windows and you can invoke practically any editor that acts like a DDE server. There are two special sequences are recognized in the DDE messages:

$f Name of corresponding source file
$l Line number of corresponding source file

If for example you want to use PFE, you can specify the fields as:

Command: pfe.exe $f
Service: PFE (or PFE32 for 32-bit version of PFE)
Topic: System
Message 1: [FileOpen("$f")] [EditGotoLine($l,0)] [EditSelectLine()]
Message 2:

Log This entry displays all messages generated from processing the current DVI file. I took a rather unusual approach to implement this feature: the program always writes the messages on a temporary file (its name is related to the process ID, so there is no conflict between multiple copies of the program running simultaneously). When we select the "Log" entry, the driver runs a browsing program asynchronously and instructs it to read the log file. The advantages of this approach are that it was much easier to code, the browsing program is better than an embedded browser, and we can switch between the two programs at will, since we run it asynchronously. The only disadvantage is that the browser displays only the log file that was current at the time that it started; therefore, new messages will not appear automatically. If however you select the "Log" entry again, the DVI driver will start the browser again with the

latest version of the log file. The browser defaults to "wbr.exe", but you can use any browser or editor that you like by changing the "Log Viewer" entry in the "Options" submenu.

Moving around

There are many ways to position the current window using either the mouse or the keyboard. When moving within a page, you can move by two different factors or a single pixel. The default settings associate the first factor with a large movement and the second factor with a small movement, but you can always adjust them from the "Options" submenu; therefore, it is better to talk about a "primary" and a "secondary" factor (instead of a "large" and a "small" factor). You can also move directly to each margin or to each corner. The following table lists the available commands and the mouse and keyboard bindings.

Operation	**Mouse**	**Keyboard**
Move by the primary scroll factor	Left button on scrollbar	Arrow keys
Move by the secondary scroll factor	Left button on small arrow at end of scrollbar	Shift-Arrow keys
Move by a single pixel	Right button on small arrow at end of scrollbar	Control-Arrow keys
Move to margin	Middle button on scrollbar; if you click on the upper portion of the scrollbar, it goes up; otherwise it goes down	Alt-Arrow keys
Move to corner	None	Home/End, Control-Home/End
Next/Previous page	Click on the last two buttons of the main menu, **or** Right button on vertical scrollbar; if you click on the upper part of the scrollbar, it goes to the previous page; otherwise, it goes to the next page	PgUp/PgDn
First/Last page	None	Control-PgUp/PgDn

If you press the control key and click the left button at any location, the viewer will scroll the window down and to the right, so that the selected location is on the top left corner. If you press the control key and click the right button at any location, the viewer will scroll the window up and to the left, so that the selected location is on the bottom right corner. If you press the control key and click *on* a ruler, you will move only on the direction of that ruler.

Finally, there is a keyboard-only interface for those accustomed to the "more" program. The space bar produces a forward movement by the primary factor, while the Enter key produces a forward movement by the secondary factor; both keys however, move you to the next page if you are at the bottom of the page. The Control-space and Control-Enter (or the "b" and "d" keys for the Unix diehards) do the opposite of the above two commands; that is, they move you backwards. The main advantage of these commands is that they let you go through the entire file using only a single key.

Specials

TeX does not support graphics directly: there is a provision for "\special{}" commands that are optionally interpreted by DVI drivers, but there is no consensus about the format and the contents of these commands. This driver accepts its own commands for inserting arbitrary graphics files in your document; it also accepts tpic and emTeX specials. Any unrecognized special commands are ignored.

1. Graphics Files

We deal with various graphics files using two techniques: the first approach is to use Windows metafiles; these files provide several advantages: they can be scaled to various resolutions, can contain almost any graphics command, are reasonably fast and compact, and are supported by most Windows applications and the system itself. There is one complication with metafiles though: most applications do not produce disk files in this format; they do however use this format for transfers to the clipboard. Therefore, we can capture the contents of the clipboard and save them to a metafile, which can later be used in our TeX documents. This task is done by the utility "clipmeta"; its operation is very simple: when you run it, it just prompts you for the filename to be used for the metafile. I tested this technique with Excel, PowerPoint, Word for Windows, 1-2-3 for Windows, Toolbook and Paintbrush; it worked perfectly in all cases.

