CONTEXT

Missing (For Generic Use)

group: CONTEXT Support Macros

version: 1997.01.04

date: 1997 July 25

author: Hans Hagen

Some support modules are more or less independant. This module, which is not part of plain CONTEXT, provides the missing macros and declarations of registers.

`\ifnocontext..`

First we take care of redundant defining. The next set of macros are a bit complicated by the fact that Plain TEX defines the `\new`–macros as being outer. Furthermore nested `\if`'s can get us into trouble.

```
1  \def\definecontextobject%
     {\iftrue}

2  \def\gobblecontextobject%
     {\setbox0=\hbox
        \bgroup
        \long\def\gobblecontextobject##1\fi{\egroup}%
        \expandafter\gobblecontextobject\string}

3  \def\ifnocontextobject#1\do%
     {\ifx#1\undefined
        \let\next=\definecontextobject
      \else
        %\writestatus{system}{beware of conflicting \string#1}%
        \let\next=\gobblecontextobject
      \fi
      \next}
```

`\writestatus`

We start each module with a message. Normally the output is formatted, but here we keep things simple.

```
4  \ifnocontextobject \writestatus \do

5    \def\writestatus#1#2%
       {\immediate\write16{#1 : #2}}

6  \fi
```

Lets see if it works.

```
7  \writestatus{loading}{Context Support Macros / Missing}
```

`\protect`
`\unprotect`

Next we present a poor mans alternative for `\protect` and `\unprotect`, two commands that enable us to use the characters `@`, `!` and `?` in macro names.

```
8  \ifnocontextobject \protect \do

9    \let\protect=\relax

10 \fi

11 \ifnocontextobject \unprotect \do

12   \def\unprotect%
       {\catcode`@=11
        \catcode`!=11
        \catcode`?=11
        \let\normalprotect=\protect
        \edef\protect%
          {\catcode`@=\the\catcode`@\relax
           \catcode`!=\the\catcode`!\relax
        \catcode`?=\the\catcode`?\relax
        \let\protect=\normalprotect}}
```

13
```
\fi
```

We start using this one it at once.

14
```
\unprotect
```

\scratch... \if... \next... We need some scratch registers. Users are free to use them, but can never be sure of their value once another macro is called. We only allocate things when they are yet undefined. This way we can't mess up other macro packages, but of course previous definitions can mess up our modules.

15
```
\ifnocontextobject \scratchcounter       \do \newcount  \scratchcounter  \fi
\ifnocontextobject \scratchdimen         \do \newdimen  \scratchdimen    \fi
\ifnocontextobject \scratchskip          \do \newskip   \scratchskip     \fi
\ifnocontextobject \scratchmuskip        \do \newmuskip \scratchmuskip   \fi
\ifnocontextobject \scratchbox           \do \newbox    \scratchbox      \fi
\ifnocontextobject \scratchread          \do \newread   \scratchread     \fi
\ifnocontextobject \scratchwrite         \do \newwrite  \scratchread     \fi
```

16
```
\ifnocontextobject \nextbox              \do \newbox    \nextbox         \fi
```

17
```
\ifnocontextobject \nextdepth            \do \newdimen  \nextdepth       \fi
```

18
```
\ifnocontextobject \ifCONTEXTtrue        \do \newif\ifCONTEXT            \fi
\ifnocontextobject \ifdonetrue           \do \newif\ifdone               \fi
\ifnocontextobject \ifeightbitcharacters \do \newif\ifeightbitcharacters \fi
```

\@@... We use symbolic name for ⟨*catcodes*⟩. They can only be used when we are in unprotected state.

19
```
\ifnocontextobject \@@escape             \do \chardef\@@escape     =  0  \fi
\ifnocontextobject \@@begingroup         \do \chardef\@@begingroup =  1  \fi
\ifnocontextobject \@@endgroup           \do \chardef\@@endgroup   =  2  \fi
\ifnocontextobject \@@ignore             \do \chardef\@@ignore     =  9  \fi
\ifnocontextobject \@@space              \do \chardef\@@space      = 10  \fi
\ifnocontextobject \@@letter             \do \chardef\@@letter     = 11  \fi
\ifnocontextobject \@@other              \do \chardef\@@other      = 12  \fi
\ifnocontextobject \@@active             \do \chardef\@@active     = 13  \fi
\ifnocontextobject \@@comment            \do \chardef\@@comment    = 14  \fi
```

\everyline \EveryLine \EveryPar In CONTEXT we use `\everypar` for special purposes and provide `\EveryPar` as an alternative. The same goes for `\everyline` and `\EveryLine`.

20
```
\ifnocontextobject \everyline            \do \newtoks\everyline          \fi
\ifnocontextobject \EveryPar             \do \let\EveryPar  = \everypar  \fi
\ifnocontextobject \EveryLine            \do \let\EveryLine = \everyline \fi
```

\!!... We reserve ourselves some scratch strings (i.e. macros).

