TEX DVI Driver Family Status

Nelson H.F. Beebe
Center for Scientific Computing
Department of Mathematics
220 South Physics Building
University of Utah
Salt Lake City, UT 84112
USA
Tel: (801) 581-5254

EMAIL: Beebe@Science.Utah.Edu (Internet)

Second Edition, 01 July 1989

1 Introduction

This document summarizes the status of the TEX DVI driver family distribution. It describes

- the current versions of documentation and software;
- what operating systems, compilers, and output devices are supported;
- how to obtain the distribution from Utah and several other sources;
- the DVI electronic newsletter, and
- work in progress.

Since the informal announcement of this software at the 1986 TEX Users Group meeting at Tufts University and the formal announcement in the April 1987 TUGboat, demand has exploded; the author estimates that over 1000 sites in 28 countries are now running this software. Our initial attempts at providing the software for free, or in return for a donation, proved economically infeasible (we ran seriously in the red), and consequently, a fixed distribution charge was instituted in the spring of 1988. The burden of exact cost accounting for each order would be too great to manage; a fixed fee results in some being subsidized at the expense of others, but not excessively so.

The distribution tapes contain not only the driver family, but also an extensive collection of Computer Modern fonts (itself requiring over 100 hr of DEC-20/60 time to generate), additional TEX-related support software, and for VAX VMS, the latest GNU Emacs release, so it remains a bargain.

Since the driver family is in the public domain, those who obtain distributions may freely re-distribute what they get. Local user groups are encouraged to share the software; we are doing this on a national scale in Poland.

In the interests of the end users, however, it must be recognized that this is an active software development project, and periodically returning to the source at Utah, or one of the official redistribution channels, should be rewarding.

2 Documentation

Documentation for the TEX DVI driver family was last updated 15 April 1987 for Revision 2.07; it is now slightly out-of-date, and will be updated for release 3.0. It consists of

- a large manual entitled *A TEX DVI Driver Family* contained in source files `dvidriver.*` intended for people who are installing the family on a new operating system, or adding support for a new device; it need not be read by people who just use the drivers;
- a UNIX manual page file, `dvi.1l`;
- a VAX VMS help file, `dvi.vms`;
- a GNU and TOPS-20 Emacs info file, `dviman2`;
- a GNU Emacs TEXinfo file, `dviman2.texinfo`, used to produce the previous info file; and
- a LATEX document, `dviman.ltx`, that prints in a similar style to a UNIX manual page.

The last five contain essentially the same information, but in different formats. They are intended to provide end-user documentation, with details of what drivers are available, how to run them, and what command-line options are available.

3 DVI Driver Versions

The version history of the family is given in Table 1. Starting with 2.10.12, an edit number will be appended to the major and minor version numbers; edit number 12 reflects 12 edits applied since 2.10(.00) was released. Major versions will appear at long intervals. Minor versions will appear every few months, and will generally add new drivers or new operating system support. Edits fix small bugs, or make small cosmetic changes that improve the user interface.

Table 1: Version history (reverse chronological order).

3.00	[??-???-89]
2.10.12	[01-Sep-88]
2.10	[01-Nov-87]
2.09	[23-Sep-87]
2.08	[15-Aug-87]
2.07	[15-Apr-87]
2.06	[1986-1987]

Note: Version 2.10 incorporated a major rewrite of dvialw and dvijep, the two most popular drivers, to make much better use of the limited memory on the laser printers, and to permit arbitrary numbers and sizes of fonts. Users running older versions are urged to update to 2.10 or later.

4 Host Operating Systems and Compilers

The operating systems supported are given in Table 2. The drivers will port to almost any other UNIX system not listed in the table with relatively little effort.

Table 2: Operating systems supported.

Operating System	Compiler(s)
Gould UNIX	cc
Sun UNIX	cc, gcc
Hewlett-Packard UNIX	cc
PC DOS	Microsoft C 3, 4, 5 cl
TOPS-20	kcc-20, pcc-20
VAX UNIX	cc, gcc
VAX VMS	cc, gcc

On PC DOS, only the Microsoft C compilers have so far been found usable. Microsoft C 5.1 has been tried and found unusable because it permanently predefines the ANSI C macro `__STDC__` to be 0 instead of 1. Borland Turbo C 1.5 is definitely *not* usable for the DVI drivers because of completely erroneous floating-point code generation. Turbo C 2.0 resolves the floating-point problems, and in November, 1988, I developed work-arounds for several new bugs in the compiler and the library.