There is also another method for importing graphics, which was developed for PageMaker and is also supported by PowerPoint, Word for Windows, Toolbook and other programs. An application can use a dynamic link library, which is responsible from importing the specified file; the application can then simply display the imported graphic. The dynamic link library is called a graphics filter, and the above mentioned applications come with a collection of such filters for various formats; this approach eliminates the need for an intermediate metafile. The DVI driver comes with filters for PCX, BMP, MSP, (non-interlaced) GIF and XPM files; apart from this, it can use any standard filter. Therefore, you can import any file, provided that you have the appropriate filter (either from dviwin or from the above mentioned applications). The DVI driver looks in "win.ini" for the entries under the section "[MS Graphic Import Filters]" (this is the same method used by Microsoft programs). The entries in this section should be in the form "*description=driver,extension*", where "*description*" is an arbitrary string describing the supported format, "*driver*" is the full pathname of the graphics filter and "*extension*" is the extension used by the format. Therefore, you can install the dviwin filters by adding the following lines to your win.ini file:

```
[MS Graphic Import Filters]
PC Paintbrush(.PCX)=XXXXXX\pcxin.flt,PCX
Microsoft Paint(.MSP)=XXXXXX\mspin.flt,MSP
Bitmaps(.BMP)=XXXXXX\bmpin.flt,BMP
Bitmaps(.DIB)=XXXXXX\bmpin.flt,DIB
Bitmaps(.RLE)=XXXXXX\bmpin.flt,RLE
Compuserve GIF(.GIF)=XXXXXX\gifin.flt,GIF
X Pixmaps(.XPM)=XXXXXX\xpmin.flt,RLE
```

where XXXXXX is the directory where you want to place the filters (it does not have to be in the path). I also tested some of the filters that come with Word for Windows, and they appear to work properly, except that the eps filter does *not* display PostScript files; instead, it imports TIFF or Metafile pictures embedded in the PostScript file; when however you print the graphic on a PostScript printer, then it sends the actual PostScript image.

There is one more thing that you should be aware of: Aldus developed an extended metafile format (called a "placeable" metafile), which circumvents some limitations of standard metafiles. Unfortunately, they chose the same extension (wmf) as the one for standard metafiles, and this can be the source of some confusion. The DVI driver has intrinsic support for both kinds of metafiles, but the placeable format is recommended. If the imported graphic contains text, you will get much better results by using scalable fonts (TrueType or ATM).

2. Special commands

The general form of the special command is

\special{*action filename, x-paper-size y-paper-size*}

where:

"*filename*" is the name of the graphics file; if you specify an extension, dviwin looks for the exact filename only; if on the other hand you do not supply any extension, it tries to find any graphics file with the specified name. Note that the filename must be followed by a comma. If you need to import graphics files from other drives or directories, you can use full pathnames, but you should use forward slashes (/) instead of the customary backslashes (\), because TeX will complain about undefined commands.

"*x-paper-size*" and "*y-paper-size*" are the desired width and height on paper. Both sizes consist of a positive number followed by a standard TeX dimension; valid dimensions are:

String	Name	Conversion
in	inch	1in = 2.54cm
bp	big point	72bp = 1in
pt	point	72.27pt = 1in
pc	pica	1pc = 12pt
sp	scaled point	65536sp = 1pt
dd	didot point	1157dd = 1238pt
cc	cicero	1cc = 12dd
cm	centimeter	2.54cm = 1in
mm	millimeter	10mm = 1cm
px	pixel	

"*action*" is a string indicating how the driver should insert the graphics file in the document; if it is "center", the driver puts the specified file (using its "natural" dimensions) at the center of the indicated rectangle. If the natural dimensions of the graphic are larger than the indicated rectangle, the graphic will be clipped to the rectangle's dimensions. If the action string is "isoscale", the driver will scale the graphic, so it just fits in the indicated rectangle, but it will use the same scale on both axes, so the aspect ratio remains the same and there is no distortion. If the action string is "anisoscale", the driver will scale the graphic to fit in the indicated rectangle without regard for the aspect ratio.

Suppose that you want to insert the metafile "fig.wmf" at a given point in the document, you want to scale it so that it takes 4.8 inches horizontally and 3.6 inches vertically, and you do not care about the aspect ratio. This can be easily done by the command: "\special{anisoscale fig, 4.8in 3.6in}". You will also need to leave some empty space for the graph (after the special command). A sample LaTeX macro might be:

```
\def\myfigure#1#2{
        \begin{figure}[ht]
        \caption{#2}
        \special{anisoscale #1, \the\hsize 3.6in}
        \vspace{3.6in}
        \end{figure}}
```

This macro takes two parameters: the filename and the caption of the figure. It uses the \hsize value to adjust for varying text widths. This is just a sample macro; you can generalize it, or you can customize it more for particular types of graphics files. Whenever the driver imports a

graphics file, it produces an entry in the log file with the name of the graphics file and its natural dimensions. This information can be useful in determining the amount of space to leave on the document.

