The **formular** Package.*

Hans-Christoph Wirth  
hcw@gmx.de  
2005/06/15

*This package provides some commands useful for typesetting fields in formulas which are intended to be filled either manually or using TeX.*

% Copyright 2001 Hans-Christoph Wirth (hcw@gmx.de)  
%  
% This program/package may be distributed and/or modified under the  
% conditions of the LaTeX Project Public License, either version 1.2  
% of this license or (at your option) any later version.  
% The latest version of this license is available at  
% http://www.latex-project.org/lppl.txt  
% and version 1.2 or later is part of all distributions of LaTeX  
% version 1999/12/01 or later.  
%  
% The quintessence of LPPL is:  
% - Distribute the package only in its full contents  
% - If you modify any files, rename them before  
%  
% This program/package consists of the files  
% formular.ins (driver file)  
% formular.dtx (source and documentation)  
% formular.sty (actual style file, generated)  
% formular.dvi (documentation, generated)

1 Examples

When typesetting forms there often arises the need for defining fields which consists of one or more lines where the customer can write something down manually. To support a unique appearance of those fields we contribute some commands which define general fields.

1.1 One-line Fields

The following piece of code declares and uses a simple one-line field **namef**:

---

*This file has version number v1.0a – (c) 2001 by Hans-Christoph Wirth, dated 2005/06/15.*
\newFRMfield{namef}{15mm}
This is \useFRMfield{namef}{John} and \useFRMfield{namef}.

The output is the following:

This is John and ______.

More complicated fields may have a description and a default content.

\newFRMfield{namef}{15mm}[Name][nobody]
This is \useFRMfield{namef}{John} and \useFRMfield{namef}
and \useFRMfield{namef}[] and \useFRMfield{namef}[Fred Long Name]

Notice that the field grows with the content:

This is John Name and nobody Name and Name and Fred Long Name Name

1.2 Different Styles

Each class of fields can have its own font style. Additionally there is a \textit{ruled} style implemented. In the following we declare two one-line fields with different appearance:

\newFRMfield{placef}{40mm}[Place]
\setFRMfontfamily{cmr}
\setFRMfontshape{it}
\setFRMfontsize{12}
\setFRMruledstyle
\newFRMfield{sigf}{30mm}[Signature]
\useFRMfield{placef}[Sometown], \useFRMfield{sigf}[U. N. Known]

\begin{tabular}{l}
Sometown \\
Place \\
\hline
\end{tabular}

\begin{tabular}{l}
\hline
U. N. Known \\
Signature
\end{tabular}

1.3 Multi-line Environments

The following piece of code declares and uses two multi-line environments. Notice that there are two styles: The description of the field may appear on a separate line or not. Each environment has a description and a minimal number of lines.

\setFRMbreakstyle
\newFRMenvironment{env1}{Foobar}{2}
\setFRMinlinestyle
\newFRMenvironment{env2}{Barfoo}{3}

\begin{env1} This is break style \end{env1}
\begin{env2} This is inline style \end{env2}

This is break style

This is inline style

1.4 Containers

A container is a collection of fields. The container specifies which fields belong to it, and where the fields are to be printed. The following piece of code declares a container which contains three fields.

\newFRMcontainer{Grades}
  \setFRMruledstyle
  \newFRMfield{Ma}{30mm[]}[
  \newFRMfield{Ph}{30mm[]}[
  \newFRMfield{En}{30mm}[english grade][]
  ]%
\parbox[t]{0.45\linewidth}{\baselineskip18pt
  Maths \dotfill \useFRMfield{Ma}\newline
  Physics \dotfill \useFRMfield{Ph}\hfill
}\parbox[t]{0.45\linewidth}{%
  English \dotfill \useFRMfield{En}}%
\begin{Grades}
  \setMa{excellent}
  \setPh{very good}
\end{Grades}

Maths . . . . . . excellent
Physics . . . . . . very good

2 Command Description

2.1 Style Parameters

The commands explained in this section select the general appearance of the fields. A call to a command affects all fields which are declared subsequently (within the same scope).
\setFRMrulewidth The command

\setFRMrulewidth\{\textit{dimen}\}

sets the thickness of the underlining rules to \textit{\langle dimen \rangle} (default: 0.1pt). The instruction \texttt{\setFRMrulewidth\{0pt\}} makes rules disappear.

