The ArcMacro package*

Alan Robert Clark
AlanRobertClark@gmail.com

2013/08/27

Abstract

The `ArcMacro` package is essentially a set of macro's for my (ARC) own personal use. It rather strongly appears though that most of my colleagues have found them useful, and this is thus an attempt to make them more "user friendly" (ie I am now providing docs :-) :-)

1 Introduction

These macros have evolved over many year's use of LaTeX, and have been revamped, of olde, to take advantage of the elegance of LaTeX 2$_\varepsilon$. I have tried to be as compliant as possible to LaTeX 2$_\varepsilon$'s standard way of doing things, including converting all my ancient `\def`'s into `\newcommand`'s, and sometimes `\renewcommand`'s, with an occasional `\providecommand` where I thought I would be stomping on toes. Under LaTeX, the non-use of `\def` slowed things down a lot, but not under LaTeX 2$_\varepsilon$!

As the years go by, things change dramatically. Hence more macros are added, and others obsoleted (though kept for document compatibility.) As of 2004, I use `dvipdfm` to generate pdf, `hevea` to generate HTML, INFO, and text, and `dvips` for postscript. I also use GNU `pic` for all my graphics, in conjunction with `Circuit Macros` for any circuit diagrams. My incarnations of these now produce a straight eps and png output. The Holy Grail, of course, is to have ONE SOURCE DOCUMENT for all these outputs! This can be achieved by some clever tricks, also contained in the ArcMacro packages (ArcMacro.sty and ArcMacro.hva!)

Basically, although the macros are all there only for the purpose of saving typing, the Macros do fall into 4 broad categories:

A Symbol-like typing savers, eg `\degrees` or `\curl{H}` which is simply shorthand for `\nabla\times\vec{H}`. Now also includes symbols defined in my obsolete ArcSym package.

B Environment typing savers. eg `\Bit` and `\Eit` are shorthand for the environments `\begin{itemize}` and `\end{itemize}`, for example. (I used to use them a lot, but since I now use Vim as an editor under Linux, I have my `.vimrc` defined very nicely and my `:ab`breviations work beautifully. `\It` inserts the entire `itemize` environment, complete with `smallitem` and an initial `\item` etc.....

C General macros. Includes such things as `\Month`, `\Day`, `\mathbox` etc.

D Graphics inclusion. Routines that automatically label, caption, include in LOF, centre, resize etc etc; simply by `inputeps{fn}{caption}` Caters for eps, pic, pcx, fig...

E Hyphenation patterns. LaTeX 2$_\varepsilon$ gets hyphenation perfectly right for most ordinary english words, but does struggle with some technical terms, like "impedance".

*This file has version number v6.5, last revised 2013/08/27.

2 The Macros

Since the macros are basically so simple, they will be documented in place! The margin contains the name of the macro, anything surrounded by square brackets means an optional choice etc as usual. The paragraph describing the macro usually starts with some `usage` information in `teleprinter text`, followed by the effect of that text.

2.1 Section A—Symbols

`\degree[s]` Used as `25\degrees C` to produce 25°C. The singular `\degree` is also defined for a `20\degree turn` to produce a 20° turn. Note the odd use of xspace here, as it gets degrees C wrong. Hence only supplied in the singular! Note that °is also supplied by amsmath, which we now use in preference....

```
\providecommand{\degrees}{\ensuremath{^\circ}}
\newcommand{\degree}{\degrees\xspace}
```

`\quid` Well, have **you** ever called it a peaouwndd? Used as 'e paid `\quid20`, 'e did... which produces £20

```
\newcommand{\quid}{\ensuremath{\pounds}}
```

`\Ohm[s]` I never remember to call it `$\Omega$`!!! Used as `20 + j 59\Ohms`, on a `50\Ohm` line. ie Singular also defined. Produces 20+j59Ω, on a 50Ω line. Note that the use of the `xspace` package facilitates this sort of usage, getting the comma and space thingy right in this example.

```
\newcommand{\Ohm}{\ensuremath{\Omega}\xspace}
\newcommand{\Ohms}{\Ohm}
```

`\therefore` Standard LaTeX 2ε does not have a `\therefore`, although the American Mathematical Society class does. Simply used as `a=b \therefore b=c` which produces a=b ∴ b=c Note the use of `\providecommand`, just in case AMS-TeX is in use, in which case the AMS-TeX version will be used instead.

```
\providecommand{\therefore}{\ensuremath{{.\kern-.05em\raise.7ex\hbox{.}.}\ }}
```

`\d` In an integral, the d of the dx should be set in *Roman*. This produces $\int x^2 \mathrm{d}x$ instead of $\int x^2 dx$, using `$\int x^2 \d x$`, and is also used in `\frac{\d y}{\d x}`, producing $\frac{\mathrm{d}y}{\mathrm{d}x}$. For some (undocumented) reason, `\d` previously put a period under the next letter! Since I certainly never need that, and cannot find where that might be defined, or documented, I used `renewcommand`. So far I haven't found anything it has broken :-)

```
\renewcommand{\d}{\ensuremath\mathrm{d}}
```

`\[i]iint` The standard way of getting a (rather dreadful) triple int under LaTeX 2ε is

$$\int\int\int_V a\,\mathrm{d}v$$

Again, this command is provided by AMS-TeX, but not standard LaTeX 2ε. It is used as `$$ \iiint_V a\, \d v$$` to produce

$$\iiint_V a\,\mathrm{d}v$$

The `\limits` command works with it too `$$ \iiint\limits_V a\, \d v$$` produces

$$\iiint\limits_V a\,\mathrm{d}v$$

but does not centralize the limit like $\mathcal{AMS}$-TEX does. (ie Use `\usepackage{amstex}` :-)

```
\providecommand{\iiint}{\ensuremath{\int\kern-0.6em\int\kern-0.6em\int}}
\providecommand{\iint}{\ensuremath{\int\kern-0.6em\int}}
```