As mentioned above, the placeable metafiles provide information about the size of the image, while the standard ones do not. Therefore, it would be impossible for the DVI driver to place such metafiles correctly: the only possible action would be to "anisoscale", ie., to just fit the graphic into the space that you specified. Therefore, I would suggest that you always instruct "clipmeta" to produce a placeable (instead of a standard) metafile. There are also cases where you want to override the dimensions specified in the metafile; in your TeX document, use the command:

\special{*action filename, x-paper-size y-paper-size x-fig-size y-fig-size*}

instead of:

\special{*action filename, x-paper-size y-paper-size*}

where "*x-fig-size*" and "*y-fig-size*" are the desired dimensions. This method is particularly useful when the graphics file contains bitmapped data, which are prone to geometric distortions. The best way to produce good looking bitmaps is to compute the bitmap dimensions with respect to the printing resolution and supply these values as the last two parameters in the \special{} command. Suppose for example that you want to print a 900x600 bitmap on a 300dpi printer; the bitmap dimensions (when printed) should be 900/300 = 3in by 600/300 = 2in. If you supply these values as the last two parameters in the \special{} command, the bitmap will be free of any geometric distortions. Note that you *need* to supply these values: the graphics filter cannot supply them for you since it does not know the resolution at which you want to print the bitmap. There is also a shortcut for the above method: if you specify the bitmap dimensions in pixels (px), dviwin will compute the actual (physical) dimensions according to the current resolution.

3. emTeX specials

The DVI driver understands the standard emTeX specials as described in the documentation of the emTeX 1.4 drivers. The following restrictions and extensions apply:

1. The 16-bit version of the program lets you define up to 10,000 points in a single page. The 32-bit version of the program increases the limit to 10,000,000 points.
2. The line cut instructions (h,v,p) are ignored.
3. You need to define a point with "em:point" before using it with "em:line".
4. The "em:graph" command is not restricted to BMP, PCX and MSP files; it will work with any graphics file, provided that you have the appropriate filter.
5. The pen size and graph dimensions can be specified using any valid TeX units, including pixels.
6. The maximum line width is 1 inch.
7. If the specified filename in an "em:graph" special has a path component, then dviwin will use that component (the dvidrv documentation says that the path component is ignored).

4. tpic specials

The program can also handle tpic specials according the tpic 2.0 specification. The following restrictions, extensions and clarifications apply:

1. The 16-bit version of the program can handle up to 16,000 segments per line or polygon. This limit increase to 16,000,000 for the 32-bit version.
2. The "tx" special is converted to an equivalent shading request.
3. If no pen size has been specified for a particular shape, the last specified pen size is used. The initial pen size is a single pixel.
4. The maximum pen size is 1000 milli-inches.

5. The program supports up to 21 gray shades (including black and white). All shades except black and white are transparent. If you use a color printer, you will get a true gray instead of a dithered pattern.
6. You cannot apply shading to a spline since it is not a closed figure.
7. If you request shading on an open polygon, dviwin first closes the polygon and then does the requested shading.
8. If you request shading on an incomplete arc, then dviwin draws either a shaded chord, or a shaded pie slice. The "ar" command takes an optional number after its normal parameters; if the number is missing or is equal to zero, then dviwin draws a chord; if the number is equal to one, then dviwin draws a pie slice.
9. Windows has a bug when drawing incomplete arcs: if the pen size, the arc radius or the resolution is high enough, the arc will be drawn with an invisible pen. You will not encounter this bug under normal use; if you do encounter it, there is a workaround: edit the file "dviwin.ini" (or dviwin2.ini for the 32-bit version) and set the option "winarcs" to zero (ie., it should read "winarcs=0"). This instructs dviwin to approximate the arc using multiple line segments; this is not as neat or fast as the standard Windows method, but it works under all circumstances.

5. Other considerations

There are a few other points that may be of interest:

1. The program first typesets all TeX text, and then does the specials in the same order as they are encountered in the dvi file. This is relevant only when the shading is not transparent.
2. emTeX or tpic line drawing specials have no clear indication for the end of the special; therefore, dviwin treats all line drawing specials as a single unit until it reaches the end of the page, or until another special command of a different kind is encountered. This implies that you can mix all sorts of special commands in a single page, but each different type finishes the previous special and starts another one.

Printing

The Print dialog lets you select the pages to be printed as well as the resolution to be used. You can specify an explicit value (from those listed in the resolution table), or you can set it to "Auto" which instructs dviwin to select the one closest to the resolution of the selected printer. The "Setup" button leads you to another dialog from where you can select or configure the active printer.