For programs that have no floating-point code, I have found Turbo C 1.5 quite usable; it compiles about 5–7 times faster than the Microsoft C 5.0 compiler. Version 2.0 remains as fast, but examination of the assembly code generated for the string primitives shows that, when optimization is selected in both compilers, the Microsoft C compiler produces substantially better object code. In the large memory model required for the DVI drivers, the inner loops for `strchr()` and `strrchr()` have 11 instructions each with Turbo C 2.0 when the `-O` option is chosen, and 5 and 7 respectively with Microsoft C 5.0 with `-Oailt`.

gcc is the GNU Project C compiler, which generates code for at least these architectures:

- Alliant FX/8,
- Altos 3068,
- Convex C1 and C2,
- Intel 386 (Compaq 386, Sequent 386, Sun 386i),
- MIPS (MIPS and DECstation 3100),
- Motorola 68xxx (Sun 2 and 3, HP 9000/300 and 320, AT&T 3B1, Integrated Solutions, Sony NEWS, NeXT),
- Motorola 88000 (in progress),
- National Semiconductor 32xxx (Sequent, Encore, NS Genix),
- Sun SPARC (Sun 4 and Solbourne), and
- VAX (UNIX and VMS).

That is a remarkable record surpassed only by pcc, but that was never available for diverse systems from one set of master sources. gcc is also *free*. On the Sun 3, it produces code about 10% more compact, and 10% faster than the native Sun cc compiler at Sun OS 3.x; Sun cc produces better floating-point code than gcc at Sun OS 4.0. gcc is the standard C compiler on the NeXT workstation.

Due to code generation errors, neither Sun cc nor gcc compile the drivers correctly on the Sun 386i under UNIX; under 386i DOS, the Microsoft and Borland compilers produce working drivers.

Version 3.0 is expected to add support for the operating systems in Table 3. These are not in the 2.10 distribution, but once 3.0 development has stabilized, pre-releases from the development directories may be made available by special arrangement on tape or for ANONYMOUS FTP; IBM PC floppy distributions will not be offered. Contact the author for details and current status.

Table 3: New operating systems supported.

Vendor	Operating System
Acorn	Archimedes
IBM	VM/CMS
Intel	RMX
Prime	Primos

5 Output Devices Supported

Devices supported by the TEX DVI driver family at version 2.10 are given in Table 4. The dvitype program is not strictly part of the distribution; it should be a standard part of every TEX distribution. Similarly, there are programs dvitty and dvidoc which attempt to display DVI files on ASCII printers and terminals; they are included in the tape and ANONYMOUS FTP distributions, but are not members of the family.

Table 4: Supported output devices.

Program	Output Device
dvialw	PostScript (Apple LaserWriter)
dvibit	Version 3.10 BBN BitGraph terminal
dvica2	Canon LBP-8 A2 laser printer (fast experimental version)
dvican	Canon LBP-8 A2 laser printer
dvidla	DEC LA50 144 × 72dpi and LA75 144 dpi printers
dvieps	Epson 9-pin family 60/72 and 240/216 dpi matrix printer
dviimp	Imagen imPRESS-language laser printer family
dvijep	Hewlett-Packard LaserJet Plus
dvijet	Hewlett-Packard LaserJet
dvil3p	DEC LN03 Plus laser printer
dvimac	Apple Imagewriter 72 and 144 dpi printer
dvimpi	MPI Sprinter 72 dpi printer
dvioki	OKIDATA Pacemark 2410 72 and 144 dpi printer
dviprx	Printronix 60h × 72v dpi printer
dvitos	Toshiba P-1351 180 dpi printer
dvityp or dvitype	DVI Translator for human-readable output

New devices for which support should be available in 3.0 are given in Table 5.

Table 5: New device drivers scheduled for future release. Names are subject to change before final release.

Program	Output Device
dviadx	Anadex Silent Scribe 72 and 144 dpi dot matrix printer
dviapo	Apollo screen display previewer
dvicon	IBM RT 6150 and 6155 console previewer
dvidsk	Hewlett-Packard DeskJet 300 dpi ink jet printer
dvielq	Epson LQ 24-pin family 180 dpi dot matrix printer
dviep2	Epson 9-pin family 120/216 dpi dot matrix printer
dvifuj	Fujitsu DL2400 24-pin 180 dpi dot matrix printer
dvigp	Phillips GP 72 dpi and 144 dpi dot matrix printer
dvihl8	Brother HL-8 with brain-damaged Hewlett-Packard LaserJet emulation
dviibm	IBM 4202 120 dpi dot matrix printer
dvilzr	Data Products laser printers (1230 and 1260) with brain-damaged Hewlett-Packard LaserJet emulation
dvio92	OKIDATA 192 72 and 144 dpi dot matrix printer
dvisun	Sun Windows screen display previewer
dviupc	AT&T UNIX PC screen display previewer
dvivga	IBM PC VGA display previewer

In addition, the TOPS-20 PostScript spooler program, `lw78.c`, has now been ported to UNIX.

