The gensymb package for LaTeX 2ε

Walter Schmidt*

(v1.0 – 2003/07/02)

1 The problem

Typesetting units of measurement with LaTeX is sometimes difficult. Why? For instance, most (but no all) typefaces have an upright µ and also a degree symbol, but there is no obvious way to use these in math mode. On the other hand, an upright Ω is part of all math fonts for TeX, but most text fonts are lacking the corresponding `\textohm`. Thus, it is not only impossible to use the same notation in text and in formulas; depending on the fonts used, it may even be impossible to typeset units properly at all, if you are restricted to the means of 'standard' LaTeX.

2 The macro package

The gensymb package provides a number of 'generic' macros, which produce the same output in text and math mode:

`\degree`	`\celsius`	`\perthousand`	`\ohm`	`\micro`
°	℃	‰	Ω	µ

Thus, you can write, for instance:

```
$ \lambda = 10.6\,\micro\mathrm{m} $
... a wavelength of 10.6\,\micro m
```

With the help of the package units, you can even apply exactly the same notation in text and formulas:

```
$ \lambda = \unit[10.6]{\micro m} $
... a wavelength of \unit[10.6]{\micro m}
```

Under normal circumstances you should use gensymb in conjunction with the textcomp package. In this case the above symbols are taken from a text font using the TS1 (text companion) encoding – even in math. As a result, they will exhibit the style of the surrounding text or of the `\mathrm` alphabet, respectively.

When textcomp is *not* used (for instance, because your text fonts are not available with TS1 encoding), the gensymb package tries to emulate the above symbols using what is available in the math fonts. However, the symbols µ and

*w.a.schmidt@gmx.net

‰ cannot be faked, and the package will issue appropriate warning messages. The option upmu is a workaround to provide at least the µ – see below.

The symbol \ohm is particular, too: Loading the textcomp package does *not ensure* that the text fonts actually include an Ω. Unfortunately, many fonts don't. LaTeX cannot detect this situation in advance, and the command \ohm will print some garbage then – possibly without any error message. A workaround is to use the options Omega or Upomega described below.

The default behavior of the gensymb package can be modified using the following options:

upmu With this option, the \micro uses internally a math symbol with the name \upmu. The responsibility to provide this macro lies with you; for instance, load the package upgreek[1] or use a set of math fonts which include an upright mu and make sure it is accessible as \upmu. This option is useful, when your text fonts are not available with TS1 (text companion) encoding, or when their \textmu is ugly or broken. The drawback is, that the style of this µ will not vary with the surrounding font. It is always upright roman, regardless of whether the surrounding text is sans serif, bold or whatever.

Omega makes \ohm always use the upright Greek Omega from the current math font, regardless of whether textcomp is loaded. This makes sense, when there is no Ω in your text font(s). The option works always, but the style of the Omega in text will not vary then.

Various packages provide an option slantedGreek to make uppercase Greek letters in formulas slanted. You may safely use this option in conjunction with the Omega option of the package gensymb: \ohm will be upright, though!

Upomega A math symbol with the name \Upomega will be used to for the \ohm. The responsibility to provide this macro lies with you; for instance, load the package upgreek[1]. Again, the style of this \ohm in text never changes. The option may be useful, when the text font does not include a \textomega of its own and the mathematical \Omega is not a suitable substitute, for one reason or another.

3 Examples

The optimum case

With text fonts providing µ as well as Ω in the text companion encoding, the default behavior of gensymb is appropriate. This holds, e.g., for the EC/TC fonts (i.e., Computer Modern with T1 and TS1 encoding), Palatino, Lucida Bright, the Fourier fonts, and most font families from MicroPress:

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage{gensymb}
```

[1] available from the same CTAN directory as gensymb

The normal case

Many commercial text fonts have a correct 'micro' symbol, but no Omega. In this case the package can be loaded with the Omega option, to make sure that the mathematical Omega is used instead:

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage{mathptmx} % for instance
\usepackage[Omega]{gensymb}
```

Using the 'classical' CM fonts

When you are restricted to the classical Computer Modern fonts with OT1 encoding, you can still take the Omega from the math fonts, but you need to take the upright mu from an an extra font such as Euler Roman. Notice, however, that `\perthousand` is not made available. You may try `\permil` instead, which is provided in the WASY fonts.

```
\usepackage[Euler]{upgreek}
\usepackage[Omega, upmu]{gensymb}
\usepackage{wasysym}
\let\perthousand=\permil
```