There is also an option called "Send specials to printer"; if it is off, the program draws all special commands to the page bitmap (just like when you view the file), and then sends the entire bitmap to the printer. If you turn that option on, the dvi driver will send any specials directly to the printer instead of putting them on the bitmap. The advantage of this approach is that it exploits any special capabilities of your printing device. If for example you have a color printer and you want to print a graphic in color, then you only need to check this option. Similarly, you can take advantage of any other special capabilities of a particular printer driver (eg., halftoning, dithering or intensity control from the Setup dialog of the printer driver). If your printer has unequal horizontal and vertical resolutions, this option will produce correct results for Windows fonts inside your graphics files. The disadvantage of the method is that it may be slower, and the program will not be strictly WYSIWYG (all specials will still be monochrome while previewing: this conserves memory and increases the speed of the program).

There are cases where you may want to use a printer-specific dvi driver for printing. Dviwin tries to facilitate this combination by providing a hook in the Print dialog; you will see a section called "External print" with an edit field. In that field, you can enter a template that specifies a command to be executed when you instruct dviwin to print. There are several special sequences that will be substituted right before the command is executed:

$1	Number of first page to be printed
$2	Number of last page to be printed
$3	Difference between pages; it will be equal to one when printing consecutive pages, or two when skipping every other page.
$r	Horizontal resolution
$x	Horizontal resolution (same as $r)
$y	Vertical resolution
$X	Page width (in inches)
$Y	Page height (in inches)
$b	Basename of dvi file (without extension)
$Rxxx	This will insert the string xxx if the pages are in reverse order
$Sxxx	This will insert the string xxx when you select to skip every other page
$(xxx)	Contents of environment variable "xxx"
${xxx}	Same as $(xxx)

If you do not specify the $b sequence, dviwin will automatically add it to the end of the template.

Suppose for example that you want to run dvips (version 5.49 or later) on the current file, copy the PS file to the printer and then delete the PS file. This can be done in the following way:

1. Write a batch file called rundvips.bat containing the lines:
   ```
   dvips -p=%1 -l=%2 %4 %3
   copy %3.ps prn /b
   del %3.ps
   ```
2. Make a PIF file called rundvips.pif for the above batch file
3. Specify the external print template in dviwin as:
   ```
   rundvips.pif $1 $2 $b $R-r
   ```

If you want to use dvijep (or dvialw or any of the Beebe drivers), you can follow these steps:

1. Write a batch file called rundvije.bat containing the lines:
   ```
   dvijep -o%1:%2:%3 %5 %4
   copy %4.jep prn /b
   del %4.jep
   ```
2. Make a PIF file called rundvije.pif for the above batch file
3. Specify the external print template in dviwin as:
   ```
   rundvije.pif $1 $2 $3 $b $R-b
   ```

The dvijep example gives you the entire functionality from the Print dialog of dviwin (ie., page selection, reverse order and skip of every other page). The dvips example does not give you the last feature (the skipping of every other page), because I could not figure out how to instruct dvips to do it.

There are wide variations in the parameter format of various dvi drivers, but I hope that these sequences are general enough to satisfy most of them; if you write a batch file to invoke other dvi drivers, please send me a note, so I can help other users facing the same problem.

Apart from the dialog template, you will also see two checkboxes: the first one enables or disables the external printing; the second one lets you preview or edit the external command right before it is executed.

There are some more things that you need to consider: when dviwin executes an external printing command, it closes the dvi file but it does not close any font files. The other dvi driver will also

need to access the font files, so you may run into three problems: the first one is that there may not be enough file handles available in the system; this can be easily solved by increasing the value in the "FILES=..." statement in your config.sys file. The second problem can arise if you are using the share program; in that case, you may encounter many sharing violations; this can be solved by setting the file attributes of the font files to "Read Only". The last potential problem can happen if you switch to dviwin while the other dvi driver is still executing; if share is loaded, then you will get a sharing violation about the dvi file and dviwin will complain that it cannot access the file. There is no easy solution to this problem: you either have to wait until the other dvi driver has finished (which should take only a few seconds), or avoid using share.

