Output Devices

The DVI Driver Standard, Level 0

The TUG DVI Driver Standards Committee

Abstract

The TUG DVI Driver Standard defines functional and interface requirements for computer programs (DVI processors) that read and translate files in the DVI page description language. This document is the subset of the DVI standard (level 0) applying to minimally functional DVI processors. The specifications here should be considered minimal; developers are encouraged to write drivers exceeding these specifications.

(The version of the Level 0 Standard presented here is draft 0.05. It has been reviewed by the TUG DVI Driver Standards Committee and is now being presented to the TUG membership at large for review.)

The complete standard will be presented as a series of tiers requiring increasingly stringent control over the output of DVI processors.

Notes from the Secretary

Over the last year and a half the TUG DVI *Driver Standards Committee has been developing this document, the level-0 standard draft, which is presented now to the membership-at-large. The Committee plans to transform this draft into an official TUG standard as quickly as possible.*

If you have any comments on this draft, you may address them within two months after publication to the Secretary. Please note that it is intentional that this draft does not define inclusion of graphics, \special *syntax, page selection, etc. Refer to the section below and to the report published in TUGboat 12, no. 2 (May 1991) for the reasons.*

Owing to their length we have omitted the appendices from this publication. They contain the description of relevant file formats: Appendix A: DVI*; Appendix B:* GF*; Appendix C:* PK*; and Appendix D:* TFM*. These appendices are available over the Net, and may also be ordered on demand from the Secretary (the address is given at the end of this article).*

This Notes section is not part of the draft.

1 Purpose of the level-0 standard

The level-0 standard (hereafter called the *standard*) is meant to be a base standard to which all DVI-processing programs must adhere. It provides a base level of support for both DVI-to-output-device translators (so called *drivers*) and DVI-to-DVI preprocessors (e.g., dviselect). The standard hereafter calls such DVI-processing programs "DVI *processors*" or just "*processors*". This standard allows all reasonable documents to be rendered (i.e., printed or displayed) accurately. When we refer to accurate rendering, we mean that when the data generated by the DVI processor(s) are transmitted to an output device the latter shall produce a page accurately depicting the page described by the DVI file (disregarding resolution effects and output technology).

The basis for many of the specifications in this standard is the possible output of TEX82 although some requirements are based on assumptions that cannot occur with TEX82-based output; functions which can be implemented via a pre-processor are generally omitted (e.g., page selection and sorting).

2 The DVI file

DVI processors must be able to read and *interpret* any valid DVI file as specified in appendix A. They shall also correctly *render* any DVI file which falls within all of the limits specified below. If these requirements cannot be met due to limitations of the computer or the output device they shall be fulfilled as completely as possible and the limitations documented. Aside from this exception, these specifications are a *minimum*; good processors will probably be able to handle DVI files exceeding these limits (DVI files which exceed the limits are likely to be rare, but might still occur).

Explanation: This exception is necessary because certain output devices have varying capacity depending on the amount of on-board memory or similar conditions. For example, an HP LaserJet Plus with 512 KB of memory is capable of holding in memory only 3056 distinct downloaded characters; a full page bitmap is also not possible with this configuration.

DVI commands The DVI processor must be able to interpret every DVI command listed in Appendix A.

Explanation: Some commands, e.g., *put4*, are generally used for conditions outside those enumerated below; despite this, DVI-translating programs are expected to accurately interpret these commands and execute them if they do specify an action within the specified minimum limits.

Characters

Number of characters in a font

The DVI processor must be able to handle fonts which have characters at any code c in the range $0 \leq c < 256$.

Explanation: Some printers with download possibilities require fonts with more than a certain number of characters to be broken into two or more device fonts when downloaded to the printer. Please note that this requirement alone is not sufficient to allow the exception for device limitations given in section 2 to apply.

Character size

The DVI processor must be able to render any character up to a size of 600 pt (horizontal) by 800 pt (vertical) unless this is not possible due to device constraints as outlined in section 2.

Explanation: This size is the glyph size, not the size given in the TFM files. These two sizes are not connected; i.e., the glyph might be outside the bounding box given by the dimensions of the TFM files.

Number of characters per page

The DVI processor must be able to render a page containing up to 20 000 characters unless this is not possible due to device constraints as outlined in section 2.

Unusual characters

The DVI processor must correctly render any character which meets the specifications in appendices A, C, and B (if the processor uses GF format). This includes, but is not limited to: (a) characters with empty bitmaps (e.g., the SLiTEX fonts), including characters whose horizontal escapement is 0, (b) characters whose printable image is wider than their horizontal escapement, and (c) characters with a negative horizontal escapement.

Rules

Rule size

The DVI processor must be able to render rules of any size up to 600 pt (horizontal) by 800 pt (vertical) unless this is not possible due to device constraints as outlined in section 2.

