Accessing TEX parameter values*

RWD Nickalls †

1 Introduction

It is sometimes useful for mathsPIC[1] (Syropoulos and Nickalls 2000) to be able to access TEX parameter values. For example, mathsPIC could be made to automatically scale the height and width of a graph to the as yet unknown page size of the TEX document. Knowledge of the TEX parameters `\textheight` and `\textwidth` during processing would allow the height and width of a graph to be defined as a function of these page parameters.

One way of doing this is to include in the TEX file some code to write the required parameter values to a temporary data file, in the form of mathsPIC `var` commands or macros. Using the mathsPIC `system()` command mathsPIC can run the TEX file, and then access the parameter values by inputting the data-file.

2 Useful TEX commands

```
\settowidth{\variablename}{text}
\settoheight{\variablename}{text}
\settodepth{\variablename}{text}
```

For example, the following code places the length of the word August into the variable `\auglength`. Note that the command `\the` gives the value in points, while `\number` returns the value as an integer in scaled points (see Table 1).

```
\newlength{\auglength}
\settowidth{\auglength}{August}
Length of the word August in points is \the\auglength
Length of the word August in scaled points is \number\auglength
```

The output is as follows (note that `\the\auglength` returns the numeric points but includes the characters `pt` on the end).

```
The length of the word August in points is 31.44452pt
The length of the word August in ‘scaled’ points is 2060748
```

*This article was presented at the ukTUG meeting in Oxford on 24 November, 2001.

†Department of Anaesthesia, Nottingham University Hospitals, City Hospital Campus, Nottingham, UK. Email: dick@nickalls.org [www.nickalls.org]

[1] http://www.ctan.org/tex-archive/graphics/mathspic/perl/

TEX has a number of parameters which are classified according to type, e.g. integers, dimensions, glue, muglue, token lists. For a full list of all the TEX parameters which have values which can be accessed see Knuth (1990; page 272–275).

The token list `\jobname` is very useful as it holds the filename (but not the filename extension) which TEX is currently working on. This command can therefore be used for generating temporary files, e.g. `\jobname`.dvi, `\jobname`.toc, or even for just printing the 'correct' filename on a document.

3 Outputting data to a file

TEX requires a file-number as a handle to identify an 'open' file, and has the command `\newfile` specifically for allocating an unused file-number to a variable name. We then use the commands `\openout`, `\write`, and `\closeout` to open, write to, and close the file. For example, the following code will write `...blah blah blah ..` to the file `texfiledata.dat` using the file handle `\outfile`.

```
\newwrite\outfile
\openout\outfile=texfiledata.dat
\write\outfile{...blah blah blah....}
\closeout\outfile
```

It is important to note that the `\write` command only writes the data to the file when the `.dvi` file is created, so if these commands are in a file which may not actually create a `.dvi` file, then we may need to force output (`dvi` file creation) by including something writable like `\strut` on a line.

Alternatively, we can force TEX to write to the file immediately (i.e. without waiting until it gets to the end of file processing) by using the `\immediate` command, remembering to include it with *all* the commands `\openout`, `\write`, and `\closeout`, as follows.

```
\newwrite\outfile
\immediate\openout\outfile=texfiledata.dat
\immediate\write\outfile{...blah blah blah....}
\immediate\closeout\outfile
```

Since we are interested in accessing the values of TEX parameters, we need to explore some of the TEX commands for accessing such values. For example, the commands `\the`, `\number` and `\showthe` all reveal the numeric value, but in slightly different formats and location, as follows.

```
\the\textwidth      ---> 400.0pt       includes the characters pt
\number\textwidth   ---> 26214400      scaled points (see Table)
\showthe\textwidth  ---> only writes to the log file
```

For the purposes of mathsPIC accessing the numeric value as a variable or macro, it is most convenient to use the `\number` command (yields an integer value in the 'scaled points' used internally by TEX)[2] and to incorporate it into a mathsPIC variable or macro so it is ready to be used once the temporary file it is written to has been input by mathsPIC. For example, the following code allocates the scaled point value of `\textwidth` to the mathsPIC variable `w555`.

[2] 65536 scaled points = 1 printers point (pt).

```
\immediate\write\outfile{var w555 = \number\textwidth}
```

If the `\textwidth` was 400pt (14.058 cm), then the output of the above line would be as follows (65536 scaled points = 1 printers point pt).

```
var w555 = 26214400
```

In practice, it is useful to have this line commented to indicate which TEX parameter the value relates to, as follows.

```
var w555 = 26214400%  \textwidth (scaled points)
```

However, since what came after the `%` symbol would be ignored in this setting, we need to define the `%` as a character we can print to a file, by allocating it the catcode 12 (instead of the catcode of 14 which it normally has), and calling it by a different name (`\percentchar`), and delimiting the whole line by a curly brace (to keep it local), as follows.[3]

```
{\catcode`\%=12   \global\def\percentchar{%}}%
```

So now, the following code will also include a trailing comment indicating the TEX name of the parameter.

```
\newcommand{\comment}{\percentchar\space}
\immediate\write\outfile{var w555 = \number\textwidth
         \comment\textwidth is \the\textwidth}
```

For example, the resulting line in the output file is as follows.

```
var w555 = 29368707% \textwidth is 448.1309pt
```

Table 1: Note that for accurate working always use scaled points (sp) and convert in a single step using the following convertion factors (modified from: Beccari C, 1991)

	sp
mm	186467·98
cm	1864679·8
pt	65536
in	4736286·70

[3] From: Abrahams, Berry and Hargreaves (1990), p 292.