6 DVI Driver Distribution

Distributions of the DVI driver family are available from a number of sources; they are described in this section.

The master files reside on the author's main host machine, science.utah.-edu, a DEC-20/60 TOPS-20 system. ANONYMOUS FTP (any password) to that machine can retrieve the file `00readme.txt` which tells how to find the DVI distribution, and much else, on local machines. Distribution formats of individual files, compressed UNIX tar files, and IBM PC `.arc` files are available on science.utah.edu. It is important to read the `00readme.txt` file carefully and follow its instructions about what files to retrieve, and how; otherwise you risk getting corrupted, or incomplete, data.

VAX VMS backup save sets of the DVI family, Computer Modern fonts, a PostScript printer spooler, and several TeX-related programs are available on ctrsci.utah.edu, a VAX 8600 running VMS 4.4, where again a `00readme.txt` file gives retrieval details. The ANONYMOUS FTP password is *GUEST*; no other string will be accepted.

Distribution from Utah must be requested in writing, by postal mail to

TeX DVI Driver Distribution
Center for Scientific Computing
Department of Mathematics
220 South Physics Building
University of Utah
Salt Lake City, UT 84112
USA

or by electronic mail to dvi-request@science.utah.edu or the author, or by FAX to the telephone number (801) 581-4801. A telephone number and street address should be provided to facilitate resolution of questions, and delivery by express freight companies (who cannot deliver to postal boxes). Telephone orders will be accepted only in unusual circumstances.

Nine-track tape (1600 bpi, and for VMS only, 6250 bpi), Sun 1/4*in* cartridge tape, and IBM PC floppy distributions are available from Utah for a fixed US$100 charge, which includes documentation, media, and shipping. Shipping is by UPS ground service within the lower 48 states, or airmail to Alaska, Hawaii, and international destinations. Do *not* send tapes or floppies with your request. Prepayment in checks drawn on a US bank, or international postal money orders, is preferred because it reduces paperwork. Purchase orders will be accepted when prepayment is not possible; an invoice will accompany the shipment.

Faster shipment by UPS, DHL, Federal Express or Airborne Express is possible by special arrangement, *provided our local staff has the time to handle the order*. It carries a surcharge equal to the approximate freight charges; prepaid or collect shipments should be used.

We try to fill orders within one to two weeks of receipt, but it sometimes takes longer because of our local responsibilities, or because an order is missing information, like tape format and density, that we need before it can be completed.

Distributions are available from the following other channels; allow up to 2 months after release of new versions for these to be available. All of these are volunteers, and have full-time commitments elsewhere. They are updated by net distribution where possible, but it still takes them some effort for them to incorporate the changes in their local distribution mechanism.

Peter Abbott
Aston University
Computing Services
Aston Triangle
Birmingham B47 ET

England
Email: AbbottP%uk.ac.aston.mail%uk.ac.rl.gb@nss.cs.ucl.ac.uk
or
Email: PAbbott@nss.cs.ucl.ac.uk
[only net retrieval]

Massimo Calvani
Department of Astronomy
University of Padova
Vicolo dell'Osservatorio
35122 Padova
Italy
Email: Calvani%vaxfpd.infnet%iboinfn.bitnet@cunyvm.cuny.edu
[only net retrieval]

Lance Carnes
Personal TEX, Inc.
12 Madrona Ave
Mill Valley, CA 94941
Email: "well!pti"@lll-lcc.arpa
[only IBM PC floppies]

Edgar M. Cooke
Software Research Assoc. Inc.
1-1-1 Hirakawa-cho
Chiyoda-ku
Tokyo 102
Japan
Email: kddlab!srava.sra.junet!cooke@uunet.uu.net
[only net retrieval]

Richard J. Kinch
Kinch Computer Co.
501 S Meadow St
Ithaca, NY 14850
Email: -unknown-
[only IBM PC floppies]

Mark Kosten
LaTrobe University

Bundoora, Victoria
Australia
Email: "munnari!latvax8.lat.oz.au!ccmk"@uunet.uu.net
[net retrieval and tape distribution]

Joachim Lammarsch
Universität Heidelberg
Rechenzentrum - Im Neuenheimer Feld 293
Heidelberg 6900
West Germany
Email: $rz92%dhdurz1.bitnet@cunyvm.cuny.edu
[only net retrieval]

Jon Radel
P. O. Box 2276
Reston, VA 22090
U.S.A.
Email: jonradel%icecream.princeton.edu@princeton.edu
[only IBM PC floppies]

We would prefer that IBM PC floppy distributions be handled through Personal TEX or Jon Radel; their prices are also lower. Preparation of floppies takes substantial personal time for the author which would be better spent on other activities.