Placement of rules on the page

The lower left corner of a rule is to be placed on the page at the location given by rounding the current DVI coordinates as indicated in section 2. The height and width of the rule are given by the formula $\lceil Kn \rceil$ where n is the dimension in DVI units and K is a constant which converts from DVI units to device units.

Explanation: Devices with aspect ratios unequal to one will need to maintain separate constants for vertical and horizontal dimensions.

No rule is rendered if $n \leq 0$, as specified in appendix A.

Number of rules per page

The DVI processor must be able to render a page containing up to 1000 rules unless this is not possible due to device constraints as outlined in section 2.

Stack The DVI processor must be able to handle DVI files whose *push/pop* stack nests up to 100 levels.

Positioning on the page

Location of the origin

The DVI processor must locate the origin $(0, 0)$ at a point one inch (25.4 mm) from the top of the page and one inch (25.4 mm) from the left side of the page.

Explanation: While these default margins are inconvenient for users of non-U.S.-sized paper, the advantage of having a universally standard default location of $(0, 0)$ and the widespread assumption of these defaults in most macro packages outweighs the inconveniences. For some DVI processors (e.g., screen previewers), this specification refers to a virtual page and not the physical output.

Changes in position due to characters and rules

The definition of DVI files refers to six registers, (h, v, w, x, y, z), which hold integer values in DVI units. In practice, we also need registers *hh* and *vv*, the pixel counterparts of h and v, since it is not always true that $hh = \text{pixel\_round}(h)$ or $vv = \text{pixel\_round}(v)$ where $\text{pixel\_round}(n)$ is defined as $\text{sign}(Kn) \cdot \lfloor \text{abs}(Kn) + 0.5 \rfloor$ with $\text{sign}(i)$ resulting in -1 if $i < 0$ and in 1 otherwise.

Whenever the DVI processor encounters an instruction that changes the current position, it must update h and v. If the change in position is due to a command which sets a character, the processor adds the horizontal escapement value from the PK or GF file to *hh* to get the new value for *hh*.

For a horizontal movement of x DVI units from any other command, *hh* will be set to $hh + \text{pixel\_round}(x)$ if $x < \textit{word\_space}$ for a horizontal movement to the right or if $x > -\textit{back\_space}$ for a horizontal movement to the left. *word_space* is defined as *space* − *space_shrink*, and *back_space* is defined as 0.9 *quad* if the processor uses TFM files. If the processor does not use TFM files the design size of the current font in the DVI file (after all necessary magnifications have been applied) may be used for a *quad*, and 0.2 *quad* must be used for *word_space*. If x exceeds the bounds outlined above, *hh* is set to $\text{pixel\_round}(h + x)$. In this way, rounding errors are absorbed by interword spaces.

For a vertical movement of y DVI units, *vv* is set similarly except that *vv* is set to *vv* +

pixel_round(y) if $-0.8quad < y < 0.8quad$ and set to pixel_round($v + y$) otherwise. This allows vertical rounding errors to be absorbed in the interline spacing while still allowing fractions and super- and subscripts to be printed consistently.

After any horizontal movement, a final check is made as to whether $dist > max\_drift$ with $dist$ defined as abs(hh − pixel_round(Kh)) and max_drift defined as outlined below. If it is, then hh is set to pixel_round(Kh) + sign($dist$) · max_drift. A similar check is made with vv and v. max_drift should be set to 2 for output devices with device units smaller than or equal to 0.005 in (0.127 mm), 1 for output devices with device units greater than 0.005 in (0.127 mm) but less than or equal to 0.01 in (0.254 mm) and 0 for output devices with device units greater than 0.01 in (0.254 mm).

Explanation: This method for tracking the positions is oriented towards the typesetting of text. It does not fix positioning problems with lines consisting completely of characters of a fixed-width font, where one line consists only of characters without any movements and the next line contains movements. Other problematic areas are line graphics produced with line segment characters in fonts. These line segments may not align.

Range of movement

The DVI processor must handle movements in the DVI file up to a total of $2^{31} - 1$ DVI units in any direction from the origin.

Objects off the page

Any printable object which would lie entirely off the physical page must not be rendered; any changes to positioning must still be obeyed. Any printable object which would lie partially off the physical page must either be clipped so that the portion of the object that lies off the page is not printed or else omitted entirely, unless this is not possible due to device constraints as outlined in section 2.

Explanation: Because some output devices do unpredictable things when objects are rendered partially or completely off the edge of the page, it is up to the DVI processor to make sure that objects printed partially off the page are handled correctly.

Fonts

Font numbers

The DVI processor must be able to accept a font number k, given by a *fnt_def* command, in the range $0 \le k < 256$.