`\e` I like the e of an exponential to be Roman. `$\e^x$` (e^x) not `$e^x$` (e^x)

```
\newcommand{\e}{\ensuremath\mathrm{e}}
```

`\( \)` Mathematical Brackets that need stretching, used as `\(` and `\)` instead of having to type `\left(` etc. Thus `$$\( x^2 \over y_1^3 \)$$` comes out as

$$\left(\frac{x^2}{y_1^3}\right)$$

Note that this is a redefinition of the standard LaTeX 2_ε command to begin an in-text equation. I *always* use a simple `$ eqn $` for this, as it is much easier and intuitive. (I always use `$$ display eqn $$` too :-)

```
\renewcommand{\(}{\ensuremath\left(}
\renewcommand{\)}{\ensuremath\right)}
```

`\Zin` Input impedance is used all over, `\Zin` will produce Z_{in}. Naturally, characteristic impedance also features, `\Zo` producing Z_0.

```
\newcommand{\Zin}{\ensuremath{Z_{\mathrm{in}}}\xspace}
\newcommand{\Zo}{\ensuremath{Z_0}\xspace}
```

`\implies` I can never remember which particular arrow I want. Hence `\implies` gives me that zero tangential E fields $\Rightarrow$ no volt drop.

```
\providecommand{\implies}{\ensuremath{\Rightarrow}\xspace}
```

`\[bold tilde]vec` The standard LaTeX 2_ε vectors are rather insipid. My preference is for a vector to be set in Bold type. However, when I typeset my lecture notes, I do not typically remember to convert the printed Bold to a chalked tilde. So as a "special case", I need tilde's under my vectors for "chalk and talk" exercises. It is simply used as `\boldvec` or `\tildevec` at any point in the document, after which all `\vec` commands are affected. I do not provide a method of regaining the standard vectors, because I never use them :-) Thus the three versions of `$\curl{H}$` are: $\nabla \times \vec{H}$; $\nabla \times \underset{\sim}{H}$; $\nabla \times \mathbf{H}$

```
\newcommand{\tildevec}
  {
    \def\vec##1{\raisebox{-1.ex}{$\stackrel
      {\textstyle ##1}{\scriptstyle\sim}$}}
  }
\newcommand{\boldvec}
  {
    \def\vec##1{{\bf ##1}}
  }
```

`\curl` Which provides the usual notation for curl. Thus Ohms law `$$\curl{H}=\vec{J}$$` is

$$\nabla \times \mathbf{H} = \mathbf{J}$$

```
\newcommand{\curl}[1]{\ensuremath{\nabla\times\vec{#1}}}
```

`\unitvec` A unit vector is *always* Boldface, with a hat. Hence `$2\unitvec{x}+3\unitvec{y}$` produces $2\hat{\mathbf{x}} + 3\hat{\mathbf{y}}$.

```
\newcommand{\unitvec}[1]{\ensuremath{\hat{\mathbf{#1}}}}
```

`\dB` The humble dB is very often maligned, especially in math mode. `20\dB` will produce 20 dB correctly, regardless of mode. (`$x=10\dB$` $\Rightarrow x = 10$ dB)

```
\newcommand{\dB}{\ensuremath{\thinspace\mathrm{dB}}\xspace}
```

`\symXXXX` Replaces my old ArcSym package, which used ZapfDingbats characters, sent out to a .pcx form, and coverted via `bm2font` to tfm and pk. This worked for years, but then I got a fax-modem (204×198 dpi), an LQ400 (180×180 and 360×180) and we changed to a LJ4 at work (600 dpi). Unfortunately `bm2font` produces different `.tfm` files as well as different `.pk` files, hence it is not sufficient to simply generate the `.pk`'s at the different resolutions.

With the advent of LaTeX 2ε, we can simply use the ZapfDingbats font directly, via the Virtual Font interface (This requires the `pifont` package, and (obviously `dvips`)). The rest of the symbols I have `convert`ed via ImageMagick to eps bitmaps, which are, of course, scalable!

Update on Tues Jan 15 2002: The School now has a new logo, and the wits logo was a rather cruddy bitmap. Anton Frolich created a vector eps version of the Wits logo, and I used PSTricks TeX macros to add the School add-ons to the logo. Hence both the Wits and School logo's are now full vector graphics. (ie infinitely scalable without resolution loss.

The table lists the various symbols:

`\symTel`	☎
`\symAtel`	✆
`\symEnv`	✉
`\symCut`	✂
`\symCross`	✝
`\symSig`	

```
\newcommand{\symTel}{\ding{37}}
\newcommand{\symAtel}{\ding{38}}
\newcommand{\symEnv}{\ding{41}}
\newcommand{\symCut}{\ding{33}}
\newcommand{\symCross}{\ding{62}}
\newcommand{\symSig}{\includegraphics[width=35mm]{arcsig.eps}}
```

Naturally, you should scan your own signature (No, mine isn't downloadable :-)

`\ts` Abbreviation for a mathematical thinspace, put between numbers and units, as in `10\ts dBi`, which renders as 10 dBi.

```
\newcommand{\ts}{\ensuremath\thinspace}
```

`\circledChar` Puts a circle around a Character! as in `\circledChar{A}`, which renders as Point Ⓐ.

```
\newcommand{\circledChar}[1]{%
  \raisebox{.5pt}{\textcircled{\raisebox{-.9pt}{#1}}}}
```

2.2 Section B—Environments

`centre` Blasted Americans. Can't even *speak* the language, never mind *spell* it. :-)

```
\newenvironment{centre}{\begin{center}}{\end{center}}
```

`\eqn` Since I moved to HeVeA as my LaTeXe translator, it prefers command-style rather than environment-style commands

```
\newcommand{\eqn}[1]{\begin{equation}#1\end{equation}}
```

The rest of this group are not actually environment declarations, but act like them :-) This is a general Class of macros that use a capital B for begin, capital E for end, and an abbreviated keyword. For completeness, I have included some of these that I personally don't use, but I know others do. I generally use my .vimrc, or the TEX shortcuts.

