The `fp` package

Author: Michael Mehlich*

Contributions by: Denis Girou
Acknowledgment to: Denis Girou, Miroslav Balda, Ricardo Sanchez Carmenes
Documentation: Peter Wang (based on the original ReadMe.txt file, added examples to show syntax)

E-mails: mmehlich@semdesigns.com; michael@mehlich.com; shuodaowang@gmail.com .

January 14, 2019

Abstract

Fixed point arithmetic for TEX with numbers ranging from
−9999999999999999.99999999999999999
to +9999999999999999.99999999999999999

Contents

1 Usage:

- LaTeX 2_ε:
  `\usepackage[<options>]fp`
  where the following options are known:

  `[nomessages]`: don't print messages about the functions that are just computed.

  `[debug]`: print debug messages (mainly for `\FPupn`).

- LaTeX2.09:
  `include lfp.sty` in the document preamble, i.e.
  `\documentstyle[...,lfp,...]...`

- TeX:
  `\input fp.tex`

- MsDos/Windows Users:
  It may be necessary to rename some files such that they just have a length of eight characters (plus a three character suffix). The following renaming examples works for emtex:

Original name	Name for emtex
`defpattern.sty`	`defpaern.sty`
`fp-addons.sty`	`fp-adons.sty`
`fp-random.sty`	`fp-radom.sty`

2 Basic functions:

- **\FPset#1#2**: Defines a variable that you can later print.
- **\FPprint#1**: Prints the value of a variable.

  Example:

```
\FPset\x{2} %sets x=2
$x=\x$.\\ %prints x=2          x = 2.
$x=\FPprint\x$.\\              x = 2.
x=\x.\\                        x=2.
x=\FPprint\x .                 x=2.
```

- The following commands are very straightforward:
  `binary and unary operations:`

```
\FPadd#1#2#3 % #1 := #2+#3
\FPdiv#1#2#3 % #1 := #2/#3
\FPmul#1#2#3 % #1 := #2*#3
\FPsub#1#2#3 % #1 := #2-#3
\FPabs#1#2 % #1 := abs(#2)
```

```
\FPneg#1#2 % #1 := -#2
\FPmin#1#2#3 % #1 = min(#2,#3)
\FPmax#1#2#3 % #1 = max(#2,#3)
```

binary and unary relations:

```
\FPiflt#1#2...\else...\fi % #1 < #2 ?
\FPifeq#1#2...\else...\fi % #1 = #2 ?
\FPifgt#1#2...\else...\fi % #1 > #2 ?
\FPifneg#1 ...\else...\fi % #1 <  0 ?
\FPifpos#1 ...\else...\fi % #1 >= 0 ?
\FPifzero#1...\else...\fi % #1 =  0 ?
\FPifint#1 ...\else...\fi % #1 is integer ?
%repeat last test
\ifFPtest  ...\else...\fi % repeat last test
```

Trigonometric functions (Note: only accepts float numbers for the input variables):

```
\FPpi  % 3.141592653589793238
\FPsin#1#2 % #1 := sin(#2)
\FPcos#1#2  % #1 := cos(#2)
\FPsincos#1#2#3 % #1 := sin(#3), #2 := cos(#3)
\FPtan#1#2 % #1 := tan(#2)
\FPcot#1#2  % #1 := cot(#2)
\FPtancot#1#2#3  % #1 := tan(#3), #2 := cot(#3)
\FParcsin#1#2 % #1 := arcsin(#2)
\FParccos#1#2      % #1 := arccos(#2)
\FParcsincos#1#2#3   % #1 := arcsin(#3), #2 := arccos(#3)
\FParctan#1#2  % #1 := arctan(#2)
\FParccot#1#2  % #1 := arccot(#2)
\FParctancot#1#2#3 % #1 := arctan(#3), #2 := arccot(#3)
```

Examples:

```
\FPset\x{-1}
\FPset\y{2}
\FPadd\xay\x\y
\FPmin\xoy\x\y
$x=\x, y=\y$ \\
\FPifgt\xay\y $x+y>y$.
\else $x+y<y$.\fi \\ \\ \\
The result $x+y$
\FPifint\xay  is an integer.
\else is not an integer.
\fi\\ \\
$\min(x,y)=\xoy$.
```

$x = -1, y = 2$

$x + y < y$.

The result $x + y$ is an integer.

$\min(x, y) = -1$.