7 Electronic Newsletter

A network mailing list is maintained for the issuance of newsletters; there were 285 subscribers on 29 May 1989. Requests for addition or deletion should be sent to the author at the address on the first page of this document. Users on any network reachable from the Internet can be included; that includes at least Arpanet, CSnet, MILnet, NSFnet, and SPAN (mainly US), Bitnet (US, Canada, and Europe), NetNorth (Canada), EARNnet (Europe), JUNET (Japan), Janet (Britain), Usenet (worldwide), and national university nets in Australia and New Zealand. A total of 19 newsletter issues have so far appeared. Back issues are included in all DVI distributions as the files `00mail.*`.

Regrettably, local staffing and funding do not permit postal mailings of the newsletter; if there is a demand for it, and subscribers are willing to pay for it, it could be arranged in the future.

There is a smaller related electronic list for beta testing of a powerful LATEX editing mode in GNU Emacs; this will probably become part of the standard

GNU Emacs distribution in 1989. It contains many useful functions, and recognizes *every* macro in the LaTeX *User's Guide and Reference Manual.* Send requests for addition to the list to the author at the same address as for the newsletter. The code is also included in the VAX VMS distribution tapes in the file `[.emacs-18-52.lisp]latex-mode.el`.

8 Future Directions

NB: Anything discussed in this section is subject to possibly substantial changes.

Source release of the 3.0 development is still many months off in the future; changes are still frequent, and a premature release would open a horrendous bag of worms that would make management of the development beyond my limited resources. Every time I have attempted to predict when a piece of software would be complete, I have been wrong, so I shall not attempt to set any date for the version 3.0 release.

Here is a summary of new features in the 3.0 development:

- For devices which require page bitmaps, the bitmaps are constructed in strips with multiple passes through the commands for the current page in the DVI file. The amount of memory to be used for a bitmap strip is set to a default value, but the value may be changed by a command-line option. This removes the need to limit the printable page size that existed in earlier versions on small machines, particularly the IBM PC.

- File search paths are supported for all supported operating systems. A separate search path, `DVIINPUTS`, is provided for startup files, PostScript header files, and files required by `\special{}` commands. It defaults to the same value as `TEXINPUTS`, so normally, it need not be specified.

- Font file names are constructed from a format that may be set at run-time in the `FONTFMT` environment variable, and that variable can specify several name formats. This generalizes, and replaces, the `FONTLIST` variable used in earlier versions.

- A new font file encoding, Group 4 FAX, implemented by Michael Ferguson and colleagues at INRS Telecommunications in Montréal, has been added. This is even more compact than `PK` format, and will be referred to as `FX4` format. A utility for converting from `GF` or `PK` formats to `FX4` format is provided.

- `\special{}` strings of arbitrary length are supported. Although the TeX input buffer size typically limits input strings to about 500 characters, macro-generated strings can be larger.

- NUL characters in \special{} strings are supported. In C, such characters terminate strings by default, so extra processing is necessary to avoid such truncations.
- Support for Wizard C on PC DOS has been replaced by support for its descendant, Borland Turbo C 2.0 or later.
- On most systems, the operating system and compiler can now be determined automatically, eliminating the need to edit machdefs.h. This is supported by a new header file, os.h.
- ANSI C is the standard programming language. Header files stdlib.h, stddef.h, and unixlib.h, and several string utilities str*.c, support the illusion of an ANSI environment on pre-ANSI C systems.
- Dependency lists in makefiles are now generated automatically by a UNIX script, FIXMAKEFILES, which uses a script, fixmf, plus nawk (using files include*.awk), and sed. This greatly facilitates makefile updating when the source files are often changing, and ensures that dependencies are always correct.
- Handling of font magnifications has been revised by elimination of the floating-point mag_table[] array from gblvars.h, and its replacement by an integer table, stdmag[], in stdmag.h. That file is automatically generated by genmag.c. The contents of stdmag[] can be replaced at compile-time, or at run-time, by definitions of the environment variable FONTMAGS. Revised code in openfont.h will now find the closest matching magnification when font substitution is called for; previously, the mag_table[] mechanism required an exact match, making it awkward to support magnification families that were not already represented in the mag_table[] array.
- Environment variable handling and naming has been standardized across all operating systems, and the names are now set in the preprocessor macros, ENV_xxx, in machdefs.h.
- FASTZERO and zerom() have been eliminated in favor of an implementation of the ANSI library routine, memset(), where required.
- Paper dimensions no longer are fixed in the dvixxx.c files. Instead, a general, programmable, and easily-extendable, mechanism for specification of paper characteristics has been implemented in paper.c. A list of all the standard paper sizes I could find is present in paper.dat, and the more popular ones have been extracted and stored as initializing values for stdform[] in gblvars.h. gensiz.c can be used to generate much of paper.dat.

