Output Devices

Output Devices and Computers

Table I: Proof-Quality Devices

	Canon LBP-10	Diablo 630	Epson MX-80	Facit 4542	Fla. Data OSP	GE3000	HP2680	Imagen Imprint 10	Laser-grafix	Qume Sprint 5	screen prevue	Symbolics LGP-1	Varian	Versatec	Xerox Dover	Xerox 9700
Amdahl (MTS)								U. British Columbia								Univ. Michigan
Apollo						COS Info.		OCLC	ScanLaser		Yale			OCLC		COS Info
CDC Cyber														U. Köln		
DEC10								Stanford; Vanderbilt	Talaris					GA Techn; Vanderbilt		Univ. Delaware
DEC20					Math Reviews			SRI; Columbia	Talaris			Univ. Wash.	AMS	Univ. Wash.	CMU	
DG MV8000									Texas A&M							
Ethernet							Stanford	Imagen							Stanford	
HP1000			JDJ Wordware													
HP3000		TeXeT					TeXeT			TeXeT						
IBM(MVS)											GMD Bonn			U. Milan		CIT
IBM(VM)								SLAC						SLAC; Weizmann		Univ. Delaware
Prime									Texas A&M					Livermore		
Siemens BS2000	GMD Bonn										GMD Bonn					
Sun					Textset			Sun Inc.			Textset					Textset
VAX (Unix)								UC Irvine	Talaris			U. Wash.		U. Wash.	Stanford	
VAX (VMS)				INFN CNAF				Kellerman &Smith †	Texas A&M			Calma	Sci. Appl.	Kellerman &Smith †		

Notes:

* Still running TEX80

† Graphics supported

Most of the interfaces listed here are not on the standard distribution tapes. Some of them are considered proprietary. Information regarding these interfaces should be obtained directly from the sites listed.

Table II: Typesetters

	Agfa P400	Alphatype CRS	APS-5/Micro-5	Compugraphic 8400	Compugraphic 8600	Harris 7500	Linotron 202
Amdahl (MVS)*			Wash. St. U.*		Wash. St. U.*		
Apollo			COS Info.				
CDC Cyber *					RECAU*		
DEC 20		AMS	Textset				Adapt, Inc.
HP3000				Univ. Sheffield			
IBM 370 *			Info. Handling*				
IBM (VM)	IAM, U. Bonn						
Sun			Textset				
Univac 1100 *					U. Wisconsin*		
VAX (UNIX)						SARA	
VAX (VMS)			Intergraph †	K & S †			

Output device data is being maintained by Rilla Thedford. Anyone desiring more information or relaying new information can send it to her on the Arpanet:

`Rilla_Thedford%UMich-MTS@MIT`

Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue, as usual, has been prepared (all with TEX82) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Autologic APS-5 (1440 dpi): font and border samples which appear in Donald E. Knuth, A course on METAFONT Programming, p. 105 ff. and elsewhere; also, the Textset advertisement, p. 159.
- Canon CX (300 dpi): The Metafoundry advertisement, p. 160.
- HP 2688A Laser Printer (300 dpi): Lance Carnes, *"small TEX"*; HP 3000.
- QMS Lasergrafix 1200 (300 dpi): QMS and Talaris advertisements, pp. 156–158.
- Versatec (200 dpi): Carlos A. Felippa, *Feedback from TEX users at Lockheed*, p. 143; VAX 11/780 (VMS).
- Xerox Dover (384 dpi): figures of Chinese characters in John Hobby and Gu Guoan, *A Chinese Meta-Font*, p. 119.

Dvi_QMS: An Example of a Driver

Tom Rokicki
Texas A&M University

This is a description of a `.dvi` driver implemented at Texas A&M University under VAX/VMS for the QMS-1200 and QMS-800 laser printers. Hopefully others will benefit from our experiences and ideas.

