

# The ldcsetup package\*

DPC, FMi

2006/08/21

## Abstract

Basic code shared by several other packages currently under development.

```
1 (*package)
\CodeStart is in effect here so we better stop it again.
2 \CodeStop
3 \RequirePackage{keyval,calc}
4 \makeatletter
```

## 1 Ignoring White Space

Within a package, or in a document preamble, you can use `\IgnoreWhiteSpace` this makes white space ignored, and `@` a letter, and `~` a space character. It lasts until the end of the current package or preamble.

Puts an ignored space at the end of the line, so `\ '`  does the right thing. (Perhaps should instead just change catcode of `^M`)

```
5 \def\IgnoreWhiteSpace{%
6   \edef\@tempa{%
7     \catcode'\noexpand\~=\the\catcode'\~\relax
8     \catcode'\noexpand\ =\the\catcode'\ \relax
9     \catcode'\noexpand\^I=\the\catcode'\^I\relax
10    \catcode'\noexpand@\=\the\catcode'\@\relax
11    \endlinechar=\the\endlinechar\relax
12  }%
13  \ifx\@currname\@empty
14    \expandafter\AtBeginDocument\expandafter{\@tempa}%
15  \else
16    \expandafter\AtEndOfPackage\expandafter{\@tempa}%
17  \fi
18  \catcode'\~ =10\relax
19  \catcode'\ =9\relax
20  \catcode'\^I=9\relax
21  \makeatletter
22  \endlinechar=' %
23  \relax
24 }
```

---

\*This file has version number 574, last revised 2006/08/21.

The new syntax requires : and \_ to be letters; instead of putting that into `\IgnoreWhiteSpace` we make a new command which helps to gradually change packages from old to new syntax.

```

25 \def\InternalSyntaxOn{%
26   \edef\@tempa{%
27     \catcode'\noexpand\~=\the\catcode'\~\relax
28     \catcode'\noexpand\ =\the\catcode'\ \relax
29     \catcode'\noexpand\^^I=\the\catcode'\^^I\relax
30     \catcode'\noexpand\@=\the\catcode'\@\relax
31     \catcode'\noexpand\:=\the\catcode'\:\relax
32     \catcode'\noexpand\_=\the\catcode'\_\relax
33     \endlinechar=\the\endlinechar\relax
34 %Mi fix is not properly reset
35     \endlinechar=13\relax
36   }%
37 %%% ???
38 % this needs further work, don't think it is working
39   \ifx\InternalSyntaxOff\relax
40     \expandafter\def\expandafter\InternalSyntaxOff\expandafter
41 {\@tempa\let\InternalSyntaxOff\relax}%
42   \fi
43   \ifx\@currname\@empty
44     \expandafter\AtBeginDocument\expandafter{\@tempa}%
45   \else
46     \expandafter\AtEndOfPackage\expandafter{\@tempa}%
47   \fi
48   \catcode'\~ =10\relax
49   \catcode'\ =9\relax
50   \catcode'\^^I=9\relax
51   \makeatletter
52   \catcode'\_ =11\relax
53   \catcode'\: =11\relax
54   \endlinechar=' %
55   \relax
56 }
57 \let\InternalSyntaxOff\relax
    Do it for this package.
58 \IgnoreWhiteSpace

```

## 2 KV fixes

Allow<sup>1</sup> key names to start with \.

```

59 \def\define@key#1#2{%
60   \ifnextchar[
61     {\KV@def{#1}{\string#2}}
62     {\@namedef{KV@#1@\string#2}####1}}
63 \def\setkeys#1#2{%
64   \def\KV@prefix{KV@#1@\expandafter\string}%
65   \KV@do#2,\relax,}
    Make KV # safe.