\setFRMrulesep The command

\setFRMrulesep\{\textit{dimen}\}

sets the vertical distance between the font baseline and the underlining rules (default: 2pt).

\setFRMmargin The command

\setFRMmargin\{\textit{dimen}\}

sets the horizontal indentation of the content of multi-line \texttt{FRM}environments (default: 5pt).

\setFRMbaselineskip The command

\setFRMbaselineskip\{\textit{dimen}\}

sets the baselineskip of the content of multi-line \texttt{FRM}environments (default: 18pt).

\setFRMfontencoding The \texttt{\setFRMfont...} commands select the font used for the content, and the corresponding \texttt{\setFRMdfont...} commands select the font of the description of one-line fields. For a description of the meaning of the parameters we refer to command \texttt{\usefont} in standard \LaTeX\ documentation. Notice that \texttt{\setFRMfontsize} has only one parameter, since the baselineskip is selected with a separate command.

All above commands can be supplied with an optional parameter specifying a (already declared) field. This enables to change the appearance of fields after declaration. The following example illustrates this:

\begin{verbatim}
\newFRMfield\{foo\}\{3cm\}
\setFRMfontsize\{foo\}\{20pt\}
\end{verbatim}

To this end, there are some more commands which have only effect when this optional parameter is supplied:

\setFRMcontent The commands

\setFRMcontent\{\langle field \rangle}\{\langle content \rangle\}
\setFRMdescription\{\langle field \rangle}\{\langle content \rangle\}

set the (default) content and the description of the field, while

\setFRMwidth\{\langle field \rangle}\{\langle width \rangle\}

changes the (minimal) width.
2.2 One-line Fields

A FRMfield consists of three ingredients:

1. The content of the field. This is a (maybe empty) one-line string.
2. A horizontal rule. The length of the rule is at least the minimal width of
   the field, but it grows with the content or the description. It is guaranteed
   that the rule exceeds the content at least by the amount which is set with
   \setFRMmargin.
3. The description. This is a (maybe empty) one-line string printed below the
   rule.

\newFRMfield A FRMfield must be declared as follows:

\newFRMfield{⟨field id⟩}{⟨width⟩}
\newFRMfield{⟨field id⟩}{⟨width⟩}{⟨description⟩}
\newFRMfield{⟨field id⟩}{⟨width⟩}{⟨description⟩}{⟨default content⟩}

The field is associated with the style setting which is valid at this moment.

\useFRMfield When a field is declared, it can be used with the command

\useFRMfield{⟨name⟩}
\useFRMfield{⟨name⟩}{⟨content⟩}

If the ⟨content⟩ is not supplied to \useFRMfield, then the ⟨default content⟩ is
printed.

\renewFRMfield The macro \renewFRMfield is nearly the same as \newFRMfield, but it changes
an existing field rather than defining a new one.

\setFRMruledstyle \setFRMplainstyle Besides the style parameters discussed above, the user has the additional choice
between the following two styles: With

\setFRMplainstyle
the content is underlined by a single rule. With

\setFRMruledstyle
the content is printed over a field of rules. The height of the ruled field adjusts
automatically to the current font size.

2.3 Multi-line Environments

A FRMenvironment consists of three ingredients:

1. The content of the field. This is a (possibly empty) text.
2. A collection of rules which underline the content.

3. The description. This is a one-line string printed at the beginning of the environment.

A FRM environment must be declared as follows:

```
\newFRMenvironment{⟨envid⟩}{⟨description⟩}{⟨default lines⟩}
```

As one may expect, the current style settings are associated with the environment from this point on. The FRM environment is used in the following way:

```
\begin{⟨envid⟩}{⟨content⟩}\end{⟨envid⟩}
\begin{⟨envid⟩}{⟨lines⟩}{⟨content⟩}\end{⟨envid⟩}
```

This will print a FRM environment with description ⟨description⟩ and content ⟨content⟩. The environment extends to at least ⟨default lines⟩ many lines (or to ⟨lines⟩ lines, if the optional argument is supplied). If the ⟨content⟩ does not fit into this space, the environment is further extended and a warning is issued.

