

The dozenal Package, v5.1

Donald P. Goodman III

January 22, 2015

Abstract

The `dozenal` package provides some simple mechanisms for working with the dozenal (duodecimal or “base 12”) numerical system. It redefines all basic \LaTeX counters, provides a command for converting arbitrary decimal numbers into dozenal, and provides new, real Metafont characters for ten and eleven, though the commands for producing them can be redefined to produce any figure. As of v2.0, it also includes Type 1 versions of the fonts, used automatically whenever T1 encoding is selected for the document. This package uses the `\basexii` algorithm by David Kastrup.

Contents

1	Introduction	1
2	Usage	2
3	The Fonts	4
4	Implementation	5

1 Introduction

While most would probably call it at best overoptimistic and at worst foolish, some people (the author included) do still find themselves attracted to the dozenal (base-twelve) system. These people, however, have been pretty hard up¹ in the \LaTeX world. There is no package file available which produces dozenal counters, like page and chapter numbers, nor were there *any* (I made a pretty diligent search) dozenal characters for ten and eleven, leaving dozenalists forced to use such makeshift ugliness as the “X/E” or “T/E” or “*/#” or whatever other standard they decided to use. While this sort of thing may be acceptable in ASCII, it’s absolutely unacceptable in a beautiful, typeset document.

¹This is an Americanism for “out of luck” or “in difficult circumstances,” for those who do not know.

Enter the `dozenal` package. This package automates all the messiness of being a dozenalist and using L^AT_EX. It redefines all the counters (though you'll have to redefine them yourself if you're using your own), provides an algorithm (generously donated by the intrepid David Kastrup) for converting arbitrary positive whole numbers into dozenal (this is eT_EX, but all modern distributions will compile that), and finally, it includes original dozenal characters, specifically designed to blend in well with Knuth's Computer Modern fonts, though they should do fine with the more common body fonts, as well.

This document was typeset in accordance with the L^AT_EX DOCSTRIP utility.

2 Usage

The `dozenal` package provides four new commands (though I can only take credit for two of them). The first, and by far the most important given the purpose and content of this package, is `\basexii`. This is a very simple command which takes the following structure:

```
\basexii{<number>}{<ten symbol>}{<eleven symbol>}
```

What the above means is that the command is `\basexii` and it takes three mandatory arguments: first, the number to be converted into dozenal; second, the symbol that should be used for ten; and third, the symbol that should be used for eleven. This number should be positive and whole; that is, it should be zero or higher, and it should not contain a fractional part. T_EX is a typesetting program, after all; if you want a robust decimal to dozenal converter, there are many options that any dozenalists caring enough to use this package will already know about.

This `\basexii` algorithm was produced by David Kastrup, well known and admired in the T_EX world for his many useful packages and other contributions. He posted this algorithm on comp.text.tex; it is included here with his kind and generous permission.

That one would want to use the same ten and eleven symbols throughout a document seems a reasonable assumption; therefore, I have provided a simplified version of the `\basexii` command, `\dozens`. `\dozens` takes only a single argument, the number to be converted; the ten and eleven symbols used are those produced by the commands `\x` and `\e`, to which we'll get in a moment.

`\x` and `\e` are the commands used to quickly and easily access the symbols for ten and eleven without having to use active characters (the T_EX gurus will know what that means; if you don't know, that's okay; you don't need to to use this package effectively). In any case, `\x` and `\e` default to using the special dozenal characters that are part of this package; they could be easily redefined if for some reason you don't like the Pitman characters (which this package, and the Dozenal Society of Great Britain, prefer), in the following manner:

```
\renewcommand\x{X}
```

Or whichever characters you like to use. If you prefer the Dozenal Society of America's proposed characters (a stylized X and E), then this package will disappoint

you. May I suggest χ and ξ as a stopgap while you locate or produce real characters of your own? Sorry; I'm an American myself, but I much prefer the Pitman characters for a variety of reasons (feel free to email me if you care), and creating fonts in METAFONT, even small and inconsequential ones like this, is too much work for characters that I don't even like.

The `dozenal` package also redefines all the standard L^AT_EX counters, such as `section` and `enumii`. If you've defined your own counters, you'll need to define them yourself; however, this is an easy matter:

```
\renewcommand\thecounter{\basexii{\arabic{counter}}{\x}{\e}}
```

For example. Of course, you can fill in the `\x` and `\e` with whatever you want (though it would make more sense to simply redefine `\x` and `\e`, so that all the counters would use the same characters), or you could use the `\dozens` command instead. Whatever your pleasure might be.

nocounters The `nocounters` option to the package prevents the redefinition of these counters. The effect of this is that the commands of the package (`\basexii`, `\dozens`, etc.) are made available, but all the counters will still be in decimal. This permits using dozenal characters in an otherwise decimal document; it also proves useful in dozenal document in which these counters are undefined (e.g., `minimal`).

typeone The `typeone` option forces `dozenal` to provide Postscript Type 1 fonts rather than METAFONT bitmaps to T_EX. Both of these are produced from the same font files, though, so the difference is very slight.

\basex Finally, as of v5.0, we can convert numbers back to decimal from dozenal, if we wish. We do this with the `\basex` macro, which takes a single argument, which is the dozenal number you wish to convert to decimal. This is subject to some pretty harsh restrictions, however. First, the only tokens allowed in the number are 0–9, `\x`, and `\e`. Second, these will only work with *unexpanded* `\x` and `\e`. If you don't know what this means, then good for you; T_EX programmers will sympathize.