The code started as `DVI_TYPE`, and was modified by Bart Childs to drive the QMS-1200 from a Data General MV8000. Norman Naugle and I took this program and modified it to work under VMS. As I did not know `WEB` or TEX at the time, and the pressure was on to get a working driver, I modified the Pascal output of `TANGLE`. This initial program slowly evolved until now it supports dynamic as well as static downloading of fonts, graphics integration, legal size and notebook size paper, and both landscape and portrait modes of operation. It has been used rather extensively here at A&M, and has performed well.

Operation of the program is straightforward. After executing TEX and creating a `.dvi` file, users type in the command

```
$ dviqms filename
```

which then creates a file with the extension `.bit`. This file contains all the necessary commands for the laser printer to print the file, including font downloading, rule commands, character positioning, font selection, raster images, and, of course, the characters themselves. This file is then submitted to the print queue, which produces the final document.

The key to the speed of `Dvi_QMS` is the font downloading. There is a set of fonts that is assumed to always be downloaded—at our installation this list consists of fifteen fonts such as roman and italics at three different sizes—and is assumed to contain the fonts most commonly used. This file can be changed at any time. This list of fonts still leaves approximately seventy kilobytes of memory in the laser printer for the EUNICE `troff` fonts and dynamic downloading of TEX fonts. If a character from one of the listed fonts is used in a document, `Dvi_QMS` will not download or bit map it; rather, the font select and character code are sent to the laser printer directly. Of course, if the laser printer is powered down or confused, these fonts need to be re-downloaded to run TEX. This takes approximately five minutes and is a small price to pay for the resultant speed.

Since it is impossible to know which fonts a user might need, `Dvi_QMS` will dynamically download characters from the additional fonts on a per job basis. To do this, `Dvi_QMS` does a pre-scan of the `.dvi` file to determine the font usage. Based on this usage and the laser printer memory required by each font, a desirability index is calculated. Fonts are then downloaded on a priority basis as the remaining memory in the laser printer allows. Any additional characters used are bit-mapped. This technique introduces a significant reduction in both the size of the output file and the execution time of `Dvi_QMS`.

A true font-caching scheme was considered, but rejected for the following reasons. What we wanted was a program that would work on any VAX/VMS system with the QMS printer, regardless of other software. Secondly, the necessary interlock files and direct queue control necessary in a caching scheme would have been more difficult on a system

where either several other packages make extensive use of the laser printer, or many users run TEX simultaneously. I feel that the techniques used in Dvi_QMS result in a more efficient scheme.

In Dvi_QMS I have added several options available through the \special primitive of TEX. Since these options are directly related to the formatting of the document, I felt they should reside in the TEX source rather than be Dvi_QMS run-time options.

The first of these extensions is the \special{landscape} option. This tells the driver that the output should be printed with the text parallel to the long side of the page. Of course, the \hsize and \vsize need to be reset to the appropriate parameters. The \special{long} option will instruct the driver to use legal sized (14-inch long) paper rather than the normal size. Again, the \vsize needs to be adjusted.

The \special{include *filespec*} command takes a file and includes it directly into the .bit file produced by Dvi_QMS. This file often consists of raster commands or vector commands output from another graphics package. Alternatively, it can be a simple company logo created by hand. Surrounding the \special with a \hbox with the appropriate dimensions in a TEX macro makes the use failsafe.

There is also a facility for including low-level commands for the printer directly in the source text. In the current implementation of Dvi_QMS, you would type \special{'*quic commands*'}. This allows you to create macros that draw vectors or circles on the page from the TEX level. There is also a continue option which will defeat the re-positioning at the end of the \special, to allow a long low-level instruction sequence to be created by several different \specials.

The program allows the starting page, total number of output pages, and number of copies to be specified as options on the calling line. This way, a single page can be output for review, or twenty copies of a document can be created without submitting twenty copies of the file into the queue.

Dvi_QMS supports both the QMS-800 and QMS-1200 laser printers. This selection will default, but can be overriden on the command line.