`\Beqn \Eeqn` B/E equation. Used as `\Beqn x^2=2 \Eeqn` which produces

$$x^2 = 2 \tag{1}$$

```
\newcommand{\Beqn}{\begin{equation}}
\newcommand{\Eeqn}{\end{equation}}
```

`\Beq \Eeq` Why the bloody 'ell does 'e drop the bloody n ???? Even Frigging Manuscript used an n!!! Ah Well – for compatibility etc. Consider it ClarkWare's contribution to Ubuntu, APCF style. There is, however, no usage information :-)

```
\newcommand{\Beq}{\Beqn}
\newcommand{\Eeq}{\Eeqn}
```

`\Beqna \Eeqna` B/E equation arrays. `\Beqna x=2\\y=3\Eeqna` produces

$$x = 2 \tag{2}$$
$$y = 3 \tag{3}$$

```
\newcommand{\Beqna}{\begin{eqnarray}}
\newcommand{\Eeqna}{\end{eqnarray}}
```

The list environments generally have too much whitespace associated with them. This is basically because every list item is separated by a `\par`, which is great for the Americans. They indent the start of a para, and leave no extra space between them. We, civilised as we are, use no indentation on the first line of the para, but have a `\baselineskip` separating them. It is this that stuff's up the list environments. It is particularly noticeable on one word lists. Compare

Ordinary env.

- First
- Second
- Third

Using `\Bit \Eit`

- First
- Second
- Third

`\Bit \Eit`

```
\newcommand{\smallitem}{%
  \setlength{\topsep}{0pt}
  \setlength{\parsep}{\parskip}
  \setlength{\itemsep}{-\parsep}}
\newcommand{\Bit}{\begin{itemize}\smallitem}
\newcommand{\Eit}{\end{itemize}}
```

`\Benum \Eenum` Enumerate. Used as `\Benum\item First\item Second\Eenum` to produce

1. First
2. Second

```
\newcommand{\Benum}{\begin{enumerate}\smallitem}
\newcommand{\Eenum}{\end{enumerate}}
```

`\Banum \Eanum` For Anumerate :-) Used for exam question purposes, but I also find it useful elsewhere. Note that *no provision* is made for the correct sequence to be followed when using this in a nested environment!!! `providecommand` is used so that the `exam` class by Dean Redelinghuys, can

be used with ArcMacro, without complaint. Not that *I'm* complaining, anyone is perfectly entitled to steal my code, and then inconvenience me when I'm not looking... It is used as `\Banum\item First\item Second\Eanum` to produce

a. First
b. Second

```
\providecommand\Banum{\renewcommand{\theenumi}
                  {\alph{enumi}}\begin{enumerate}\smallitem}
\providecommand\Eanum{\end{enumerate}\renewcommand{\theenumi}{\arabic{enumi}}}
```

`\Bdesc \Edesc` Description. Used as `\Bdesc \item[blue] A colour\item[red]ditto\Edesc` to produce

blue A colour
red Nog 'n kleur.

```
\def\Bdesc{\begin{description}\smallitem}
\def\Edesc{\end{description}}
```

2.3 Section C—General Macros

ie. Those that cannot be classified. Can you be classified? :-)

`\re` This is used in a business letter type of place, but useful elsewhere. Underlines, centres and bolds the text passed as a parameter. Usage:

`\re{Complaint about your complaints department on the third floor.}` produces

Re: Complaint about your complaints department on the third floor.

```
\newcommand{\re}[1]
  {
    \begin{center}
      \begin{bf}
        \underline{Re: #1}
      \end{bf}
    \end{center}
  }
```

`\Day \Month` Get the month Spelt out—Surely this should be std? Note cap M and D. Really!! Have you ever tried typing numbers like this—GroenDakkies ek is lief vir jou..... Used as `The \Day of \Month` which produces The Seventeenth of September (for today) (and *NO*, I am not going to provide a `\Year` command :-) :-)

```
\newcommand{\Month}{\ifcase\month\or
January\or February\or March\or April\or May\or June\or
July\or August\or September\or October\or November\or December\fi\xspace}
\newcommand{\Day}{\ifcase\day\or
First\or Second\or Third\or Fourth\or Fifth\or Sixth\or Seventh\or
Eighth\or Ninth\or Tenth\or Eleventh\or Twelfth\or Thirteenth\or
Fourteenth\or Fifteenth\or Sixteenth\or Seventeenth\or Eighteenth\or
Nineteenth\or Twentieth\or Twenty First\or Twenty Second\or Twenty Third\or
Twenty Fourth\or Twenty Fifth\or Twenty Sixth\or Twenty Seventh\or
Twenty Eighth\or Twenty Ninth\or Thirtieth\or Thirty First\fi\xspace}
```

`\fullref` ie. A `\ref` and a `\pageref`. This does not try to emulate the excellent `varioref` package which substitutes "on facing page" or "on this page" or etc. Usage: The symbol macros are defined in `section\fullref{sec:simple}` which produces: section 2.1 on page 2.

```
\newcommand{\fullref}[1]{~\ref{#1} on page~\pageref{#1}}
```