- A startup file, `dvi.ini`, is read in `initglob.h` and options found there are parsed by `fileargs()` in `filarg.c` and then set by calls to `option()`. After processing that file, driver `dvixxx` then does the same thing with `dvixxx.ini`. Both files are looked for in the `DVIINPUTS` search path; neither file need exist. This makes it simple to have private values of options that are used regularly, without having to specify them on the command line. It also makes it possible to get around operating system limitations on command line arguments, and environment variable syntax. The reason for having two initialization files is that some things, like paper types, can be defined for all drivers in one file, `dvi.ini`. The `dvixxx.ini` file can then provide modifications specific to one output device.

- Private and library functions that are `void` are no longer typecast as `(void)` when called; some compilers (e.g. Lattice C on IBM PC and VM/CMS, and Prime compilers) erroneously flag this as an error.

- `exit()` is called with the ANSI standard arguments `EXIT_SUCCESS` and `EXIT_FAILURE`, instead of 0 and 1. VAX VMS unnecessarily changed the interpretation of `exit()`'s arguments, and the ANSI committee had to support it. The private version, `EXIT()`, is no longer used in the DVI drivers.

- All `main()` functions are declared of type `int`, and return `EXIT_SUCCESS` or `EXIT_FAILURE`.

- `texidx.c` has been fixed to correctly handle out-of-core sorting.

- `qsort()` and `qsort.c` (used in `texidx`) has been renamed `shsort()` and `shsort.c`, avoiding conflicts with library functions of the same name, and more properly describing the underlying algorithm (shellsort, instead of quicksort). Shellsort is stable, whereas quicksort is not, and a stable sort is required for correct ordering of index subfields.

- Memory allocation and freeing is now handled in the DVI driver code by `xalloc.c`, which contains functions `xmalloc()`, `xcalloc()`, `xmemused()`, `xrealloc()`, and `xfree()`. These provide work-arounds for some deficiencies in existing `malloc()`/`free()` implementations, additionally keep a record of memory utilitization, and provide for debug tracing of memory usage. The `x*alloc()` functions do not return when memory is exhausted, removing the necessity for checking function return values in their callers; in the rare event that recovery is possible in such a case, `malloc()` can still be called.

- A new preprocessor symbol, `DVI`, is defined at compile time; it is used in a few routines that are separately compiled, and need to know whether they are being used in the DVI drivers, or in other code.

- A new preprocessor symbol, TEST, is used in some files to bracket built-in test code with a main() function. This avoids having to carry around a second file that serves as a test program. filarg.c, paper.c, and xalloc.c currently use this facility.

- When a font substitution is made, a Boolean flag, substitute, is now set in the font_entry structure. In prtpage.h, when that flag is set, setchar() is called instead of setstr(), so that single characters are set at a time, rather than strings of characters. This will improve the positioning of substituted characters.

- Ten new device drivers have been added: dviadx, dvidsk, dvielq, dviep2, dvigp, dvihl8, dviibm, dvilzr, dvisun, and dviupc. One driver, dvigd, has been deleted. The drivers dvi*72 have been dropped; they can be produced by compiling the corresponding high-resolution versions with HIRES undefined. This change reduces the amount of code to be maintained; in most cases, users will prefer the higher resolution version anyway, so that is the default.

- Documentation files have been updated, and a new file, doc/dvistatus.-ltx (whose output you are now reading), will be kept up-to-date as a record of the current status of the DVI family.

- The definition of PIXROUND() in gendef.h has been corrected; it was wrong for negative arguments, and would result in one-dot positioning errors.