A special case

Imagine that you are using the Bitstream Charter text fonts in conjunction with Euler-VM for math.[2] Charter, like most typefaces from Bitstream, has a wrong mu symbol. The missing Omega could be taken from the math font (Euler), but it would not go well with Charter in a combination such as MΩ. As a workaround, load the upgreek package and take both `\micro` and `\ohm` from the 'Adobe Symbol' font:

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage{charter,eulervm}
\usepackage[Symbol]{upgreek}
\usepackage[Upomega, upmu]{gensymb}
```

4 Using gensymb together with other packages

- gensymb goes well with the package units, but it cannot be used in conjunction with the package SIunits, because the latter has its own means to typeset prefixes.
- Using the package mathcomp together with gensymb is possible, if you need the additional 'math companion' symbols. Both packages will use only one common math symbol font.

[2] I do not actually recommend this combination...

5 Known deficiencies

The current version of the package does not work properly in conjunction with the LY1 font encoding. This will possibly be fixed in a future release, if there is an actual need.

6 Implementation

The options are implemented using switches:

```
⟨*package⟩
\newif\ifgns@Upomega
\newif\ifgns@Omega
\newif\ifgns@upmu
\DeclareOption{Upomega}{\gns@Upomegatrue}
\DeclareOption{Omega}{\gns@Omegatrue}
\DeclareOption{upmu}{\gns@upmutrue}
\ProcessOptions\relax
```

These are the generic commands:

```
\newcommand\celsius{}
\newcommand\degree{}
\newcommand\ohm{}
```

The following macros will be used later to actually define the generic commands. We start with a macro to fake `\degree` and `\celsius` and to generate a warning message for `\perthousand`:

```
\def\gns@usefakedsymbols{%
\renewcommand{\degree}{\ensuremath{^\circ}}
\DeclareRobustCommand{\celsius}{%
  \ifmmode^\circ\mathrm{C}\else$^\circ$C\fi}
\PackageInfo{gensymb}{%
  Faking symbols for \protect\degree\space and \protect\celsius}
\PackageWarningNoLine{gensymb}{%
  Not defining \protect\perthousand}
}
```

The following macro sets up a SymbolFont for the text companion symbols to be used in math. If `\tcmu` is already defined, we assume that the mathcomp package is already loaded, and that the 'math companion' symbols need not be defined once again:

```
\def\gns@setupmathcomp{%
\expandafter\ifx\csname tcmu\endcsname\relax
  \DeclareSymbolFont{gns@font}{TS1}{\familydefault}{m}{n}
  \ifx\mv@bold\@undefined\else
    \SetSymbolFont{gns@font}{bold}{TS1}{\familydefault}{\bfdefault}{n}
  \fi
  \DeclareMathSymbol{\tccelsius}{\mathord}{gns@font}{137}     % {'211}
  \DeclareMathSymbol{\tcdegree}{\mathord}{gns@font}{176}      % {'260}
  \DeclareMathSymbol{\tcperthousand}{\mathord}{gns@font}{135} % {'207}
  \DeclareMathSymbol{\tcmu}{\mathord}{gns@font}{181}          % {'265}
  \DeclareMathSymbol{\tcohm}{\mathord}{gns@font}{87}          % {'127}
  \PackageInfo{gensymb}{Math companion symbols declared}
\else
```

```
  \PackageInfo{gensymb}{Math companion symbols found}
\fi
}
```

A macro to define `\degree`, `\celsius` and `\perthousand` so as to use text companion symbols:

```
\def\gns@usetcsymbols{%
\DeclareRobustCommand{\degree}{%
  \ifmmode\tcdegree\else\textdegree\fi}
\DeclareRobustCommand{\celsius}{%
  \ifmmode\tccelsius\else\textcelsius\fi}
\DeclareRobustCommand{\perthousand}{%
  \ifmmode\tcperthousand\else\textperthousand\fi}
\PackageInfo{gensymb}{Using text companion symbols for %
  \protect\degree, \protect\celsius\space and \protect\perthousand}
}
```

A macro to define `\ohm` so as to use the math symbol `\Upomega`:

```
\def\gns@useUpomega{%
  \expandafter\ifx\csname Upomega\endcsname\relax
    \PackageError{gensymb}
    {You have requested the option Upomega,\MessageBreak
    but the command \protect\Upomega\space is undefined}
    {Load the upgreek package additionally and try again!}
  \fi
  \DeclareRobustCommand{\ohm}{\ifmmode\Upomega\else$\Upomega$\fi}
  \PackageInfo{gensymb}{Using \protect\Upomega\space for \protect\ohm}
}
```

A macro to define `\ohm` so as to use `\Omega` (or `\upOmega`, if it is provided):

```
\def\gns@useOmega{%
  \expandafter\ifx\csname upOmega\endcsname\relax
   \DeclareRobustCommand{\ohm}{\ifmmode\Omega\else$\Omega$\fi}
   \PackageInfo{gensymb}{Using \protect\Omega\space for \protect\ohm}
  \else
   \DeclareRobustCommand{\ohm}{\ifmmode\upOmega\else$\upOmega$\fi}
   \PackageInfo{gensymb}{Using \protect\upOmega\space for \protect\ohm}
  \fi
}
```

A macro to define `\ohm` so as to use the text companion symbol:

```
\def\gns@usetextohm{%
  \DeclareRobustCommand{\ohm}{\ifmmode\tcohm\else\textohm\fi}
  \PackageInfo{gensymb}{Using \protect\textohm\space for \protect\ohm}
}
```

A macro to define `\micro` so as to use the math symbol `\upmu`:

```
\def\gns@useupmu{%
  \expandafter\ifx\csname upmu\endcsname\relax
    \PackageError{gensymb}
    {You have requested the option upmu,\MessageBreak
    but the command \protect\upmu\space is undefined}
    {Load the upgreek package additionally and try again!}
  \fi
  \DeclareRobustCommand{\micro}{\ifmmode\upmu\else$\upmu$\fi}
  \PackageInfo{gensymb}{Using \protect\upmu\space for \protect\micro}
```

```
}
```

A macro to define \micro so as to use the text companion symbol:

```
\def\gns@usetextmu{%
  \DeclareRobustCommand{\micro}{\ifmmode\tcmu\else\textmu\fi}
  \PackageInfo{gensymb}{Using \protect\textmu\space for \protect\micro}
}
```

A macro to issue a warning, if \micro cannot be made available:

```
\def\gns@usenomu{%
  \PackageWarningNoLine{gensymb}{%
    Not defining \protect\micro}
}
```

The actual work is executed AtBeginDocument, so that we can detect the presence of the textcomp package and the availability of \Upmu and other commands, regardless of the sequence of loading the packages.

```
\AtBeginDocument{%
```

First, we check for textcomp. If it is loaded, we set up the text companion symbols for use in math and define \degree, \celsius and \perthousand so that they use these. Otherwise, the symbols are faked as far as possible:

```
\@ifpackageloaded{textcomp}{\gns@setupmathcomp\gns@usetcsymbols}%
  {\gns@usefakedsymbols}
```

Now we define \ohm. In case the options Upomega or Omega have been specified, behave acordingly. Otherwise, use the text companion symbol, if available. Default is to use the mathematical Omega:

```
\ifgns@Upomega
  \gns@useUpomega
\else
\ifgns@Omega
  \gns@useOmega
\else
\@ifpackageloaded{textcomp}%
  {\gns@usetextohm}%
  {\gns@useOmega}
\fi\fi
```

Next, define \micro. In case the option upmu has been specified, behave acordingly. Otherwise, use the text companion symbol, if available. Default is not to provide \micro at all.

```
\ifgns@upmu
  \gns@useupmu
\else
\@ifpackageloaded{textcomp}%
  {\gns@usetextmu}%
  {\gns@usenomu}
\fi
```

Finally, we destroy the commands that are no longer needed:

```
\let\gns@usefakedsymbols\relax
\let\gns@setupmathcomp\relax
\let\gns@usetcsymbols\relax
\let\gns@useUpomega\relax
\let\gns@useOmega\relax
```

```
\let\gns@usetextohm\relax
\let\gns@useupmu\relax
\let\gns@usenomu\relax
\let\gns@usetextmu\relax
}
⟨/package⟩
```

The next line of code is only to prevent DocStrip from adding the character table to all modules:

```
\endinput
```