- Solving equations:

```
\FPlsolve#1#2#3
% #1 := x with #2*x+#3=0
\FPqsolve#1#2#3#4#5
% #1,#2 := x with #3*x^2+#4*x+#5 = 0
\FPcsolve#1#2#3#4#5#6#7
% #1,#2,#3 := x with #4*x^3+#5*x^2+#6*x+#7 = 0
\FPqqsolve#1#2#3#4#5#6#7#8#9
% #1,#2,#3,#4 := x with #5*x^4+#6*x^3+#7*x^2+#8*x+#9 = 0
```

Example:

```
\FPset\ca{-4}
\FPset\cb{2}
\FPlsolve\res\ca\cb
The root for
$\ca x+\cb=0$ is\\
$x=\res$.
```

The root for $-4x + 2 = 0$ is
$x = 0.500000000000000000$.

- Evaluate expressions:

  ```
  \FPeval#1#2
  % #1 := eval(#2) where eval evaluates the expression #2
  ```

  Example:

  ```
  \edef\x{11}
  \FPeval\resulta{\x/2}
  \FPeval{resultb}{clip(neg(x)/2)}
  resulta = \resulta .\\
  resultb = \resultb .\\\\
  \FPeval\resulta{round(resulta:3)}
  round(resulta:3) = \resulta.
  ```

  resulta = 5.500000000000000000.
  resultb = -5.5.
  round(resulta:3) = 5.500.

  Attentions:

  – The `#1` variable can be written as either "`\resulta`" or "`{resulta}`", but not "`\resulta{}`" in the above example.

  – When referring to variables in the expression `#2`, one can use "`\x`" or "`\x{}`", or simply "`x`" in the above example.

  – The unary prefix operation "`-`" is not known, therefore one should use the function `neg()` instead.

  – All the results from `\FPeval` are real numbers so rounding may be necessary.

  Known operations:

+	-	*	/	abs	neg
pow	root	exp	ln	min	max
e	pi				
round	trunc	clip			
sin	cos	tan	cot		
arcsin	arccos	arctan	arccot		

  Most of the operations are self-explanatory. A few notes here:

`pow(#1,#2)`	returns #2 to the power of #1
`root(#1,#2)`	returns the $\#1^{th}$ root of #2
`exp(#1)`	returns e (defined below) to the power of #1
`ln(#1)`	returns ln(#1) (base e)
`min(#1,#2)`	returns minimum of #1 and #2
`e`	returns $e = 2.718281828459045235$
`pi`	returns $\pi = 3.141592653589793238$
`round(#1:#2)`	round #1 to #2 decimal places
`trunc(#1:#2)`	truncate #1 to #2 decimal places
`clip(#1)`	remove all the trailing "0"s in #1
`sin(#1)`	sin of #1 in rad. Similarly for others
`arcsin(#1)`	arcsin of #1

- Evaluate upn-expressions:

```
\FPupn#1#2  % #1 := eval(#2) where eval evaluates the
upn-expression #2
```

Known operations:
+,add,-,sub,*,mul,/,div,abs,neg,min,max, round,trunc,clip,e,exp,ln,pow,root,pi,sin,cos, sincos,tan,cot,tancot,arcsin,arccos,arcsincos, arctan,arccot,arctancot,pop,swap,copy

where

`pop`: removes the top element
`swap`: exchanges the first two elements
`copy`: copies the top element

Examples:

```
\FPupn\result{17 2.5 + 17.5 - 2 1 + * 2 swap /}
is equivalent to
\result := ((17.5 - (17 + 2.5)) * (2 + 1)) / 2
and evaluates to
\def\result{-3.000000000000000000}
Afterwards the macro call
\FPupn\result{\result{} -1 * 0.2 + sin 2 round}
^^ the "{}" is necessary!
is equivalent to
\result := round_2(sin((\result * -1) + 0.2))
and evaluates to
\def\result{-0.06}
Example 2:
As "result" is an abbreviation of "\result{}" you may
write
\FPupn{result}{17 2.5 + 17.5 - 2 1 + * 2 swap /}
and
```

```
\FPupn{result}{result -1 * 0.2 + sin 2 round}
instead leading to the same results.
```

This is even true for other macro names using e.g. "`x`" for "`\x{}`" and so on. But be careful with it. We may introduce new constants in further versions overwriting these abbreviations.

3 Known bugs:

- Does not work with multido.sty/multido.tex

  Reason:
  multido uses the same macro names `\FPadd` and `\FPsub`

  Recommended Solution:
  Patch `multido.tex`, i.e. apply the following substitutions:
  `FPadd -> mdo@FPadd`
  `FPsub -> mdo@FPsub`

- Incompatibility with french style of babel.
  This only affects macros using the colon (:)

  Recommended Solution:
  Load the fp-package before babel with french style
  Other Possible Solution:
  Use `\catcode`\:=12` after loading babel with french style

- Others:
  Currently not known, but, though we do not, we could give a warranty of their existence ...

4 Copying:

- Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice, the above history with your modifications added, and this permission notice appear in all copies and modified versions.

- The copyright holder disclaim all warranties with regard to this software, including all implied warranties of merchantability and fitness, in no event shall the copyright holder be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.