`\filedescribe` File name and date—One would shove this just before `\end{document}`. It helps to keep the (manual) revision control in order. For an automatic system, you need rcs.sty and RCS, of course :-) We can demonstrate it in a `minipage`:

This is a minipage
ArcMacro.TEX **September 17, 2020**

```
77 \newcommand{\filedescribe}{\vfill\bfseries\tiny\jobname .{\TeX }
78                            \space\footnotesize\today
79                           }
```

`\matlab` To typeset the MATLAB® logo.

```
80 \newcommand{\matlab}{\textsc{Matlab}\raisebox{1ex}{\tiny\Pisymbol{psy}{210}}\xspace}
```

`\mathbox` Draws a box around an equation. So for an important equation, we can ensure displaystyle, and issue `\Beqn \mathbox{\curl{H}=\vec{J} } \Eeqn` which produces:

$$\boxed{\nabla \times \mathbf{H} = \mathbf{J}} \tag{4}$$

```
81 \newcommand{\mathbox}[1]{\fbox{$\displaystyle #1$}}
```

`\CR` As opposed to `\cr` in TEX or `\\` in LATEX. It is less useful in LATEX 2_ε, but is useful sometimes :-) Example of use in next command.

```
82 \newcommand{\CR}{\nonumber\\[2ex]}
```

`\ds` Shortcut to ensure `displaystyle` is used. The non-usage of it tends to really make matters unseeable. Again, $\mathcal{AMS}$-TEXprovides ***far*** better alignment environments. Compare

```
% $$\begin{array}{rl}
%     \ds {\rm VSWR} &= \ds{V_{\rm max}\over V_{\rm min}} = {|V_i| + |V_r|
%             \over |V_i| - |V_r|} \CR
%                &= \ds {1+|V_r/V_i| \over 1-|V_r/V_i|}
%   \end{array} $$
%
```

which produces:

$$\begin{aligned} \text{VSWR} &= \frac{V_{\max}}{V_{\min}} = \frac{|V_i| + |V_r|}{|V_i| - |V_r|} \\ &= \frac{1 + |V_r/V_i|}{1 - |V_r/V_i|} \end{aligned}$$

as compared to (non `\ds`, and `\CR` replaced by the standard `\\`):

$$\begin{aligned} \text{VSWR} &= \tfrac{V_{\max}}{V_{\min}} = \tfrac{|V_i|+|V_r|}{|V_i|-|V_r|} \\ &= \tfrac{1+|V_r/V_i|}{1-|V_r/V_i|} \end{aligned}$$

```
83 \newcommand{\ds}{\displaystyle}
```

`\NB` `\NB{Belangrik}` Puts text in a shadow box with `\par` 's above and below, like:

Belangrik

Unfortunately, I dont know the equivalent LATEX 2_ε-ism for `\long\def`

```
84 \long\def\NB#1{\par\shabox{#1}\par}
```

`\hr` Horizontal Rule—a la HTML `<HR>` ie `\hr` produces

```
\newcommand{\hr}{\vspace*{2mm}\hrule}
```

`\href` `\href{reference}{anchor text}` Used as `\href{ftp://www-em.ee.wits.ac.za/pub/misc/smithchart.ljt}{Smith Chart}`. Hypertext reference. In LaTeX 2_ε, this simply inserts the "anchor text", whereas when the file is run through tth (TeX to HTML), the anchor text is highlighted, the content being the reference.

```
\providecommand{\href}[2]{#2}
```

`\PARstart` This macro allows `\PARstart{S}{tarts}` to produce:

STARTS a paragraph with a big start. Stolen from IEEEtrans.cls. It really is quite effective in starting a paragraph, and I need to continue waxing lyrical to get at least another line!!

```
\providecommand{\PARstart}[2]{\begingroup\def\par{\endgraf\endgroup\lineskiplimit=0pt}
    \setbox2=\hbox{\uppercase{#2} }\newdimen\tmpht \tmpht \ht2
    \advance\tmpht by \baselineskip\font\hhuge=cmr10 at \tmpht
    \setbox1=\hbox{{\hhuge #1}}
    \count7=\tmpht \count8=\ht1\divide\count8 by 1000 \divide\count7 by\count8
    \tmpht=.001\tmpht\multiply\tmpht by \count7\font\hhuge=cmr10 at \tmpht
    \setbox1=\hbox{{\hhuge #1}} \noindent \hangindent1.05\wd1
    \hangafter=-2 {\hskip-\hangindent \lower1\ht1\hbox{\raise1.0\ht2\copy1}%
    \kern-0\wd1}\copy2\lineskiplimit=-1000pt}
```

`\exercise` Used to differentiate "3-minute exercises" in my lecture notes. Used as: `\exercise{This is an exercise\Bit\item One\item Two\Eit}`

3-min Exercise 1

This is an exercise

- *One*
- *Two*

.

```
\newcounter{exerciseCounter}
\newcommand{\exercise}[1]{
  \begin{center}
  \stepcounter{exerciseCounter}
  \ovalbox{\begin{minipage}{0.97\textwidth}
  {\Large\begin{center} 3-min Exercise \theexerciseCounter\end{center}}
  {\slshape #1}\vspace{3mm}\end{minipage}\par}
  \end{center}
}
```

2.4 Section D—Graphics Inclusion.

`\inputfig` 20190820: Scrap all the old crap. Complete re-write. Usual use is with pdflatex(), but still use the latex-dvips-ps2pdf chain for specific eps-only things: EXAMPLES....