Printer Alignment

If a printer has a resolution of 300 dots per inch and can print on standard "Letter" paper, you would expect to be able to use 2550 (8.5 x 300) pixels horizontally, and 3300 (11 x 300) pixels vertically. Most printers cannot print on the entire page, and Windows reports a smaller number of available pixels. For example, the HP LaserJet II reports 2400x3160 pixels, the DeskJet reports 2400x3130 pixels, and the Apple LaserWriter Plus reports 2394x3231 pixels. This is not a big problem, since any document has more than adequate margins; if however you want to position the text accurately, you need to know how the missing pixels are divided among the four sides of the paper. The DVI driver assumes that the missing pixels are symmetrically distributed; for example, on a LaserJet II, it assumes that there are 75 missing pixels on each vertical side, and 70 missing pixels on each horizontal side. If this assumption is wrong, you can set the missing pixels manually; this is done by using the "Print Setup" entry in the "File" submenu and selecting the "Align" button on the setup dialog. At that point, you have to enter the missing pixels on the left and top sides (printer origin). Your selections are stored in dviwin.ini (or dviwin2.ini for the 32-bit version) under the section "[Printer Origin]". When using the manual option, you may need to experiment in order to find the correct number of pixels. This can be easily done by processing the plain TeX file:

```
\nopagenumbers
\parindent = 0pt
\vbox{
        \hrule height 0.01pt
        \vrule width 0.01pt height \vsize
        \hfill
        \vrule width 0.01pt height \vsize
        \hrule height 0.01pt}
\end
```

which produces an empty box containing the entire printable area by plain TeX. If the alignment is correct, the upper left corner of the box should be at 1 inch left and 1 inch down (relative to the upper left corner of the paper), while the lower right corner should be at 7.5 inches left and 9.9 inches down. I run alignment tests on several printers, and settled on the following values:

Printer	Horizontal	Vertical
HP DeskJet	75	60
HP LaserJet II	65	74
HP LaserJet IIIP	75	75

Keep in mind that many printers are not very accurate in positioning the paper, so you should be very careful if you need pinpoint accuracy.

Interaction with TeX and other programs

The program tries to simplify the usual "edit-compile-view" cycles; whenever it loses the input focus (this happens when you minimize it or switch to another program), it closes the current DVI file but remembers its position within that file. When it receives the input focus, it checks for any changes in the DVI file; if there are no changes, it continues from the same point. If on the other hand the file has changed (presumably by TeX), it reloads the file and positions you at the same place as before; it also tries to use as many of the old fonts as possible in order to minimize the "refreshing" time. This approach lets you switch from dviwin to the editor (or TeX) and back easily with no need for complicated interprocess mechanisms (so you can use it with any version of TeX). The only thing that you should *not* do is switch to dviwin while TeX is still running, because the DVI file is not yet valid; the program will complain if this happens.

The program can also accept commands from another program. There are several command line switches to trasmit requests from the other program to dviwin. The formal usage is:

dviwin [-1] [-c] [-g NN] [-s 'NN XXXX'] [-r] [filename]

where items in brackets are optional. The "-1" switch requests a single instance of the program; if for example you write an editor macro to run dviwin, you don't need to check if dviwin is already running; if this is the first instance of the program, the "-1" switch is ignored; otherwise, the second instance of dviwin sends any requests to the first instance and terminates. The "-c" switch instructs dviwin to terminate itself. This switch makes sense only when combined with the "-1" switch, and only when another instance of dviwin is already running; otherwise, it is ignored. The "-g NN" switch instructs the program to go to page NN; if you also specify a filename, it will load that file and then go to the requested page; otherwise, it will go the requested page of the current document (in this case, you want to transmit this request to the first instance of the program, so you also need the "-1" switch).

The "-r" switch instructs dviwin to flush the font cache and re-read all fonts. This can be useful when generating many missing fonts in one pass; the "Missing fonts" section describes this in more detail. If the first instance of dviwin has been started with the "-1" switch, then you need to specify the "-1" here too. If on the other hand you did not specify the "-1" switch when you started the first instance of dviwin, then you should not specify it here either.

There are two common problems while previewing a DVI file: the first one is to find the position in a DVI file that corresponds to a given position in the TeX source file (we'll call this the "forward search" hereafter). The second (and probably more useful) one is to find the position in the TeX source file that correponds to a position in the DVI file (we'll call this the "inverse search"). Some DVI drivers have attempted to solve these problems by analyzing the contents of the DVI file and trying to deduce the contents of the TeX source files. I would suggest however that this approach will work only under very limited conditions, because it has to assume a certain encoding of the TeX file (ie., this may work with cmr fonts, but it may fail with other fonts), it cannot handle anything but simple text (ie., it cannot work with equations, tables, etc.), it cannot find text expanded by a macro (if it does find the text, it will find it in the macro definition instead of the macro usage), etc.