21
```
\ifnocontextobject \!!stringa            \do \def\!!stringa     {}       \fi
\ifnocontextobject \!!stringb            \do \def\!!stringb     {}       \fi
\ifnocontextobject \!!stringc            \do \def\!!stringc     {}       \fi
\ifnocontextobject \!!stringd            \do \def\!!stringd     {}       \fi
```

`\!!...` The next set of definitions speed up processing a bit. Furthermore it saves memory.

```
\ifnocontextobject \!!zeropoint        \do \def\!!zeropoint   {0pt}    \fi
\ifnocontextobject \!!tenthousand      \do \def\!!tenthousand {10000}  \fi

\ifnocontextobject \!!width            \do \def\!!width       {width}  \fi
\ifnocontextobject \!!height           \do \def\!!height      {height} \fi
\ifnocontextobject \!!depth            \do \def\!!depth       {depth}  \fi

\ifnocontextobject \!!plus             \do \def\!!plus        {plus}   \fi
\ifnocontextobject \!!minus            \do \def\!!minus       {minus}  \fi
```

`\smashbox` The system modules offer a range of smashing macros, of which we only copied `\smashbox`.

```
\ifnocontextobject \smashbox \do

  \def\smashbox#1%
    {\wd#1=\!!zeropoint
     \ht#1=\!!zeropoint
     \dp#1=\!!zeropoint}

\fi
```

`\dowithnextbox` Also without further comment, we introduce a macro that gets the next box and does something usefull with it. Because the `\afterassignment` is executed inside the box, we have to use a `\aftergroup` too.

```
\ifnocontextobject \dowithnextbox \do

  \def\dowithnextbox#1%
    {\def\dodowithnextbox{#1}%
     \afterassignment\dododowithnextbox
     \setbox\nextbox}

  \def\dododowithnextbox%
    {\aftergroup\dodowithnextbox}

\fi
```

`\setvalue` `\getvalue` The next two macros expand their argument to `\argument`. The first one is used to define macro's the second one executes them.

```
\ifnocontextobject \setvalue \do

  \def\setvalue#1{\expandafter\def\csname#1\endcsname}
  \def\getvalue#1{\csname#1\endcsname}

\fi
```

`\protected` The next command can be used as prefixed for commands that need protection during tests and writing to files. This is a very CONTEXT specific one.

```
\ifnocontextobject \unexpanded \do

  \let\unexpanded=\relax

\fi
```

`\convertargu..` The original one offers a bit more, like global assignment, the the next implementation is however a bit more byte saving.

```
\ifnocontextobject \convertargument \do

  \def\doconvertargument#1>{}

  \long\def\convertargument#1\to#2%
    {\long\def\convertedargument{#1}%
     \edef#2{\expandafter\doconvertargument\meaning\convertedargument}}

\fi
```

`\forgetall` Sometimes we have to disable interference of whatever kind of skips and mechanisms. The next macro resets some.

```
\ifnocontextobject \forgetall \do

  \def\forgetall%
    {\parskip\!!zeropoint
     \leftskip\!!zeropoint
     \parindent\!!zeropoint
     \everypar{}}

\fi
```

`\withoutpt` TEX lacks some real datastructure. We can however use ⟨*dimensions*⟩. This kind of trickery is needed when we want TEX to communicate with the outside world (by means of `\specials`).

```
\ifnocontextobject \withoutpt \do

  {\catcode`\.=\@@other
   \catcode`\p=\@@other
   \catcode`\t=\@@other
   \gdef\WITHOUTPT#1pt{#1}}

  \def\withoutpt#1%
    {\expandafter\WITHOUTPT#1}

  \def\ScaledPointsToBigPoints#1#2%
    {\scratchdimen=#1sp\relax
     \scratchdimen=.996264\scratchdimen
     \edef#2{\withoutpt{\the\scratchdimen}}}

\fi
```

`\doprocessfile` This macro takes three arguments: the file number, the filename and a macro that handles the content of a read line.

```
\ifnocontextobject \doprocessfile \do

  \def\doprocessfile#1#2#3%
    {\openin#1=#2\relax
     \def\doprocessline%
       {\ifeof#1%
          \def\doprocessline{\closein#1}%
        \else
          \read#1 to \fileline
        #3\relax
      \fi
      \doprocessline}%
    \doprocessline}
```

```
52  \fi
```

\uncatcodesp.. This one is taken from the TEX book. The CONTEXT alternative is a bit different, but we hope this one works here.

```
53  \ifnocontextobject \uncatcodespecials \do
```

```
54    \def\uncatcodespecials%
        {\def\do##1{\catcode`##1=12 }\dospecials}
```

```
55  \fi
```

That's it. Please forget this junk and take a look at how it should be done.

```
56  \protect
```

Missing (For Generic Use)