DVI Driver Considerations for High-Volume Printing Systems

Thomas J. Reid
Texas A&M University

ABSTRACT: High-speed, high-volume printing systems such as the Xerox 9700/9790 and 8700/8790 Laser Printing Systems present some challenges to DVI driver programs. These challenges stem from the fact that these printers, instead of simply printing master copies to be duplicated elsewhere, can be used to print *all* copies of a document. This article discusses the implications that this has on the design of DVI drivers as well as the services that the drivers should provide.

The printer as the duplicator

With high-speed, high-volume printing systems, it is possible to use the system to print all copies of a document. This is particularly useful for departments that produce documents that are updated frequently: it becomes economically feasible to print the documents in smaller runs and update them as needed. Thus, "demand printing" is possible. Demand printing means simply printing what you need when you need it.

However, for demand printing to be possible, the output from the printing system must be complete. This means that no manual paste-up can be done and manual page reordering must be kept to a minimum — paste-up and reordering become the responsibilities of TEX and the DVI driver.

Another complication imposed by these printing systems is the ability to print in "duplex" (printing on both sides of the page). With duplexing it is, of course, necessary to print pages on the proper side of the sheet and to pair the correct pages together on the same sheet. Imposition of documents into booklets can be done as can reference card printing. These both require that the driver have special reordering and rotation capabilities.

Finally, the driver should be able to support the paper size/type selection which is offered for the printing system. Selection of sheets from one of a number of trays should be allowed so that covers or divider pages can be printed on a different type or color of paper.

Paste-up considerations

To support paste-up operations, DVI drivers should be able to handle merging of graphics and mixed orientations on the same page of text. Instructions for doing this should be contained in the user's TEX input file. Since TEX does not have any direct provision for these features, they need to be implemented through a series of TEX macros and `\special` commands.

Details of `\specials` is the topic of another article. However, the capabilities which are important for high-speed printers are:

- *Graphics Merging:* The ability to merge a graphics file into the document and place it at a point on the page specified by the user. The driver should be able to resolve rotations by 90° increments by rotating the graphic itself, or through processing one of four input files for the graphic.
- *Document Rotation:* Since people sometimes want to produce a document entirely in landscape instead of portrait, the driver should be able to rotate the text for an entire document.
- *Rotating a Block of Text:* Rotating a block of text on specific pages involves having mixed orientations on the same page. A good example of this effect is shown in *TUGboat* Vol. 8, No. 2 on pages 166–170; the main text of the page is in landscape while the running title is in portrait.
- *Rotating Words or Characters:* Rotating specific words or characters is useful when using TEX to produce graphs; labels on the y-axis are normally rotated counterclockwise by 90°.
- *Simple Graphics:* Simple graphics drawing capability such as diagonal and curved lines should be offered.

Sorting pages

Although page sorting is not absolutely required, it is desirable for two reasons: it makes the job of page pairing (discussed in the next section) easier, and it eliminates the need for any manual sorting of the final copies of a document.

Page sorting is not as straightforward as it may seem at first. For a typical document, sorting can be done by simply arranging the pages in increasing order by their page number (plain TEX stores the page number in `count[0]` in the DVI file). However, prefatory pages are normally given negative page numbers. These pages need to be sorted in reverse numerical order and placed prior to the main body pages (which have positive page numbers). A question arises on how a page zero should be handled. What should be done to resolve duplicate page numbers?

In a driver that the author is developing, sorting is accomplished by maintaing a "section/subsection" number and a sorting page number.

The sorting page number is simply the absolute value of `count[0]` while the section/subsection number is set to the current section number for positive page numbers or one less than the current section number for negative page numbers. For this purpose, zero is treated as a negative number. If a duplicate page number is found, the current section number is incremented by two to account for the positive and negative subsections in each section. Reordering is then a matter of sorting by the section/subsection number and the sorting page number. While this logic has proven adequate for many applications, there are still some for which it is not appropriate.

Multi-part page numbers: Consider a multi-volume document where each volume is composed of parts with multiple chapters per part. An example of this might be documentation for the system services provided by an operating system. The macros used to format the document may place the volume number in `count[0]`, the part number in `count[1]`, the chapter number in `count[2]`, and the page number in `count[3]`. Within each volume, the parts are numbered sequentially from one. Similarly, chapters within parts are numbered from one as are pages within each chapter. Updates are made periodically to the document, but because of its length, major page renumbering is to be avoided. Thus, it is possible to have new pages inserted between existing pages giving page numbers such as 8.1; inserted pages make use of `count[4]`. Further, each volume contains some prefatory pages including a table of contents; the part and chapter numbers are set to zero for prefatory pages and the page number is negative (following the plain TEX standard).