For the above reasons, I would propose an alternative approach: if we view TeX as a normal compiler, we will see that the same problems are faced by source level debuggers (ie., they have to correlate the object code with the source code); the standard technique for solving this problem is to insert some symbolic information in the object code; this extra information does not affect the behavior of the object code, but it does provide the relevant information to the debugger. One can apply the same technique to DVI files by exploiting the \special{} mechanism. Suppose that we insert specials of the form:

\special{src: NN XXXXX}

where XXXXX is the current source file and NN is the current line number within that file. If the DVI driver is aware of this kind of specials, it will be able to correlate the source file with the DVI file; if on the other hand, the DVI driver does not recognize these specials, then it will simply issue some warnings, but its behavior will be otherwise unaffected.

DVIWIN is able to perform both kinds of searches (forward and inverse) if the DVI file contains the above described src specials. The command line switch "-s 'NN XXXX'" instructs DVIWIN to locate the position in the DVI file that corresponds to line NN of the source file XXXX. The program will complain if no DVI file is open, or the DVI file does not contain any src specials. If the search is successful, the program positions the mouse cursor at the correct location of the DVI file. The inverse search is done by double-clicking the left mouse button. The program then tries to figure out the corresponding location in the TeX file and issues a command or a DDE message to an editor in order to display the appropriate text. You can specify the command or DDE message in the "Inverse Search" dialog from the "Options" submenu.

The main advantage of the above described method (ie., the src specials) is that it can work with arbitrary text; it does not depend on any particular encoding, can work in math mode, with tables, etc. The main problem is how to generate these specials automatically. There are three obvious methods: the first one is to modify TeX itself to emit these specials (we should probably use a command line switch to enable this feature in order to remain fully compatible with other TeX versions). The second method is to modify the plain and LaTeX formats to emit these specials. The third method is to write some TeX macros and include them in our documents. All three methods have their advantages and disadvantages. I would suggest that the first or the second method are more convenient, because they eliminate the need of modifying any TeX documents. The disadvantage of the second method is that we will always need to modify any new formats, in contrast with the first method which requires no changes in either the documents or the formats.

The last issue about the src specials is their frequency and placement in the DVI file. The higher the frequency, the better the correlation between the TeX and DVI files (ie., we would like to have a src special for each line in the DVI file); on the other hand, these specials increase the size of the DVI file, and may slow down the DVI driver, which may be a consideration under certain circumstances. For this reason, I would propose to emit src specials at the following occasions:

1. Beginning of file
2. End of file
3. Beginning of page
4. End of page
5. Right before including a file
6. Right after including a file
7. Between paragraphs
8. Beginning of a column
9. End of a column

This strategy enables the DVI driver to do both searches and it does not increase the size of the DVI file excessively. DVIWIN also supports a modified form of the src special: if one omits the filename, then DVIWIN assumes that it is the same as in the previous src special. This reduces the size of each src special to about 10 bytes which is rather frugal. The only requirement is that the first src special in each page should contain a filename.

User defined strings

All strings of the program (messages, menu entries, etc.) reside in the file "dviwin.str" which is a plain ASCII file; this lets you modify all strings and optionally translate them to various languages. The string file should be located either in the current directory or someplace in your path; the program reads it upon startup. The format of the string file is very simple: it is composed of strings that are separated by zero or more spaces, tabs or newlines. Each string is delimited by double quotes (") and the backslash character (\) is an "escape" character as in the C language; the program recognizes the following escape sequences:

\a	Introduces the ASCII bell character (ASCII 7)
\b	Introduces a backspace (ASCII 8)
\f	Introduces a formfeed (ASCII 12)
\n	Introduces a newline (ASCII 10)
\r	Introduces a carriage return (ASCII 13)
\t	Introduces a tab (ASCII 9)
\xNN	Introduces a character whose code is equal to NN, where NN are two hexadecimal digits. NN cannot be equal to "00".

Any other character after a backslash is accepted literally (ie., it is included in the string); for example, the sequence \\ introduces a single backslash into the string and the sequence \" introduces a double quote. The only exception is the sequence \<Actual Newline> which simply continues the string to the next line; this can be convenient for long strings.

The string file can also include comments: each comment is introduced by a percent character (%) and continues to the end of the line.