The baselineskip of the content can be adjusted with

```
\setFRMbaselineskip{⟨dimen⟩}
```

The indentation of the content can be adjusted with

```
\setFRMmargin{⟨dimen⟩}
```

It defaults to 5pt.

There are two styles implemented: The content may start in the same line as the description (\setFRMinlinestyle), or the description may appear on its own line (\setFRMbreakstyle). Notice that in break style, the description's line is not counted within the line range.

### 2.4 Containers

A FRM container is a simple way to collect several FRM fields into one logical unit. The container defines the set of fields, their default content, and how to print the individual fields. This definition can go to the preamble of the document.

The content of the container will usually consist solely of commands which set the field’s content. So the user must not care about the actual typesetting of the fields.

The command

```
\newFRMcontainer{⟨cid⟩}{⟨init code⟩}{⟨apply code⟩}
```

defines a new container ⟨cid⟩. The container will be used afterwards like a environment
The \textit{init code} may contain any code needed for setup of the container. In particular, one may employ at this point:

- All style setting commands \texttt{\textbackslash setFRM...} of this package

Notice that the container always initializes the settings to the situation which was valid when the call to \texttt{\textbackslash newFRMcontainer} happens. This ensures that multiple calls to the same container always appear in the same style.

- The command \texttt{\textbackslash newFRMfield} to declare local fields.

Within the \textit{init code} of a container, a call to \texttt{\textbackslash newFRMfield\{\textit{field}\}} defines one more command, namely \texttt{\textbackslash set\{\textit{field}\}}. The command

\begin{verbatim}
\set\{\textit{field}\}\{\textit{content}\}
\end{verbatim}

is a convenient shortcut for

\begin{verbatim}
\setFRMcontent\{\textit{field}\}\{\textit{content}\}
\end{verbatim}

It may be used in the content of the container to define the content of the individual fields. (See example in Section 1.4.)

3 \textbf{The Implementation}

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{formular}[\filedate \space \fileversion]
\RequirePackage{xspace}
\def\FRM@err\{\PackageError{formular}\}
\def\FRM@warn\{\PackageWarning{formular}\}
\newlength\frm@margin
\newlength\frm@baselineskip
\newbox\frm@namebx
\newbox\frm@contbx
\newcount\frm@cnt
\newcount\frm@lbound
\newif\iffrm@breakstyle
\newif\iffrm@ruledstyle
\end{verbatim}

3.1 Global defaults

The current and global settings are stored internally in \texttt{\frm@...}. The settings of individual field \texttt{\{\textit{field}\}} is stored in \texttt{\frm@\{\textit{field}\}@...}.

Auxiliary macros to define the \texttt{\textbackslash setFRM...} commands. Those commands, when called with optional argument containing a field id, must check whether the field is declared and then modify its settings.

\begin{verbatim}
\newcommand{\FRM@generatesetcommand}[1]%
\end{verbatim}

7
This is for the handling of \setFRM...style commands. The call \FRM@generatesetstylecommand{(a)}{(b)}{(c)} generates \setFRM(a)style which itself will let \iffrm(b)style to be \if(c). For individual fields, the \if... are stored in one macro only.

This is for the handling of \setFRM... counters and lengths. Since we do not want to waste registers, we store the contents of registers in macros (rather than in registers) for the individual fields.

The first two commands generated have no effect if called without the optional parameter.