Re-using some of these commands, we are going to have `\inputfig[ht]{fn}{caption}, \inputeps[ht]{fn}{cap` where [ht] is an optional argument for the height in mm, which defaults to a pleasing 80mm, or 70mm in `\twocolumn` format. Notably, a new extension is that `\inputfig` will auto find .pdf .png and .jpg and handle them appropriately, AND height=0 will be interpreted as "raw": unscaled!!! This is especially useful for Circuit Macros.

```
\newcounter{figheight}
\setcounter{figheight}{80}
\DeclareOption{twocolumn}{\setcounter{figheight}{70}}
\ProcessOptions\relax
\newcommand{\inputfig}[3][\thefigheight]%
  {%
    \begin{figure}[!htb]%
      \centering%
      \ifnum#1=0%
        \includegraphics[scale=1]{#2}%
      \else%
        \includegraphics[height=#1mm]{#2}%
      \fi%
      \caption{#3}%    includes fig num
      \label{fig:#2}%  ie may \ref{fig:#2}
    \end{figure}%
  }
```

This section of macros keeps growing as slightly different requirements are brought forward. Ultimately, one gets into visual formatting, which is generally a *Bad Thing*. In such situations, it is probably better to introduce your own command, analogous to these, directly in your document. Its probably not a good idea to fiddle and mess around with these definitions, since they work *consistently* with a wide variety of situations. LaTeX 2ε provides a ***far*** more robust float mechanism than LaTeX, obviating the use of the **Here *Dammit*** package[1]. In addition, LaTeX 2ε provides a mechanism for providing optional parameters to commands, thus obviating the need for the include [small—medium—large—smaller—special—BlerryGroot] variations on each picture type and reduces the number of commands to be defined.

Yes, the `graphics` and `graphicx` packages do provide a "better way", and I have modified many of the commands that I regularly use to use them, but the native packages still fall short of what "we really want to do". It is thus still necessary to issue a `\inputeps{fn}{Caption}` to do all the right stuff.

In general, the approach stems from the old Lotus Manuscript days, where `\label` and `\ref` were unavailable[2] (Still are in any meaningful way in Turd, maar toemaar). Each macro takes the first parameter as the filename (with no extension), which automatically defines the `\label`, prepended by `fig:`. Thus you can refer to the figure by `\ref{fig:fn}`. It is still necessary to differentiate between the extensions, as the treatment is different. (No longer strictly true with `\includegraphics`, but I dont generally use anything other than `.fig` or `.eps`, and their treatment *is* different!

The second parameter defines the caption of the figure, which is included in the Table of Figures, and printed under the figure, prepended by `Figure` and the figure number. The third parameter is an optional width, which can be used to make the figure larger. Aspect ratio's are maintained automatically (well, in eps anyway :-)

There is one subtlety of this Clarkian Method, which is often overlooked by initiates to the Way, and that is that if you include the same graphic in the document *twice*, as you often do in an Executive Summary/Main Report type of document, then the same label is defined twice!! which causes consternation to LaTeX 2ε. The simplest way around that is to use a symbolic link to the graphic, using another name. Under MsLoss, this cannot be done, and a physical copy must unfortunately be made.[3]

`\inputfig` The general trend has been to move away from TeXCad, GnuPlot, and emlines, and to get to eps. MATLAB® puts out eps, as does `xfig` which is an excellent CAD package. Being

[1] Although it is used in the `float` package. If you are having float trouble, investigate the `float` package—it is very powerful!

[2] In Manuscript, one would keep the file names in this fashion, until the final version, where you would manually count the figures and do a global substitution of the names to numbers!

[3] Links (Shortcuts) do exist under '95, but cannot be "used" as ordinary files, which is *really stupid (aka Gatesian)*.

Linux based, I use `xfig` for everything these days, and hardly ever use any of my other, older graphics inclusion macros. Others do however :-)

To take full advantage of LaTeX 2_ε maths etc in the figure, `xfig` needs to be called by the little script `fig`:

and to then use the export `Combined PS/LaTeX`, both the `(PS part)` and the `(LaTeX)` part. Newer versions of xfig put out both parts in one operation. A further hint is that `xfig` should generally be told to have a 2:1 scale and a maximum grid, which makes the sizing about right. This combination (and script) allows the combination of any standard LaTeX 2_ε command to be embedded in the `fig` diagram, particularly Maths. `\inputfig` does not need size changes, as everything is correctly scaled from the CAD package. It is used very simply as

```
\inputfig{daisy}{$I$ in Daisy, due to $\sigma\neq\infty$, \implies boerewors?}
```

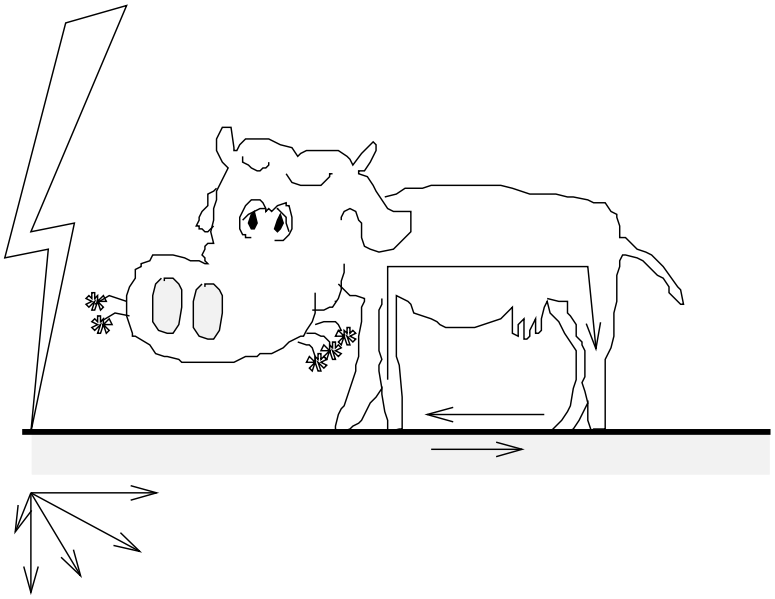


Figure 1: I in Daisy, due to $\sigma \neq \infty$, $\Rightarrow$ boerewors?

```
%% \newcommand{\inputfig}[2]%
%%   {%
%%     \begin{figure}[!htb]%
%%       \begin{center}%
%%          \input{#1.pst}%  it will pull in the pstex
%%          \caption{#2}%    includes fig num
%%          \label{fig:#1}%  ie may \ref{fig:#1}
%%       \end{center}%
%%     \end{figure}%
%%   }
%%
```