```

---

<sup>1</sup>Not needed now?

```

66 \newtoks\KV@toks
67 \def\KV@@sp@c#1\@nil#2\relax#3{\KV@toks{#1}\edef#3{\the\KV@toks}}

    Generate error messages on missing ‘,’.
    More exactly if two ‘=’ appear after a key generate an error. The current KV
    just silently ignores everything after the second ‘=’.
68 \def\KV@equal{=}
69 \def\KV@split#1=#2=#3\relax{%
70   \KV@@sp@def\@tempa{#1}%
71   \def\@tempd{#3}%
72   \ifx\@tempa\@empty\else
73     \expandafter\let\expandafter\@tempc
74       \csname\KV@prefix\@tempa\endcsname
75     \ifx\@tempc\relax
76       \KV@error{\@tempa\space \expandafter\@gobbletwo\string\@undefined}\@eha
77     \else
78       \ifx\@tempd\@empty
79         \KV@default
80       \else
81         \KV@@sp@def\@tempb{#2}%
82         \ifx\@tempd\KV@equal
83           \expandafter\@tempc\expandafter{\@tempb}\relax
84         \else
85           \KV@error{Extra~==~ sign~ after~ ‘#1’}\KV@erry
86         \fi
87       \fi
88     \fi
89   \fi}

90 \def\KV@erry{\expandafter\KV@errx\meaning\@tempd\relax ignored\MessageBreak
91   missing~comma~in~‘\expandafter\strip@prefix\meaning\@tempb’~?}

92 \def\KV@errx#1>#2==\relax{%
93   ‘#2’ }

94 \def\KV@default{%
95   \expandafter\let\expandafter\@tempb
96     \csname\KV@prefix\@tempa @default\endcsname
97   \ifx\@tempb\relax
98     \KV@error{No value specified for \@tempa}\@eha
99   \else
100     \@tempb\relax
101   \fi}

    LATEX style error message.
102 \def\KV@error#1#2{\PackageError{keyval}{#1}{#2}}

```

Instead of doing a full KV parse, and evaluating all the keys, you might want to parse a parameter list, even for undefined keys, removing spaces, splitting up the ‘,’ and ‘=’ and resolving cases where no value is supplied. This is used in `\DeclareGenericFunction` which *defines* a set of keys via such a KV parse.

A setting of , `key = value` , will result in `\KV@elt{key}{value}` being added to the list, a setting of , `key` , will result in `\KV@default@elt{key}` being added. At the end of the parse the list is executed. No keys are checked at this

stage (so no csnames are used up) the two `\KV...@elt` commands must be defined as appropriate.

```

103 \def\KV@parse#1{
104   \begingroup

      Locally fudge \KV@@def to just add to a token register rather than make a
      definition.
105 \def\KV@sp@c##1\@nil##2\relax##3{\addto@hook##3{\{##1\}}}

      Locally fudge \KV@split to just add to a token register rather than execute
      the code for the found keys.
106 \def\KV@split##1=##2=##3\relax{%
107   \def\@tempd{\##3}%
108   \expandafter\ifx\expandafter=\@firstofone##1=\else
109     \ifx\@tempd\@empty
110       \addto@hook\KV@toks\KV@default@elt
111       \KV@@sp@def\KV@toks{##1}%
112     \else
113       \ifx\@tempd\KV@equal
114         \addto@hook\KV@toks\KV@elt
115         \KV@@sp@def\KV@toks{##1}%
116         \KV@@sp@def\KV@toks{##2}%
117       \else
118         \KV@err{Extra '=' after '{##1'}\@ehd
119       \fi
120     \fi
121   \fi}

      Initialise
122   \KV@toks{}

      Do the parse
123   \KV@do#1,\relax,

      Evaluate the token register outside the group.
124   \expandafter
125   \endgroup
126   \the\KV@toks}

```

### 3 Calc fixes

Count Register assignments via calc. Here we also provide some global versions since a normal global prefix won't have any effect.

```

127 \let\SetInternalCounter\calc@assign@count
128 \def\GsetInternalCounter#1{\SetInternalCounter{\global#1}}
129 \def\gsetlength#1{\setlength{\global#1}}

```

### 4 Misc code that is best collected in one place

This section will receive odd code that would be of potential use to other packages and should eventually made it to the kernel (perhaps).

`\UndeclareRobustCommand` Remove a robust command from memory if it isn't used any longer. Of course this doesn't free the hash table but at least the space gets reclaimed.

```
130 \def\UndeclareRobustCommand#1{%  
131   \global\let#1\undefined  
132   \global\expandafter  
133     \let\csname\expandafter\@gobble\string#1~\endcsname\undefined  
134 }  
  
135 \end{package}
```