Thickness of all rules

Vertical distance between baseline and an underlining rule
3.2 Defaults for one-line environments

Style of a one-line environment

\texttt{\FRM@generatesetstylecommand{plain}{ruled}{false}}
\texttt{\FRM@generatesetstylecommand{ruled}{ruled}{true}}

Default width of fields (not user accessible)
\texttt{\def\frm@width{0pt}}

3.3 Defaults for multi-line environments

Indentation of the content of a multi-line environment

\texttt{\FRM@generatesetlengthcommand{margin}}
\texttt{\setFRMmargin{5pt}}
\texttt{\baselineskip} within a multi-line environment
\texttt{\FRM@generatesetlengthcommand{baselineskip}}
\texttt{\setFRMbaselineskip{18pt}}

Style of a multi-line environment

\texttt{\FRM@generatesetstylecommand{break}{break}{true}}
\texttt{\FRM@generatesetstylecommand{inline}{break}{false}}

Default number of lines (not user accessible)
\texttt{\frm@lbound=0}

3.4 Font selection for the fields

Font shape of the content of all fields

\texttt{\FRM@generatesetcommand{fontencoding}}
\texttt{\FRM@generatesetcommand{fontsize}}
\texttt{\FRM@generatesetcommand{fontfamily}}
\texttt{\FRM@generatesetcommand{fontseries}}
\texttt{\FRM@generatesetcommand{fontshape}}
\texttt{\setFRMfontencoding{T1}}
\texttt{\setFRMfontsize{10}}
\texttt{\setFRMfontfamily{cmtt}}
\texttt{\setFRMfontseries{m}}
\texttt{\setFRMfontshape{n}}

Font shape of the description of one-line fields

\texttt{\FRM@generatesetcommand{dfontencoding}}
\texttt{\FRM@generatesetcommand{dfontsize}}
\texttt{\FRM@generatesetcommand{dfontfamily}}
\texttt{\FRM@generatesetcommand{dfontseries}}
\texttt{\FRM@generatesetcommand{dfontshape}}
\texttt{\setFRMdfontencoding{T1}}
\texttt{\setFRMdfontsize{6}}
\texttt{\setFRMdfontfamily{cmss}}
3.5 Auxiliary Macros for saving and restoring the settings

Store the current settings for a individual field. All \TeX registers are also stored in macros.

\begin{verbatim}
\newcommand{\Frm@storeappearance}[1]{%
\expandafter\let\csname frm@#1@content\endcsname \frm@content
\expandafter\let\csname frm@#1@description\endcsname \frm@description
\expandafter\let\csname iffrm@#1@breakstyle\endcsname \iffrm@breakstyle
\expandafter\let\csname iffrm@#1@ruledstyle\endcsname \iffrm@ruledstyle
\expandafter\let\csname frm@#1@width\endcsname \frm@width
\expandafter\let\csname frm@#1@rulewidth\endcsname \frm@rulewidth
\expandafter\let\csname frm@#1@rulesep\endcsname \frm@rulesep
\expandafter\edef\csname frm@#1@lbound\endcsname{\the\frm@lbound}
\expandafter\edef\csname frm@#1@baselineskip\endcsname{\the\frm@baselineskip}
\expandafter\edef\csname frm@#1@margin\endcsname{\the\frm@margin}
\expandafter\let\csname frm@#1@fontfamily\endcsname \frm@fontfamily
\expandafter\let\csname frm@#1@fontseries\endcsname \frm@fontseries
\expandafter\let\csname frm@#1@fontsize\endcsname \frm@fontsize
\expandafter\let\csname frm@#1@fontshape\endcsname \frm@fontshape
\expandafter\let\csname frm@#1@fontencoding\endcsname \frm@fontencoding
\expandafter\let\csname frm@#1@dfontfamily\endcsname \frm@dfontfamily
\expandafter\let\csname frm@#1@dfontseries\endcsname \frm@dfontseries
\expandafter\let\csname frm@#1@dfontsize\endcsname \frm@dfontsize
\expandafter\let\csname frm@#1@dfontencoding\endcsname \frm@dfontencoding
}
\end{verbatim}

Load the current settings from a individual field. All \TeX registers which were stored in macros are retransformed into registers.