`\inputcct` A recent addition to my stable of packages to be used is the set of m4 and gpic based Circuit macros, currently at version 4.6. These allow really excellent circuits to be drawn without resorting to a CAD package. The language is really easy, figs can be seen in `xdvi` in good quality, and are rendered by `dvips` in excellent quality. Trouble with CAD and circuits is that you dont get it *quite* right, and even with `xfig`, keeping a library of elements is difficult. Used as `\inputcct{DualBridge}{Dual Bridge Power Supply}` in the usual fashion.

```
\newcommand{\inputcct}[2]%
  {%
    \begin{figure}[!htb]%
```

Figure 2: Dual Bridge Power Supply

```
136       \begin{center}%
137          \input{#1.cct}\ \box\graph%
138          \caption{#2}%
139          \label{fig:#1}% ie may \ref{fig:#1}
140       \end{center}%
141     \end{figure}%
142   }
```

`\inputeps` To include a standard .eps figure, as from MATLAB® etc, simply use

`\inputeps{dakpap}{Dakota measured vs predicted Signal Strength}`

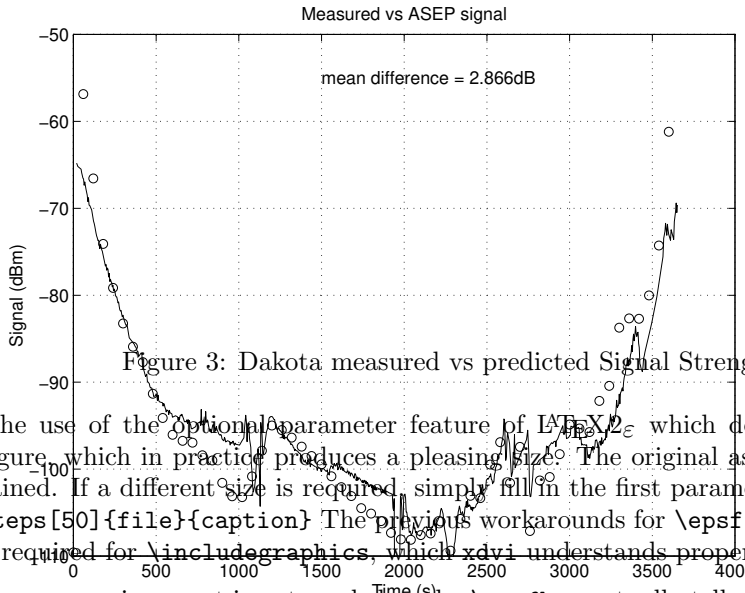
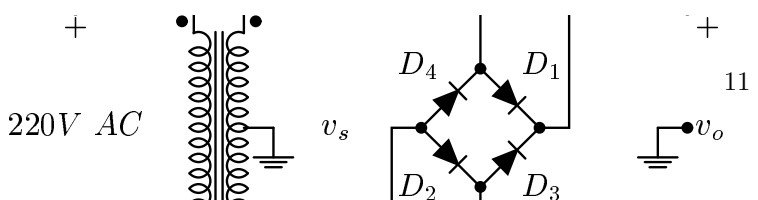


Figure 3: Dakota measured vs predicted Signal Strength

Note the use of the optional parameter feature of LaTeX 2$_\varepsilon$ which defaults to an 80mm high figure, which in practice produces a pleasing size. The original aspect ratio is always maintained. If a different size is required, simply fill in the first parameter (in mm). (as in `\inputeps[50]{file}{caption}` The previous workarounds for `\epsfig` and `\epsf` are no longer required for `\includegraphics`, which `xdvi` understands properly!

The `picture` environment is not needed as the `\epsfbox` actually tells LaTeX 2$_\varepsilon$ the bounding box details. But TeX's vertical mode must be suspended during the centre environment. This workaround is *not* necessary for the `epsfig` package (as opposed to the older `epsf` package), but `xdvi` tends to lose its marbles, and no longer calls `ghostscript` in the background when viewing an eps graphic. Hence `epsf` :-)



I need the 80 to be 65 in the twocolumn case!. This code worked first time!! Options are brilliant!!

```
\newcounter{epsheight}
\setcounter{epsheight}{80}
\DeclareOption{twocolumn}{\setcounter{epsheight}{70}}
\ProcessOptions\relax
\newcommand{\inputeps}[3][\theepsheight]%
  {%
    \begin{figure}[!htb]%
      \centering%
      \includegraphics[height=#1mm]{#2.eps}%
      \caption{#3}%    includes fig num
      \label{fig:#2}%  ie may \ref{fig:#2}
    \end{figure}%
  }
```

\input[...]eps

\inputsmalleps and \inputbigeps are deprecated commands, *only supplied for backward compatibility* they being superceded by the use of the optional parameter feature of LaTeX 2$_\varepsilon$, and in fact implemented using it! We'll give daisy a shrink using

```
\inputsmalleps{daisy1}{Honey, I shrunk the kids}
```

and a bit of a blow up using

```
\inputbigeps{daisy2}{Honey, I grew the kids}
```

It is simply better to use `\inputeps[105]{daisy2}{Honey, I grew the kids}` in the first place.