Some of the strings are used as templates for the "printf" function, so you may encounter a percent character followed by various characters. These sequences are interpreted as following:

%s	will be replaced by a string at runtime
%u	will be replaced by an unsigned integer at runtime
%d	will be replaced by a signed integer at runtime
%ld	will be replaced by a long signed integer at runtime
%g	will be replaced by a floating point number at runtime
%%	will be replaced by a single percent character at runtime

If you do modify strings that contain any of the above format specifiers, **make sure** that you do not introduce any new specifiers or change the order or type of existing ones; the only valid modification is to delete one or more specifiers provided that they are the last ones in the string. The program checks the validity of each string and will complain if any of the above rules are violated. Each string can contain up to 4096 characters. There is no limit on the total number of characters in the string file. The program assumes that the string file uses the ANSI (Windows) character set as opposed to the OEM (DOS) character set; this is relevant only for characters whose code is greater than 127. If you do use such characters, make sure that you edit the file with a Windows editor instead of a DOS editor.

Some of the strings may be used as menu entries; these strings often contain an ampersand (&) which indicates that the following character should be underlined. The sequence && introduces a single ampersand in the menu entry.

As mentioned above, the program reads the strings from the file "dviwin.str" upon startup. One can override this default string file by using the command line switch "-m". If for example you execute the command:

dviwin -m special.str

the program will read the strings from the file "special.str" instead of "dviwin.str". This facilitates

the switching among various versions of the string file.

The program comes with several versions of the string file for a few languages (these files are included in the file "strings.zip"; the file "readme.str" inside the zipfile contains more details about these files). The base file "dviwin.str" contains comments about the meaning of each string; these comments should be helpful when adapting the string file to another language. If anybody translates the strings to his native language and is willing to share it with the rest of the program's users, I will be happy to distribute the string file with the upcoming versions of the program.

Caveats

The DVI driver prints a document simply by sending a bitmap to the Print Manager. This technique is a bit simplistic, but it works on any printer supported by Windows and it is guaranteed that the printer output will be identical to the screen output. The only possible optimization is to avoid sending long sequences of whitespace (it already does this). There are much more efficient methods for printing a DVI file to some printers (eg., downloading fonts to most laser printers), but these methods do not work on *all* printers (on the other hand, my method is optimal for dot-matrix and inkjet printers or fax drivers); I prefer to use the most general technique possible in order to avoid any printer dependencies. If you use a PostScript printer and are not satisfied with this approach, you can attach a dedicated dvi driver for printing as outlined in the previous section (dviwin is still good for previewing); I cannot afford to tailor the program to every possible printer; this is supposedly the job of the operating system. I can promise you however to work on changes that will benefit the majority of the program's users.

The program relies on the Windows drivers for screen and printer output; the problem is that these drivers are often buggy: many video drivers choke on bitmaps larger than 64K; I have seen this problem with Trident, Paradise and Diamond drivers, but there are probably many more buggy ones; Trident adapters produce random GP faults if the bitmap is larger than 32K; other adapters get confused and display the first 64K bytes repeatedly. The current version of the program circumvents the common bugs, so it should work properly on all systems. I have gone to great pains to ensure that the program is very reasonable on the demands to the video driver; if you still encounter any display problems, I would humbly suggest that the video driver is hopeless and you better complain to the manufacturer.

The program's appetite for memory has been dramatically reduced since its first release (2.0); virtually no swapping should be necessary even when creating 300dpi bitmaps on a machine with 4M of RAM. I have managed to go up to 228dpi on a machine with only 2M of RAM on Standard mode (ie., with no virtual memory). If however you want to use the driver at 600dpi (or above), be prepared to have at least 6M of RAM or lots of time for swapping; the problem is that a 600dpi full page bitmap takes over 4M of RAM and there is no obvious way to decrease this requirement.

You can run multiple instances of the program with no ill effects; the only thing that you should keep in mind is that the program saves its configuration upon exiting. Therefore, if you run two instances of the program, the configuration of the last one will be retained for the future.

The 32-bit version of the program can also run under 16-bit Windows by using the Win32s extender by Microsoft (which is available at ftp.cica.indiana.edu under the name win32s.zip in the directory pub/pc/win3/util). Keep in mind however that the 32-bit version of the program has about the same speed as the 16-bit version (under 16-bit Windows), and it requires more disk space because of the Win32s extender. The 32-bit version is faster under NT because it is a native application and bears no performance penalty due to emulation. Therefore, I would suggest to use the 16-bit version under regular Windows and the 32-bit version under NT. The only valid reason for using the 32-bit version under regular Windows is to exploit its higher capacity, but

most limits are sufficiently high even for the 16-bit version.

The driver has proven very useful to me, and I hope that it serves you equally well. I have tested and debugged the program extensively, but I cannot afford to make any guarantees; anybody who uses it assumes all risks. On the other hand, if you find any bug or have any suggestion for improvements, I will be more than happy to hear about it and I will do my best to fix it. You can contact me via e-mail at "isendo@leon.nrcps@ariadne-t.gr" or regular mail at "3 Sifnou St., Athens 11254, Greece".