\begin{verbatim}
\newcommand{\Frm@restoreappearance}[1]{%
\expandafter\let\expandafter\frm@content\csname frm@#1@content\endcsname
\expandafter\let\expandafter\frm@description\csname frm@#1@description\endcsname
\expandafter\let\expandafter\iffrm@breakstyle\csname iffrm@#1@breakstyle\endcsname
\expandafter\let\expandafter\iffrm@ruledstyle\csname iffrm@#1@ruledstyle\endcsname
\expandafter\let\expandafter\frm@width\csname frm@#1@width\endcsname
\expandafter\let\expandafter\frm@rulewidth\csname frm@#1@rulewidth\endcsname
\expandafter\let\expandafter\frm@rulesep\csname frm@#1@rulesep\endcsname
\expandafter\let\expandafter\frmlbound\csname frm@#1@lbound\endcsname
\expandafter\let\expandafter\frmbaselineskip\csname frm@#1@baselineskip\endcsname
\expandafter\let\expandafter\frmmargin\csname frm@#1@margin\endcsname
\expandafter\let\expandafter\frmfontfamily\csname frm@#1@fontfamily\endcsname
\expandafter\let\expandafter\frmfontseries\csname frm@#1@fontseries\endcsname
\expandafter\let\expandafter\frmfontsize\csname frm@#1@fontsize\endcsname
\expandafter\let\expandafter\frmfontencoding\csname frm@#1@fontencoding\endcsname
\expandafter\let\expandafter\frmdfontfamily\csname frm@#1@dfontfamily\endcsname
\expandafter\let\expandafter\frmdfontseries\csname frm@#1@dfontseries\endcsname
\expandafter\let\expandafter\frmdfontsize\csname frm@#1@dfontsize\endcsname
\expandafter\let\expandafter\frmdfontencoding\csname frm@#1@dfontencoding\endcsname
}
\end{verbatim}
3.6 One-line fields

\newFRMfield Parameter list:
#1 field id
#2 minimum field width
#3 description (optional)
#4 default content (optional)
We first must fiddle around with the two optional parameters. Then we set the default values (if any) and store all settings of the individual field.

\new@FRMfield

\let\new@FRMcontainerhook\@gobble
\def\new@FRMfieldspecials#1{\expandafter\def\csname set#1\endcsname##1{\setFRMcontent[#1]{##1}}}%
\expandafter\def\csname use#1\endcsname{\useFRMfield{#1}\relax}%

\useFRMfield Parameter list:
#1 field id
#2 field content (optional)

\newcommand{\useFRMfield}[1]{% \ifundefined{frm@#1@content} \FRM@err{cannot \string\renew... undeclared field ‘#1’}\new@FRMfield(#1)\}  
\} }%This macro does the actual typesetting job. We must open a group to keep changes to the current settings (via \FRM@storeappearance...) locally.
\def\useFRMfield[#1]{% \ifundefined{frm@#1@content} \FRM@err{FRMfield ‘#1’ is not declared.}\use@FRMfield[#1]}%
\def\use@FRMfield{\@ifnextchar\[%\]}%
\def\use@FRMfield@{\use@FRMfield\[#3\]}
\def\use@FRMfield@@{\use@FRMfield\[#3\]\[#4\]}%
The baseline of the final field is the baseline of the content and has depth extending to the underlining rule. This makes alignment of more than one field on the same line less complex.

Enclose `\usebox` in braces. This circumvents a bug(?) in `pdftex.def` v0.03k of standard `graphicx` and `color` packages: Compiling with `pdflatex` would break here.

```
\leavevmode
\vtop to \frm@rulesep{% 
  \halign{\hfil##\hfil\cr Enclose \usebox in braces. This circumvents a bug(?) in \texttt{pdftex.def} v0.03k of standard \texttt{graphicx} and \texttt{color} packages: Compiling with \texttt{pdflatex} would break here.

\newFRMenvironment Parameter list:
#1 environment id
#2 description
#3 number of lines
```

```
\newcommand{\newFRMenvironment}[3]{% 
  \def\frm@description{#2}% 
  \frm@lbound=#3\relax 
  \FRM@storeappearance{env@#1} 
  \newenvironment{#1}{% 
    \FRM@restoreappearance{env@#1} 
    \@ifnextchar[% 
      \FRM@openenvironment 
      \FRM@openenvironment[\the\frm@lbound]}% 
    \FRM@closeenvironment% 
  }{% 
    \def\frm@openenvironment[\the\frm@lbound]}% 
  \@ifnextchar[ 
    \FRM@openenvironment 
    \FRM@openenvironment[\the\frm@lbound]}% 
  \FRM@closeenvironment% 
  \% 
  \}% 

\define the \texttt{begin{}} part of the \texttt{FRMenvironment.}
```

```
\def\FRM@openenvironment[\the\frm@lbound]=% 
\frm@lbound=\the\frm@lbound

13
```
Set the description of the environment. If the current style is a break style, then extend the box to \hsize. The top line gets a “strut” of height \frm@baselineskip minus \frm@rulesep. The lowest line will have depth \frm@rulesep. Hence multiple environments fit smoothly to each other.