- Support for the IBM VM/CMS operating system has been added, using the Waterloo C compiler. Since the character set is EBCDIC, rather than ASCII, this entailed a redesign of how character string output is handled. The output files created always contain ASCII text, so this means that all text characters, but not binary data, must go through EBCDIC to ASCII translation. This has been done in such a way as to avoid any extra overhead on a native ASCII host. The non-ASCII support also serves on Prime Primos, which uses an ASCII character set biased by 128.

- References to the preprocessor symbols FIRSTPXLCHAR and LASTPXLCHAR everywhere except in charpxl.h, gendefs.h, and readpxl.h have been replaced by FIRSTFONTCHAR and LASTFONTCHAR, which correctly reflect the range 0...255; PXL format fonts remain restricted by their design to 0...127.

- All uses of C type float have been replaced by type double. For dvialw in particular, this tends to ensure that identical coordinates are generated on different machines, at least to the precision they are output in.

- The PostScript header file, `dvialw.ps`, has been extensively rewritten to remove device resolution dependence, and operators that conflict with the Adobe Encapsulated PostScript File (EPSF) guidelines. Uses of the `bind` operator have also been removed in order to make it possible for `\special{}` code to redefine operators at print time.

I have made substantial progress in enhancing the support for `\special{}`. This work is currently being done with dvialw, but in such a way that it will be easy to generalize it to all other drivers.

The first step needed in this direction is to define a grammar for what the `\special{}` string looks like. Previously, most DVI drivers have defined this syntax completely *ad hoc*, with no thought to rigorous parsing, or to future extensions.

Most of the following description is extracted from the comments in dvialw.

The contents of the TeX `\special{}` command is expected to conform to a minilanguage in the same grammar as for paper specifications. The argument string should contain a series of assignment statements for one or more of the keywords given in Table 6.

Table 6: `\special`{} keywords.

Keyword	Value	Action
graphics	string	execute the generic graphics primitives in string (not yet defined)
include	filename	insert file contents with relative page positioning
literal	string	insert literal PostScript
options	string	not yet defined
overlay	filename	insert file contents with absolute page positioning

The order of these is not significant, except that if duplicate keywords are specified, the value of the last one is used. It is not necessary to supply a final newline in the strings or files; one will be provided automatically to ensure correct output.

Literal PostScript code from a file or a literal string is expected to be well-behaved, and preferably, should conform to Adobe's EPSF format version 1.2 or later, and to Adobe's PostScript Document Structuring Conventions, version 2.0 or later. It may contain a `showpage` (which is disabled temporarily here), but it should not contain any of the operators given in Table 7.
If it does, erroneous output is virtually certain. While these commands could be disabled like `showpage` is, Adobe's EPSF guidelines do not recommend doing so.

Table 7: Deprecated PostScript operators.

`banddevice`	`grestoreall`	`nulldevice`	`setpageparams`
`copypage`	`initclip`	`quit`	`setsccbatch`
`erasepage`	`initgraphics`	`renderbands`	`setscreen`
`exitserver`	`initmatrix`	`setdevice`	`settransfer`
`framedevice`	`note`	`setmatrix`	

The imported PostScript should write into its own dictionary if it needs to define objects. Because dictionary sizes must be specified when they are created, it is not possible to define a standard one in advance in the `SB` (`\special{}` begin) and `SE` (`\special{}` end) macros to protect from corruption of the dictionary, `TeXdict`, used by `dvialw`.

The language keyword should specify `"PS"` or `"PostScript"`; letter case does not matter. If any other non-empty value is found, the `\special{}` command is ignored, since it presumably applies to some other output device, and control returns immediately. However, if no language keyword is given, we assume PostScript, and continue.

Files specified by `include` and `overlay` keywords are searched for in the `DVIINPUTS` path.

In the common case of `include "filename"`, the *upper-left* corner of the bounding box will be placed at the current point. The PostScript file must then contain (usually near the start) a comment of the form

```
%%BoundingBox: llx lly urx ury
```

specifying the bounding box lower-left and upper-right coordinates in standard PostScript units of big points (1/72 inch). Alternatively, if the comment

```
%%BoundingBox: (atend)
```

is found in the file, the last 4096 characters of the file will be searched to find a comment of the form:

```
%%BoundingBox: llx lly urx ury
```

In the case of `overlay "filename"`, the PostScript file to be included will be mapped onto the page at precisely the coordinates it specifies, where the page origin is in the lower-left corner, y increasing upward, x increasing to the right. Any `%%BoundingBox` specification is ignored, since it is not required for positioning. This option might be used to print an overlay page. For actions that are to be done on every page, such as printing a logo, or a string like *Draft* or *Company Confidential*, it is more efficient to redefine `showpage` instead.