For this application, it is necessary to sort by `count[0]` through `count[4]` with `count[0]` being the major sort key, `count[1]` being next, etc. Since this application uses some special macros for handling page numbers, the following assumptions can be made about the contents of the `count` variables:

- `count[0]`, the volume number, will always be positive.
- `count[1]`, the part number, will be zero or positive. Zero can be treated as a positive number; thus, a section/subsection sort field is not needed.
- `count[2]`, the chapter number, will be positive or zero. As with the part number, zero can be treated as a positive number avoiding the need for a section/subsection field.
- `count[3]`, the page number, will always be positive for non-zero part and chapter numbers and will always be negative otherwise. The value of zero will not occur. Negative numbers are to be sorted by their absolute value. A section/subsection field is not needed since positive and negative page numbers never occur within the same part/chapter.
- `count[4]`, the inserted page number, will be zero or positive.
- Duplicate page numbers (the same values for all five `count` variables) will not occur. Therefore, no provision for handling duplicates is needed.

Preserve order: In another application, the TEX input file is created with the pages in the desired sequence. The driver should output the pages in the same order as they exist in the DVI file.

Generalized sorting: For a generalized solution, the sorting logic needs to be able to use one or more of the ten `count` values in a user-specified order. Sorting should be permitted by the actual value in the `count` variable or by the absolute value of its contents. For each sort field, an extra section/subsection field may be needed to separate positive and negative numbers or to handle duplicate page numbers. Sorting should also be permitted using a sort field consisting of the sequential number of a page within a DVI file. This would allow the DVI sequence of pages to be preserved.

Notation for sorting: Sorting fields can be given for all of the applications presented in this section. The logic implemented in the author's driver corresponds to what might be given by the notation:

```
sort S 0,
```

where "S" means to perform a major sort on a section/subsection number and "0" means to do a secondary sort by the value of `count[0]`. (This notation is not complete since it does not tell how to handle a value of zero; the presence of "S," however, implies that positive and negative numbers should be separated.) Sorting for the multi-volume application might be given as

```
sort 0 1 2 3 4
```

while

```
sort D
```

is used to preserve the page sequence in the DVI file ("D" indicates the field for the sequential page number in the DVI file).

A standard notation is needed by which the user can specify the sort fields. The notation needs to include provisions for handling positive and negative numbers as well as page zero. Also, for some systems, sorting in descending order may be desirable. The notation should be available both as a `\special` and through a command line option to the driver. Further, guidelines need to be formulated on how the ten `count` fields should be used by applications requiring more than one `count` variable to store a page number.

Page parity and pairing

Page "pairing" and "parity" are concerns for duplex printers. "Parity" is used here to refer to which side of the sheet a page is to be printed on while "pairing" means which two pages are to be paired together on the same sheet. When using only `count[0]` for the page number, the rules are easy: odd pages on the front with pages $2n - 1$ and $2n$ (where n is an integer) placed on the same sheet. Page number zero again causes a slight problem: since it is typically used for a title page, it should normally be considered odd and placed on the front of a sheet.

Page parity. In the multi-volume application of the previous section, `count[0]`, `count[1]` and `count[2]` (the volume, part and chapter numbers) should be disregarded for page parity consideration; only the page number within each chapter is to be used. However, the need for inserted pages requires that both `count[3]` and `count[4]` be treated as the page number. Parity can be determined by adding the two values together: page 8 ($8 + 0 = 8$) is even and should be placed on the back of a sheet; page 8.1 ($8 + 1 = 9$) is odd and should be placed on the front; page 8.2 ($8 + 2 = 10$) is even, etc.

In the application where the DVI page order is to be preserved, it is assumed that the user has made provisions for page parity — the sequential page counter field "`D`" can be used to determine the page parity. The driver would assign values to the "`D`" variable beginning with one and incrementing by one. Hence, the values will be alternating odd and even numbers corresponding to alternating fronts and backs.