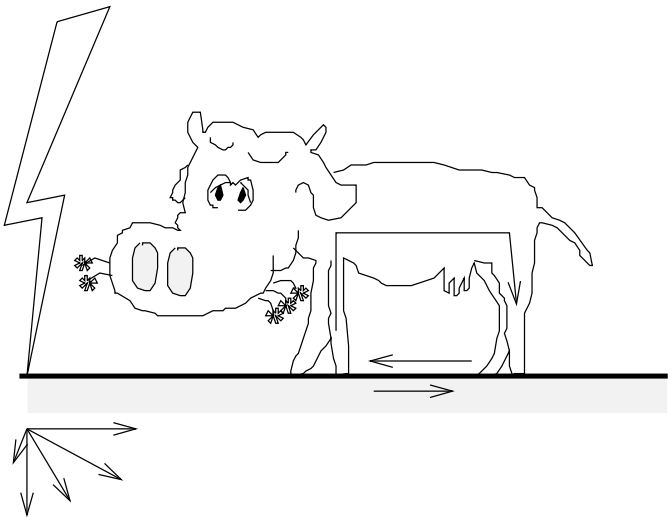


Figure 4: Honey, I shrunk the kids

```
\newcommand{\inputsmalleps}[2]{\inputeps[70]{#1}{#2}}
\newcommand{\inputbigeps}[2]{\inputeps[105]{#1}{#2}}
```

\inputepsraw

To include a "raw" .eps figure, as from Circuit Macros, that MUST NOT BE RESIZED!

```
\newcommand{\inputepsraw}[2]%
  {%
    \begin{figure}[!htb]%
      \centering%
      \includegraphics{#1.eps}%
      \caption{#2}%    includes fig num
      \label{fig:#1}%  ie may \ref{fig:#1}
```

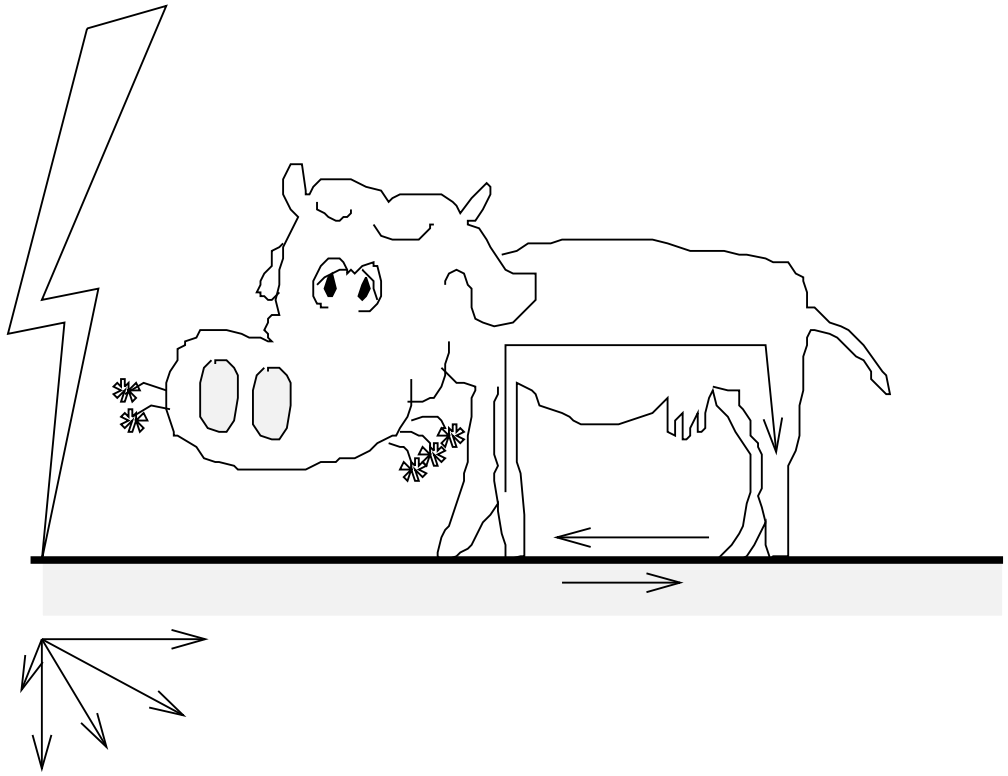


Figure 5: Honey, I grew the kids

```
    \end{figure}%
}
```

All following commands are provided as a service to humanity :-) I no longer use them, but they are provided for "backward compatibility"[4].

In the case of other file formats, I strongly prefer `.eps` as it is scaleable and portable—others aren't, and you have to tell LaTeX 2$_\varepsilon$ what *size* they are, so that it can leave whitespace, and include them by a `\special` or they wont be correctly scaled etc. Various packages put them out with different anchor points, so what works for one graphics package doesn't work for another (Particularly `.pcx` with its myriads of types). This is completely non-portable, and you are constantly having to worry about how big the thing is, what offsets to use, what resolution the target printer is (for `.pcx`) etc. Like I said—Deprecated commands, and intelligent packages like ImageMagick exist to do proper conversions etc.

In the case of the double and two figure stuff, this is generally in the realm of visual formatting again, and *MUCH* more control is offered by the `floatfig, subfigure, wrapfigure` packages. They are GOOD, and provide the (a) (b) control easily and effectively.

Thus no examples have been provided, and I haven't even bothered to translate them from LaTeX to LaTeX 2$_\varepsilon$.