4 The final code

So now we can put all this code together into one chunk within the target TEX file, or more usefully, keep it as a separate file which can then just be `\input` whenever it is required. For example inputting the following file (and LATEXing it) will output a data file containing appropriate mathsPIC commands with the embedded values of `\textwidth` and `\textheight`. Note that in the following example we have defined two mathsPIC macros called `textwidthcms` and `textheightcms` which will contain the relevant values in cms (see Table 1 for conversion factors).

```
%% grabtexdata.tex
%-----------------
\scrollmode  % prevent LaTeX stopping if there are errors
%---------------------
% make a print command  macro
\newcommand{\print}[1]{\immediate\write\outfile{#1}}
%---------------------
% make a comment % command  macro
% first need to define percentchar for the write statement
% (From "TeX for the Impatient" (1990), p 292)
{\catcode`\%=12   \global\def\percentchar{%}}%
\newcommand{\comment}{\percentchar\space}
%
% make a \macro command --> %def<space>
\newcommand{\mydef}{def}
\newcommand{\macro}{\percentchar\mydef\space}
%----------------------
% create and open a new file with filename = textfiledata.dat
\newwrite\outfile
\immediate\openout\outfile=texfiledata.dat
%----------------------
%% write file header & general info
\print{\percentchar\percentchar\space file: texfiledata.dat}
\print{\percentchar\percentchar\space accessing TeX parameter values}
%---------------------
%% now get \textwidth and \textheight values from the tex file
\print{var w555 = \number\textwidth\comment\textwidth=scaled points}
\print{var w556 = \number\textwidth\comment\textwidth=\the\textwidth}
\print{var w557 = \number\textwidth/1864679.8\comment (\textwidth in cms)}
\print{\comment ============}
\print{\macro textwidthcms()\number\textwidth/1864679.8\comment}
\print{\macro textheightcms()\number\textheight/1864679.8\comment}
\print{\comment ============}
%--------------------
% close the file
\immediate\closeout\outfile
```

In practice one would simply include the following line

```
\input{grabtexdata.tex}
```

in the TEX file we want data from (say, `myfile.tex`), and then LATEX the file to generate the output data file. Alternatively we could LATEX the file from within mathsPIC using the `system` command, and then input the resulting data file as follows.

```
system(‘‘latex2e myfile.tex’’)
input(texfiledata.dat)
```

Either way, the resulting output data file `texfiledata.dat` for a file having a standard `{article}` format is as follows

```
%% texfiledata.dat
%% accessing TeX parameter values
var w555 = 22609920% \textwidth =scaled points
var w556 = 22609920% \textwidth =345.0pt
var w557 = 22609920/1864679.8% (\textwidth in cms)
% ============
%def textwidthcms()22609920/1864679.8%
%def textheightcms()39190528/1864679.8%
% ============
```

Once the above file (`texfiledata.dat`) is input into a mathsPIC file we can then use mathsPIC commands to manipulate the textwidth and textheight values, as shown in the following example mathsPIC file.

```
\documentclass[a4paper]{article}
\usepackage{mathspic}
\begin{document}
.....
system(‘‘latex2e myfile.tex’’)
inputfile(texfiledata.dat)
var w=&textwidthcms, w2= &textwidthcms/2
var h=&textheightcms, h2=&textheightcms/2
...
\end{document}
```

Processing the mathsPIC file gives the folllowing output. Notice how useful it is to have the accompanying comments.

```
%% inputfile(texfiledata.dat)
%% ... start of file <texfiledata.dat> loop [1]
%%% Iteration number: 1
%% texfiledata.dat
%% accessing TeX parameter values
%% var w555 = 22609920% \textwidth =scaled points
%% w555 = 22609920
%% var w556 = 22609920% \textwidth =345.0pt
%% w556 = 22609920
%% var w557 = 22609920/1864679.8% (\textwidth in cms)
%% w557 = 12.1253632929364
% ============
%def textwidthcms()22609920/1864679.8%
```

```
%def textheightcms()39190528/1864679.8%
% ============
%% ... end of file <texfiledata.dat> loop [1]
%% var w=22609920/1864679.8, w2= 22609920/1864679.8/2
%% w = 12.1253632929364
%% w2 = 6.0626816464682
%% var h=39190528/1864679.8, h2=39190528/1864679.8/2
%% h = 21.0172963744231
%% h2 = 10.5086481872116
```

5 References

- Abrahams PW, Berry K and Hargreaves KA (1990). *TEX for the impatient*. (Addison-Wesley).
- Knuth DE (1990). *The TEXbook*. (Addison-Wesley).
- Syropoulos A and Nickalls RWD (2000). A Perl port of the mathsPIC graphics package. *TUG*BOAT, 21, 292–297. http://www.tug.org/TUGboat/Articles/tb21-3/tb68syro.pdf.
  The mathsPIC package is at: http://www.ctan.org/tex-archive/graphics/mathspic/perl/