Notation for page parity: Using a notation similar to that used for sorting,

```
parity 0
```

becomes the default rule while

```
parity 3 4
```

is used in the multi-volume application and

```
parity D
```

is used when the user has made provisions in his or her TEX source file. A standard notation should be developed for page parity along with the sorting notation. This standard should have a provision for page zero: is it even or odd?

Page pairing. In the driver which the author has implemented, page pairing is done by selecting one or two pages from a sorted page list. The following rules are used to select the page or pages:

1. Select the next page in the list.
2. If the number is even:
   a. Put a blank page on the front of the sheet,
   b. Put the selected page on the back of the sheet; and
   c. Return to step 1 for the next sheet.
3. If the number is odd:
   a. Put the selected page on the front of the sheet;
   b. Examine (but do not select) the next page in the list.
   c. If it is even and has a page number one greater than that of the page on the front, select the page and put it on the back of the sheet then return to step 1 for the next sheet.
   d. If it is even and does not belong to the page on the front, or if it is odd, put a blank page on the back of the current sheet and return to step 1 for the next sheet.

This algorithm works fine if only one `count` variable is used for the page number. In the case of the multi-volume document, both `count[3]` and `count[4]` need to be considered for page pairing. For example, page 7.1 could be placed on the same sheet with page 7 while page 7.2 and 8 share the next sheet. Thus, step 3c needs to be generalized for multiple page-parity variables.

Imposition

Imposition can take on many different forms. In printing books, 32 consecutive pages are combined on a sheet of paper which is then folded and cut. The resulting group of pages is called a signature. Other forms are common which place 16 pages per signature. Since high-speed printing systems typically use $8^1/_2'' \times 11''$ paper, 32 or 16 page signatures are normally not used. Instead, imposition is used for booklets that are folded once to $5^1/_2'' \times 8^1/_2''$ and reference cards that are folded two or three times. Both require additional page ordering capabilities as well as the ability to rotate text.

These two forms of imposition typically result in final pages which are smaller than the normal size that TEX produces. It is the responsibility of the user to set the proper values for `\hsize`, `\vsize`, `\hoffset` and `\voffset` and to select fonts that are of an appropriate size for the smaller page. The driver, however, should handle the rotation of a page from portrait to landscape or inverse landscape since it is difficult for the user to know in advance which side of the sheet a page will print on.

Booklets. To impose a document into a booklet which is folded, the driver needs to place two pages side-by-side on the front of a sheet in landscape (rotated by 90°) and two pages side-by-side on the back in inverse landscape (rotated by 270°). When the driver performs this rotation, it needs to compound any rotation specifically requested by the user. For example, if the user asks for a table to be rotated by 90° (i.e., landscape on a portrait page), the driver should print the table in inverse portrait ($90° + 90° = 180°$) on the front of a sheet or portrait ($90° + 270° = 360° = 0°$) on the back.

It is also necessary to reorder the pages so that after the set of sheets comprising the document (that is, a "signature") is folded, the pages will be in the proper sequence.

The author has implemented this by building a list of the pages to appear in a signature. The list entries are set a pair at a time using the normal duplex page pairing logic. After the list is built and padded with blank pages to contain a multiple of four pages, entries are selected four at a time beginning from the center and working toward the two ends. The four selected page entries are printed on the front and back of a sheet.

Reference cards. Reference cards normally have three panels on each side although they can have more if larger card stock is used. As with imposition, it is necessary to print the front side in landscape and the back side in inverse landscape. Special reordering is also necessary, but it is simpler than that for booklets. The author proposes two such ordering arrangements. Both involve selecting six pages (for a three panel card or eight pages for a four panel one) from the sorted page list. The set can then be printed with pages 1, 2 and 3 on the front and 4, 5 and 6 on the back, or it can be printed with 2, 3 and 4 on the front and 5, 6 and 1 on the back.

Page selection / document updates

On proof-mode printers, page selection is useful because it allows an author to print a subset of an entire document. In a demand printing environment, page selection is also important because it may be desirable to print many copies of one section of a longer document. A special case of page selection should also be available which allows only updated pages to be printed. Updated pages may be indicated by a non-zero value or a date in one or more of the `count` variables. In the multivolume application, an update package consisting of only revised pages would be much preferable to reprinting an entire chapter, part, or volume.