If the PostScript file cannot be opened, or the `\special{}` command string cannot be recognized, or for relative positioning, the bounding box cannot be determined, a warning message is issued and the `\special{}` command is ignored.

Any literal string, and the section of the PostScript file between the comment lines

```
%begin(plot)
%end(plot)
```

or the entire file, if the `%begin(plot)` comment cannot be found, is copied to the output file as

```
SB % filename
(xcp-llx) (ycp-ury) translate  % if relative positioning
...literal string...
...PostScript file contents...
SE % filename
```

The `SB` and `SE` macros revert to standard PostScript units of big points, and bracket the inserted PostScript text with `save` and `restore` commands. The `translate` command positions the (0,0) origin of the inserted PostScript such that the *upper-left* corner of the bounding box is at TEX's current point.

For literal inserted PostScript without an `include` or `overlay` command, the origin is moved to TEX's current point.

The `save` and `restore` in `SB` and `SE` ensure that the inserted PostScript code cannot change the environment existing before the `\special{}`. Should it be necessary to do so (e.g. to remember things from one `\special{}` to the next, or to redefine an operator, like `showpage`), you should just output `SE` followed by your PostScript, followed by another `SB`. The intervening PostScript will then apply to dvialw's private dictionary, `TeXdict`.

In order to support things like landscape mode, change bars, and grey shading, it is necessary to have paper dimensions, the bounding box size, and the current point available to inserted PostScript code. These are stored in the PostScript macros `PaperHeight`, `PaperWidth`, `BoxHeight`, `BoxWidth`, `CurrentX`, and `CurrentY` in the outer level dictionary, `TeXdict`. `PaperHeight` and `PaperWidth` are set only once, at the beginning of the job. The other four are redefined before each `SB` is output. Their values are all in standard PostScript units of big points, *not* pixels. For an `overlay` command, the size of the bounding box will be the page size.

The origin can be moved to the lower-left page corner by the PostScript sequence

```
CurrentX neg CurrentY neg translate
```

This is useful in order to obtain absolute page positioning, such as for a page logo overlay.

The size of the bounding box in big points is saved in PostScript macros `BoxWidth` and `BoxHeight`. They can be used to perform geometric transformations on the included PostScript. For `overlay "filename"`, they are set to the page size.

Here are some examples:

```
% Display a picture with the upper-left corner at the current point
\special{language "PostScript", include "pict.eps"}

% Display a picture at its original absolute page position
\special{language "PostScript", overlay "pict.eps"}

% Use literal PostScript  to draw a 1in box  with lower-left corner at
% TeX's current point
\special{language "PostScript",
        literal "
            newpath
                0 -72 translate % move origin from upper-left to lower-left
                0 0 moveto
                0 72 rlineto
                72 0 rlineto
                0 -72 rlineto
                -72 0 rlineto
            closepath
            4 setlinewidth
            stroke
            showpage"}

% Display a figure at half size
\special{language "PostScript",
        literal "0.5 0.5 scale",
        include "pict.eps"}

% Display the  figure in landscape  mode  by rotating  the coordinates
% about the center of the bounding box
\special{language "PostScript",
        literal "BoxWidth 2 div BoxHeight 2 div translate
                90 rotate
                BoxWidth -2 div BoxHeight -2 div translate",
        include "pict.eps"}
```

Landscape mode for a complete document is properly handled by a paper type, and that is now working for dvialw. Landscape mode for a single page is tricky, because there is the question of which way to rotate the page, and how to support portrait mode page headers with a landscape mode table. The latter cannot be accomplished on most devices because of restrictions they impose. However, it is possible with PostScript, and I have a demonstration file that shows how to do it.

dvialw has been revised to make it possible to output PostScript conforming to Adobe's EPSF specifications. To do this, each page must be independent of every other page, and for a job with downloaded fonts, this poses a very large overhead. Normally, fonts are not reloaded every page, but apart from this, the output PostScript follows the EPSF guidelines. In particular, none of the deprecated operators are used, and I have demonstrations that show that it is possible to incorporate dvialw output as input pictures that can be arbitrarily transformed. This can even be iterated to produce a figure showing a TeX page inside a TeX page inside a TeX page inside a . . . .

dvialw has also been changed to remove almost all dependence on output device resolution. Now, only a single parameter sets the resolution in dots/inch. This should make it easy to extend to support PostScript printers of other resolutions, including the NeXT 400-dpi laser printer, and phototypesetters.