`\inputtwofig` Two figs on top of one another. One caption. See `subfigure` package for a better way.

```
\def\inputtwofig#1#2#3 % 3 optional parameters
          {
             \begin{figure}[!htb]
               \begin{center}
         \input{#1.pst}\\ %  it will pull in the pstex
```

[4]Ever tried running a dos 3.0 serial port terminal package under NT :-)

```
         \input{#2.pst} %second one
                \caption{#3}   % includes fig num
                 \label{fig:#1} % ie may \ref{fig:#1}
              \end{center}
            \end{figure}
          }
```

\doublefig Two figs side by side. One caption. See `subfigure` for a better way.

```
\def\doublefig#1#2#3 % 3 optional parameters
          {
             \begin{figure}[!htb]
               \begin{center}
                 \begin{minipage}[b]{70mm}
                   \input{#1.pst}\\ %  it will pull in the pstex
                 \end{minipage}
                 \hspace{5mm}
                 \begin{minipage}[b]{70mm}
                   \input{#2.pst} %second one
                 \end{minipage}
                 \caption{#3}   % includes fig num
                  \label{fig:#1} % ie may \ref{fig:#1}
               \end{center}
             \end{figure}
           }
```

\inputpic For a `pic` from TEXCaD, under MsLoss.

```
\def\inputpic#1#2
          {
             \begin{figure}[!htb]
               \begin{center}
         \input{#1.pic} % default ext is .PIC
                \caption{#2}   % includes fig num
                 \label{fig:#1} % ie may \ref{fig:#1}
               \end{center}
             \end{figure}
           }
```

\pic A simple insertion of a .pic file — no caption, figure number, and no floating about. Most useful for lecture note purposes!

```
\def\pic#1{\begin{center}\input{#1.pic}\end{center}}
```

\inputpcx For the auto inputting of .pcx using the STANDARD Prof H. dims.... Using the Standard Clarkian Labelling... ie refer using `\ref{fig:FilenameNoExt}` Use MAT2PCX.bat Filename from a Matlab Metafile, or HPGL2PCX.bat Filename from Freelance, AutoSketch, 123W.... for the correct sizing, offsetting, orientation etc.... (Orientations are DIFFERENT from these packages)

(Works for MATLAB® 3.5 only)

```
\def\inputpcx#1#2
          {\setlength{\unitlength}{1in}
             \begin{figure}[!htb]
               \begin{center}
                 \begin{picture}(5,3.33)(0,0)
                   \put(0.0,3.33){\special{em:graph #1.pcx}}%
                 \end{picture} %
               \caption{#2}   % includes fig num
               \label{fig:#1} % ie may \ref{fig:#1}
             \end{center}
```

```
 \end{figure}
}
```

\inputtwopcx For the auto inputting of two .pcx's using the STANDARD Prof H. dims.... Using the Standard Clarkian Labelling... ie refer using \ref{fig:FilenameNoExtPCX1} Note First parm used as ref.. Use MAT2PCX Filename from a Matlab Metafile, or HPGL2PCX Filename from Freelance, AutoSketch, 123W.... for the correct sizing, offsetting, orientation etc....

Usage: \inputtwopcx{FilenameNoExtPCX1}{FileNameNoExtPCX2}{Caption}

```
\def\inputtwopcx#1#2#3
          {\setlength{\unitlength}{1in}
             \begin{figure}[!htb]
               \begin{center}
                 \begin{picture}(5,6.66)(0,0)
                   \put(0.0,3.33){\special{em:graph #2.pcx}}%
                   \put(0.0,6.66){\special{em:graph #1.pcx}}%
                 \end{picture} %
               \caption{#3}   % includes fig num
               \label{fig:#1} % ie may \ref{fig:#1}
             \end{center}
           \end{figure}
          }
```

\inputsmallpcx For the auto inputting of one .pcx using dims suitable for \twocolumn Using the Standard Clarkian Labelling... ie refer using \ref{fig:FilenameNoExtPCX1} Note First parm used as ref.. Use MAT2PCXS Filename from a Matlab Metafile (S=small) for the correct sizing, offsetting, orientation etc....

Usage: \inputsmallpcx{FilenameNoExtPCX1}{FileNameNoExtPCX2}{Caption}

```
\def\inputsmallpcx#1#2
          {\setlength{\unitlength}{1in}
             \begin{figure}[H]
               \begin{center}
                 \begin{picture}(3.1,2.1)(0,0)
                   \put(0.0,2.1){\special{em:graph #1.pcx}}%
                 \end{picture} %
               \caption{#2}   % includes fig num
               \label{fig:#1} % ie may \ref{fig:#1}
             \end{center}
           \end{figure}
          }
```

\inputepspic Load an eps next to a small pic. Useful for 2d radiation patterns with a small pic of the orientation of the plane. Since there are so many, enforce HERE !! At 58mm, get 3 per page – with 59mm, get only two!!!!!!

```
\def\inputepspicH#1#2#3 % 3 optional parameters
          {
           \setlength{\unitlength}{1mm}
           \begin{figure}[H]
             \begin{center}
               \begin{minipage}[c]{85mm}%
                 \begin{picture}(80,58)(-0,0)%
                 \epsfysize=58mm \epsfbox{#1.eps}%
                 \end{picture}%
               \end{minipage}%
               \hspace{15mm}%
               \begin{minipage}[c]{50mm}%
                 \input{#2.pic} %second one
               \end{minipage}%
```

```
256               \caption{#3}   % includes fig num
257                \label{fig:#1} % ie may \ref{fig:#1}
258             \end{center}
259           \end{figure}
260           }
```

2.5 Section E—Odd Hyphenations.

```
261 \hyphenation{im-pe-dance}
262 \hyphenation{di-pole}
263 \hyphenation{mono-pole}
264 \hyphenation{ad-mit-tance}
```