Dvi_QMS typically creates .bit files approximately three times as large as the .dvi file. This expansion is due to the fact that the output file is a straight ascii file, with all the positioning in decimal and raster information in hexadecimal. Any fonts dynamically downloaded are also included. For typical text, using mostly roman type in normal, magstephalf or magstep1, the running time is about half that of TEX. For files like those created by WEB, the execution is about 65% of TEX. Math adds a little more, but seldom does Dvi_QMS take more than half of the total time required to process a document.

Currently I am working on recoding the program in WEB to make it available on most other machines. A modified version of Dvi_QMS is being written that will be capable of driving most bit-mapped devices. Some GKS graphics primitives are being considered. TEX at TEXas A&M University will continue to thrive.

\special

Tom Rokicki
Texas A&M University

I am constantly deluged by requests for assistance with TEX from people just starting to use the Electrical Engineering VAX at A&M. I generally start them off with an analogy between TEX and compiled programming languages. TEX compiles the source, which is then 'linked' with the pixel files into a form understandable by the printer. The compiler will catch the programming errors, and the linker should be as unobtrusive as possible. Similarly, the .dvi driver should be very quiet, requiring as little attention from the user as possible.

It often becomes necessary, however, to use some particular features of the local output device, or to include some graphics which TEX is either incapable of or not very adept at producing. To this end, the primitive \special has been made a part of TEX. This command has the form

\special{*token list*}

where *token list* is expanded immediately. (Of course, a string of alphabetic characters is a token list.) Typically, the use is of the form

\special{*keyword arguments*}

Here, *keyword* is some sort of command, requiring optional *arguments*, that would be recognized and acted upon by the .dvi driver. As an example, the *landscape* keyword is recognized by the driver here

at A&M to print the text parallel to the long edge of the paper.

Why should the uses of `\special` concern us? After all, only the few of us who write `.dvi` drivers will be able to implement uses for `\special`. Also, most uses of `\special` do not relate directly to TEX itself, but are primarily concerned with integrating other systems (especially graphics) with TEX. Finally, `\special` was intended as an arena for experimentation—the worst we could do would be to limit the possibilities by imposing a set of rules.

The uses of `\special` are of concern for several reasons. TEX source should remain compatible system to system as much as possible for obvious reasons. `\special` opens a Pandora's box of incompatibility. In addition, `\special` allows expansion of TEX to include such things as graphics in a system-independent way, if a convention can be agreed upon. Also, `.dvi` drivers should be written to handle foreign `\special`'s in a reasonable fashion.

Additionally, implementing `\special` commands is quite easy, once it is realized that they can be used to solve a particular problem. A common example of this is the ability to include graphics in TEX that were created by a different software package. This is typically done with a `\special` command which inserts the output of the graphics package into the output created by the `.dvi` driver, insuring that the positioning of the graphics is done by TEX. For devices with the ability to change the coordinate reference, this is quite trivial on the part of the `.dvi` driver. It is definitely a lot easier than trial-and-error positioning or running the paper through the output device twice—once for the TEX and once for the graphics.

So, experiment with `\special`. Implement your own extensions or ask the local gurus about `\special`. Some basic implementation guidelines are obvious: ignore foreign `\special`'s; do not make the argument a lone file name. Some other guidelines have been suggested: only put one command or group of related commands in each `\special`; model high-level graphics extensions after an existing standard (GKS); use a Pascal-type approach to keywords—*keyword (arg1, arg2...)*. The key here is to experiment. Perhaps by the next Users Group meeting we will have enough input and ideas to attempt some sort of standardization.

Read about A&M's driver and how it uses `\special` (in this issue, page 138). Let us know about your own uses or ideas. I can be reached through CSnet (or ARPAnet) at:

`ROKICKI @ TAMU.CSNET`

or through the post:

Tomas Rokicki
Electrical Engineering
Texas A&M University
College Station, TX 77843

Thanks to Calvin Jackson of Cal Tech, Albert B. Meador of PAR Technology, Inc., and Mark Blanford of RE/SPEC for their valuable ideas and encouragement.