With the assistance of in-line PostScript code, change bars proved to be easy to implement; I thought about how to do it one evening, wrote the code the next morning, and had it working almost immediately. The essential idea is to redefine the PostScript `showpage` operator to handle completion of an open change bar at end-of-page, and provide for its continuation on the following page. From the user's point of view, change bars merely involve bracketing the changed text with `\BeginChangeBar` and `\EndChangeBar` macros; no restrictions whatever are placed on the intervening text.

Grey shading proved harder to do, but the problem has now been solved. Because PostScript has an opaque painting model, it is impossible to draw the shading after text has been set, since the shading would replace the text. Drawing the shading first would be more convenient, since it could permit shading to be handled like change bars, and flow automatically across page boundaries. Instead, it is necessary in the TeX code to place the text to be shaded in a box, then to calculate the box dimensions, output PostScript code to draw a grey box slightly larger than that, then output the box itself. The tricky part is to get both boxes correctly aligned.

I have written PostScript macros that make it easy to take an input PostScript file and arbitrarily scale it to fit a rectangle of specified size. Here is an example that scales a figure to fit a half-size page:

```
\special{language = "PS",
        literal = "PageWidth 2 div PageHeight 2 div RESIZE",
        include = "spec8a.eps"}
```

The use of the predefined `PageWidth` and `PageHeight` values ensures that this will work correctly for any paper size.

`dvialw.ps` also contains a macro `sx sy SCALE` to scale an input figure by factors `sx` and `sy` from its natural size.

At INRS Telecommunications in Montréal, Michael Ferguson has implemented TeX macros that actually read the PostScript file to find the dimensions from the `%%BoundingBox` comment, and then automatically compute the space needed for the figure, freeing the user of having to insert explicit TeX spacing requests around a `\special{}`. Those macros should be easily adaptable to this new `\special{}` support code in dvialw.

In order to deal with the fact that `\special{}` code is not standardized, and each driver implementer has done it differently, it may be desirable to provide a run-time regular expression editing capability for `\special{}` strings. This would add the not-insignificant code for regular expression parsing, which may require excessive memory. It would, however, make it fairly straightforward to provide built-in (and user-definable) support for syntaxes provided by other DVI drivers. No code to do this has yet been implemented, and it would probably be better instead to write a separate DVI-to-DVI filter to do the job. While such changes can obviously also be done in the original TeX manuscript, DVI files have become a medium of documentation exchange on the Internet, and will often be processed on a system other than the one they were generated on.

Most of the preceding remarks seem to be dependent upon PostScript, and indeed, features like change bar support seem to need the programmability of PostScript. What about ordinary graphics files? In the TUG DVI committee, we have been wrestling with this problem for many months. In May, 1989, I gave a talk in Paris on the topic of TeX and graphics, where I discussed various possibilities. It is published in *Cahiers GUTenberg*, No. 2, Mai 1989, pp. 13–53.

While simple pictures can be done (usually laboriously) with TeX macros, TeX memory limits put a very severe restriction on the complexity of pictures that can be handled, and TeX is almost completely lacking in graphics primitives. Thus, handling of complex pictures seems destined to be relegated to requests in a `\special{}` string.

The problem is then to decide on a format that can be generated by graphics packages, produced by translation from popular graphics file formats (e.g. Tektronix and HPGL), and written by hand for simple cases.

After long thought, my inclination is not to invent something new here, but instead to adopt a subset of PostScript. A subset is required to simplify the parser. PostScript is a stack-based extensible language, and parsing cannot be done correctly until *every* operator can be interpreted; some operators take variable numbers of arguments, and the stacks cannot be managed without knowing how many arguments an operator must consume. I have therefore modified my <PLOT79> graphics system to produce such a subset, and have produced a working parser for that subset in about 220 lines of code and 130 lines of comments. With planned extensions to the subset, it seems likely that an adequate parser may be around 1000 lines of code, which represents about 10% of the current size of a typical DVI driver.

Although PostScript has a large vocabulary, a subset of 20 operators is sufficient for most line graphing applications, and it should be possible to write the parser in such a way that a vector of pointers to functions can be provided by the caller, so that the same parsing code can be used by all the DVI drivers, with separate implementations of the primitives for different drivers. However, all of the bitmap device drivers could use essentially identical code, since in each case, the primitives merely have to be expanded into dots in a bitmap of known resolution.