\setbox\frm@namebx\hbox
\iffrm@breakstyle to\hsize\fi
{\frm@description \enspace
\skip0=\frm@baselineskip \advance\skip0-\frm@rulesep
\vrule width0pt height \skip0\hfil}\

Start the box containing the body.
\setbox\frm@contbx=\vtop\bgroup
\advance\hsize-2\frm@margin
Leave out the horizontal space needed for the description. If the description extends the whole line, allow for a line break after this.
\hskip-\frm@margin \hskip\wd\frm@namebx \hskip-\frm@margin\penalty0\relax \hskip2\frm@margin \FRM@selectfont}

Define the \end{} part of the environment.
\newcommand{\FRM@closeenvironment}{%
Close the content box and count the number of lines below the baseline. Set the box with depth zero.
\par\egroup
\frm@cnt=\dp\frm@contbx
\dimen0=\frm@baselineskip
\divide\frm@cnt\dimen0\relax
\leavevmode\rlap{\dp\frm@contbx0pt \kern\frm@margin \usebox\frm@contbx}%

Now set the rules box. It must be shifted down, since the rules should be below the actual baseline.
\raise-\frm@rulesep\vtop{%
The first line contains the description and a rule (if it does not extend the whole line). The description is shifted back to be aligned with the baseline.
\hbox to\hsize{%
\dp\frm@namebx0pt%
\raise\frm@rulesep\hbox{\usebox{\frm@namebx}}\leaders \hrule
height \frm@rulewidth \hfill}%

If there is no break style, then the first line contains already one rule and the line bound must be decreased by one.
\iffrm@breakstyle\else \advance\frm@lbound-1\fi\relax
If the bound on the number of lines is exceeded then issue a warning.
\ifnum\frm@cnt>\frm@lbound\relax
In each case, \texttt{frm@cnt} contains the number of remaining rules. Plot them now. The depth of the final box is determined by the lowest rule.

\begin{loop}
\ifnum\frm@cnt>0\relax
\advance\frm@cnt-1\relax
\kern\frm@baselineskip \kern-\frm@rulewidth
\hrule height \frm@rulewidth\relax
\repeat
\endloop

\section{3.8 Containers}

\texttt{\textbackslash newFRMcontainer} Parameter list:
\begin{itemize}
\item \texttt{#1} container id
\item \texttt{#2} init code
\item \texttt{#3} apply code
\end{itemize}

We store the current settings for restore at the beginning of the init code. Moreover, we use the hook to let \texttt{\textbackslash newFRMfield} also define the \texttt{\set...} shortcut afterwards.

\longdef\newFRMcontainer#1#2#3{%
\FRM@storeappearance{con@#1}%
\newenvironment{#1}{%\let\new@FRMcontainerhook=\new@FRMfieldspecials
\FRM@restoreappearance{con@#1}{#2}{#3}}{#3}
\}

Seems to be a convention:

\endinput

\section*{Index}

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

\begin{tabular}{lll}
N & R & S \\
\new@FRMfield & \renewFRMfield & \setFRMcontent \\
\newFRMcontainer 6, 278 & \setFRMdescription & \setFRMfontencoding \\
\newFRMenvironment & 6, 225 & \setFRMfontfamily \\
\newFRMfield 5, 161 & \setFRMfontsize & \setFRMfontshape \\
\end{tabular}
Change History

v1.0a
\useFRMfield: Circumvent problem with pdflatex and graph-
icx/color.sty ............... 13