However, when producing duplex print, it is generally necessary to print both pages which share a sheet even though only one of them may have been requested or updated. This is easily accomplished by producing a sorted page list containing all of the pages. Each page list entry will have a flag indicating whether or not it has been selected or updated. The rules for page pairing shown earlier can then be modified by adding a fourth step which tests the flag(s) of the selected page(s) to see if the sheet is to be printed. If either or both flags are set, the sheet is processed in the normal manner. Otherwise, the selected page or pages are discarded.

Media

The Xerox 9700/9790 and 8700/8790 provide two input trays and allow the user to select paper from one tray or the other. Thus, it is possible to select main body pages from the "main" tray while pulling covers or dividers from the "aux" tray.

Many centers offer a variety of types and weights of paper for their printing systems. At the author's site, a selection of pre-drilled and non-drilled white paper, plain paper and cover stock in several colors, bond paper, and label stock are offered. It is possible for the user to request that two different types of paper be placed in the two feed trays.

It would be advantageous for paper type codes to be placed in a TEX file (in the form of `\specials` and command line arguments). The ability to select from one of a number of trays is also desirable. The DVI driver should then translate these type codes into a form for the printer according to the operational policies of the center. This would, of course, be site dependent since operational policies are non-standard.

Another consideration for media is the ability to print paper sizes other than $8\,1/2'' \times 11''$. This requires some changes in origin placement but is generally fairly simple.

Future printing systems

Although high-speed, high-volume printing systems do not yet offer color printing capabilities, future systems likely will. Through TEX macros and `\special` commands, the user should be able to control the color of the text and/or graphics on a page.

Binding is another feature which may be offered by newer printing systems. If a printer has options for user-controlled binding, the DVI driver should be able to support them.

Any standards or guidelines which are developed concerning `\specials` and DVI drivers need to be open-ended so that capabilities of future printing systems can be incorporated into them.

Conclusion

Demand printing applications made possible by high-speed, high-volume printing systems make additional demands of the document preparation software. TEX represents a sound system for handling the typesetting aspect of the document preparation. However, DVI drivers need to take on a more prominent role in the document preparation cycle by providing the non-typesetting capabilities needed for support of these printing systems.

Since the significance of the responsibilities of the DVI driver are raised to the level of those of TEX, more attention to the DVI driver is needed. While TEX represents a standard for typesetting, no such standard currently exists for DVI drivers — each driver author is left to his or her own imagination. Efforts are presently underway to propose a standard for `\specials`; similar efforts are needed to formulate standards and/or guidelines for DVI drivers.

`\special` issues

Glenn L. Vanderburg and Thomas J. Reid
Texas A&M University

ABSTRACT: As more and more DVI translation programs appear, it is important to have a standard set of `\special` commands, and a standard format for those commands, so that attention can be focused on the documents being prepared rather than on the printer being used. This article contains a discussion of the various problems associated with specials and describes a format and set of specials which could serve as the foundation for a standard.

One of the most farsighted and useful features of the TEX language is the `\special` command. In acknowledging the incompleteness of TEX and of the DVI format, Professor Knuth gave us the mechanism by which we could do much to extend the language and adapt it to changing needs and capabilities.

In *The TEXbook* ([16] page 229), Knuth has this to say:

> Whenever you use `\special`, you are taking a chance that your output file will not be printable on all output devices, because all `\special` functions are extensions to TEX. However, the author anticipates that certain standards for common graphic operations will emerge in the TEX user community, after careful experiments have been made by different groups of people; then there will be a chance for some uniformity in the use of `\special` extensions.

The time has come to define the standards which Knuth proposed. The TEX community is maturing rapidly. DVI drivers exist for many printers, but most such programs are still being developed. Powerful page-description languages now exist, and TEX users are eager to exploit their capabilities. Many drivers have implemented some specials, and some lessons have been learned. Questions and articles in *TUGboat* and *TEXhax* have demonstrated the kind of functionality which is required. A standard for the format (and function) of special commands is important, and it will be most useful now, while so many drivers are being developed. The authors have been considering this problem for more than a year, in connection with the development of a DVI driver for the Xerox 9700 family of printers, and would like to present their ideas and suggestions. Many of the ideas in this article are due (at least in part) to Dr. Bart Childs, who has spent a great deal of time considering these issues and sharing his ideas. This is not meant to imply that he endorses the recommendations