Packing List

Make sure that you have all the relevant files:

Filename	Description
dviwin.exe	DVI driver
dviwin.str	String file for dviwin.exe (English version)
dviwin.hlp	Help file for dviwin.exe
dviwin.wri	Printable documentation for dviwin.exe
wbr.exe	Text file browser
wbr.str	String file for wbr.exe (English version)
wbr.hlp	Help file for wbr.exe
wbr.wri	Printable documentation for wbr.exe
clipmeta.exe	Utility for exporting metafiles from the clipboard
clipmeta.str	String file for clipmeta.exe (English version)
clipmeta.wri	Printable documentaton for clipmeta.exe
miscwin.dll	Utility routines shared by all executables
graphio.dll	Utility routines for importing/exporting graphics files
pcxin.flt	Graphics filter for PCX files
bmpin.flt	Graphics filter for BMP, DIB and RLE files
mspin.flt	Graphics filter for MSP files
gifin.flt	Graphics filter for GIF files
xpmin.flt	Graphics filter for XPM files
grview.exe	Demonstration program for using the GraphIO library
ctl3dv2.dll	3D dialog routines
demo.dvi	Demonstration DVI file
demo.tex	Source for demo.dvi
demo.wmf	Graphics file used by demo.dvi
dviwin2.hlp	Help file for dviwin.exe with large fonts
wbr2.hlp	Help file for wbr.exe with large fonts
helpme.wri	Answers to common questions and problems
dviwin.sub	Sample font substitution file
genpk.pif	Sample file for generating missing fonts automatically
genpk.bat	Sample file for generating missing fonts automatically
genpk2.cmd	Same as above for OS/2
genall.pif	Sample file for generating all missing fonts in one pass
genall.bat	Same as above
genall2.cmd	Same as above for OS/2
dde2exe.exe	Utility program used for the forward search
whats.new	List of changes between releases
tpicspec.tex	Specification of tpic specials (borrowed from Transcript)
emspec.txt	Specification of emTeX specials (borrowed from dvidrv.doc)
graphio.zip	Sources to graphio.dll and the graphics filters. When

	you unzip it, you *must* use the "-d" option of pkunzip.
dviwin32.zip	32-bit version of all executables. It contains the files dviwin2.exe, wbr2.exe, clipmet2.exe, grview2.exe, miscwin2.dll, graphio2.dll, bmpin2.flt, pcxin2.flt, mspin2.flt, gifin2.flt, xpmin2.flt, dde2exe2.exe and ctl3d32.dll.
strings.zip	This file contains various versions of the string files for different languages.

The only required files for the DVI driver are "dviwin.exe", "dviwin.str", "miscwin.dll" and "graphio.dll". You can avoid the browser entirely by using another viewer or editor. If you intend to use graphics in your documents, I would strongly recommend the utility "clipmeta.exe" and the graphics filters "pcxin.flt", "bmpin.flt", "mspin.flt", "gifin.flt" and "xpmin.flt". The files "ctl3dv2.dll" and "ctl3d32.dll" are distributed by Microsoft which requires them to reside on your windows system directory (typically c:\windows\system). All other files (except for the printable documentation) must reside either in a directory specified by the "PATH" environment variable, or the base directory of Windows. Of course, you will also need the font files as described earlier in this document. If you use a high resolution adapter (1024x768 or higher) you may find the standard fonts in the help files a bit hard to read; the files dviwin2.hlp and wbr2.hlp are identical to dviwin.hlp and wbr.hlp respectively, except that they use larger fonts for easier viewing. To use them, just rename them to dviwin.hlp and wbr.hlp.

Acknowledgements

This program would not be in its current form without the help of the kind users around the world who identified many missing features and incompatibilities with various hardware/software. I am indebted to all their suggestions and I would like to especially thank Leonid Pryadko, Mike Reid, Darrel Hankerson and Young Ryu for their numerous ideas, productive discussions and extensive testing. The Russian, Spanish and German translations of the string files have been kindly contributed by Leonid Pryadko, Juan M. Aguirregabiria and Volker Pohlers respectively. The BMP, PCX and MSP graphics filters have been adapted from the code by Maurice Castro and Rusell Lang in dvips. The GIF graphic filters has been adapted from the code by Jonathan O' Neal in giftest.c, which in turn is based on code by Steven A. Bennett. Of course, I remain fully responsible for any remaining bugs or omissions.

Licensing Agreement