Distinct fonts

The DVI processor must be able to handle any document containing 64 or fewer distinct fonts.

Specials A "special" is the parameter to the DVI commands *xxx1*, *xxx2*, *xxx3*, and *xxx4*. This standard does not define the meaning of any special. Specials not officially defined by the DVI processor standards committee should be flagged with a warning when read from the DVI file. If any specials are encountered that are ignored by the processor, the processor must issue a warning message. These warning messages may optionally be turned off at run time.

3 Configuration

It must be possible for the installer of a DVI processor to configure such things as the location and naming scheme of fonts, default paper size, etc., without having to recompile or relink the processor.

Explanation: "etc." means "make as many things configurable as possible." This should be more detailed (hint due to Karl Berry).

4 Font files

Font formats The DVI processor must be able to read PK fonts with the location specifiable at run time. The PK format is given in appendix C. GF support is optional. The GF format is given in appendix B.

Explanation: The PK format is the preferred format for bitmap fonts because (a) it is the most compact format in the TEX world and (b) included in the PK format are pieces of information about the font (e.g., the horizontal escapement in pixels for each character) which are essential for fulfilling the typesetting requirements of section 2.

The scaling number The magnification and resolution of a font are combined into a scaling number in one of two ways:

Resolution number
: The resolution number is given by *resolution* × *magnification* where the resolution is given in dots per inch (on devices with a aspect ratio unequal to one, the horizontal resolution should be used) and a magnification of 1 indicates normal sizing. This is the preferred specification for GF and PK files.

Magnification number
: The magnification number is given by 5 × *resolution* × *magnification* where both values are as above.

Magnifications

Minimum set of magnifications

The DVI processor must be able to use fonts at least at the following magnifications of its target resolution:

1 (magstep0),
1.095 (magstep0.5),
1.2 (magstep1),
1.44 (magstep2),
1.728 (magstep3),
2.074 (magstep4),
2.488 (magstep5),
2.986 (magstep6),
3.583 (magstep7),
4.3 (magstep8), and
5.160 (magstep9).

Explanation: The term magstep$n$ stems from the TEX and METAFONT control sequences with the same name. Its meaning is 1.2^n.

DVI processor authors are encouraged to support all possible magnifications.

Margin of error

If a DVI file requests a font at a size that does not exist, but the requested size is within 0.2 % of a supported magnification with the font at that size existing, the DVI processor must use the latter font without warning.

Explanation: TEX and METAFONT compute font magnifications with different precisions. Further, calculations done by TEX and/or a DVI processor are subject to roundoff errors. The margin prescribed is sufficient for accomodating most of these errors. It is *not* intended to compensate for fonts requested at an incorrect size.

Missing fonts If a font is missing the DVI processor must continue processing and, after issuing an appropriate warning message, deal with the missing font in one of three ways:

1. Insert appropriate white space where characters of the font would appear.
2. Insert black rectangles of the size of the characters given in the TFM file for the font.
3. Print the characters from that font at a different size or from another font at the same size.

If method 1 or 2 is used and the processor is unable to determine size information for the font in question, then the processor may simply ignore any character setting command that occurs while the current font is that font.

Under no circumstances should a missing font cause a fatal error.

◇ The TUG DVI Driver Standards Committee
c/o Joachim Schrod, Secretary
Technical University of Darmstadt
Institut für Theoretische Informatik
Alexanderstr. 10
W-6100 Darmstadt
Federal Republic of Germany
schrod@iti.informatik.th-darmstadt.de

Resources

New books on TEX

Victor Eijkhout

Editor's note: [Like the rest of this issue of *TUGboat*, this review was postponed from autumn 1991; by the time you read this, a corrected reprint of the subject book should be in circulation, incorporating some of the comments that appear here as well as other amendments.]

Even though English seems to be understood by just about everyone nowadays, TEX books in other languages still serve a useful purpose. Sometimes it looks as if the whole of Germany learned LATEX from Helmut Kopka instead of from Leslie Lamport, and in France Raymond Seroul's *Le petit livre de TEX* is very popular. In both cases, the rest of the world is getting a chance to see what it's been missing. Kopka's introductory volume is being translated, and Seroul's book has just appeared, under joint authorship with its translator, Silvio Levy.

A Beginner's Book of TEX (Springer Verlag, New York, 1991, ISBN 0-387-97562-4) is more than just a translation of the earlier book[1]. Levy is described as 'translator-turned-coauthor', and the most visible difference is the incorporation of the features of TEX version 3. The result is a rather handsome volume. For one, the text is very well-written, never feeling like a translation. The worst errors that I found were the misspelling 'wierd' which appears twice; the idiom 'head over heels' is used where something

[1] In this reviewer's opinion, however, the title has suffered from the translation. The original title had more of a *je ne sais quoi.*