To illustrate these limitations, let's define a new counter and dozenize it. Here, we define the counter and give it a nice value which will ensure that its dozenal value will have an `\e` in it:

```
\newcounter{testcount}\setcounter{testcount}{47}
```

In dozenal, of course, "47" is "3 ξ ." Now, let's redefine that counter so that its results will be dozenal:

```
\renewcommand\thetestcount{\basexii{\value{testcount}}{\x}{\e}}
```

Now we get to do lovely things like the following:

```
\thetestcount = 3 $\xi$ 
```

Now we can try to get that number in decimal with `\basex`. But don't try it; `\basex{\thetestcount}` doesn't work because the `\x` and `\e` are already expanded. Instead, use L^AT_EX's built-in functions for chores like this:

`\arabic{testcount} = 47`

On the other hand, if you have an actual string you want converted, you can send it directly to `\basex` without worrying about expansion, because through much trial and error and banging head against wall, the expansion issues have already been resolved:

`\basex{3\e} = 47`

So `\basex` is of limited utility, but it's a nice tool to add to the box.

3 The Fonts







The fonts provided by the dozenal package are essentially complete fonts which contain only the Pitman dozenal characters; these are 7 for ten and 8 for eleven. These characters are designed to blend well with the Computer Modern fonts; they work passable well with Times-type fonts and with kpfonts, and possibly with others.

The characters also come in all the appropriate shapes and sizes; a few examples follow.

	Roman	<i>Italic</i>	Boldface
Footnotesize	7 8	<i>7 8</i>	7 8
Normalsize	7 8	<i>7 8</i>	7 8
LARGE	7 8	<i>7 8</i>	7 8
Huge	7 8	<i>7 8</i>	7 8

They will work in paragraph or math mode without distinction.

As of v4.0, `dozenal` also includes fonts for tally marks specifically designed for use in the dozenal base. In many European countries tallies are kept in a very similar way; this font demonstrates a way that such tally marks can be made consistent as well as dozenal.

1	2	3	4	5	6
					

`\tally` These are accessed by the `\tally` command, which takes one argument: the number, 1–6, which you want to put in tallies. Entering “X” or “E” will yield “7” or “8” respectively. Other characters will produce nothing. Any character produced by `\tally` is a METAFONT bitmap, not an outline, even if your document is otherwise producing outlines.

The fonts are called `dozch`, if for some reason direct access to them is needed. The Type 1 versions are prefixed `fdz`; e.g., `fdzr8a`.

4 Implementation

First, we name the package that we provide. This is not exactly the most difficult part of the code.

```
1 \ProvidesPackage{dozenal}[2012/05/13 v4.0 support for dozenal fonts]
```

Now we need to make sure that we have fixltx2e loaded, so that the `\TextorMath` magic will work.

```
2 \RequirePackage{fixltx2e}
```

Now we ensure that `ifpdf` is loaded, so that we can test for pdf or dvi modes.

```
3 \RequirePackage{ifpdf}
```

Now we declare the option “nocounters”, which prevents `dozenal` from redefining all the counters. This prevents errors in document classes which don’t have these counters, such as `minimal`. Defines the command `\nocounters` if and only if the options is named.

```
4 \DeclareOption{nocounters}{%
5 \def\nocounters{}%
6 }%
7 \newif\iftypeone\typeonefalse
8 \DeclareOption{typeone}{\typeonettrue}
9 \ProcessOptions\relax
```

We then define the font that we’re using for our METAFONT-produced Pitman characters. Incidentally, we also define the command `\doz`, though I can’t foresee any decent use for it except in packages and preambles; it is then used to define `\x` and `\e`, which provide the ten and eleven symbols for all the counter redefinitions. This includes definitions for both T1 and OT1 encodings, so it will work with either.

```
10 \DeclareFontFamily{OT1}{dozch}{}
11 \DeclareFontShape{OT1}{dozch}{m}{n}{<-6> dozchars6
12 <7> dozchars7 <8> dozchars8 <9> dozchars9 <10-11>
13 dozchars10 <12-16> dozchars12 <17-> dozchars17 }{}
14 \DeclareFontShape{OT1}{dozch}{b}{n}{<-> dozchb10 }{}
15 \DeclareFontShape{OT1}{dozch}{bx}{n}{<-6> dozchbx6
16 <7> dozchbx7 <8> dozchbx8 <9> dozchbx9 <10-11>
17 dozchbx10 <12-> dozchbx12 }{}
18 \DeclareFontShape{OT1}{dozch}{m}{sl}{<-8> dozchsl8
19 <9> dozchsl9 <10-11> dozchsl10 <12-> dozchsl12 }{}
20 \DeclareFontShape{OT1}{dozch}{bx}{sl}{<-> dozchbxsl10 }{}
21 \DeclareFontShape{OT1}{dozch}{m}{it}{<-7> dozchit7
22 <8> dozchit8 <9> dozchit9 <10-11> dozchit10
23 <12-> dozchit12 }{}
24 \DeclareFontShape{OT1}{dozch}{bx}{it}{<-> dozchbxi10 }{}
25 \ifpdf
26 \pdfmapfile{=fdz.map}
27 \fi
28 \DeclareFontFamily{T1}{dozch}{}
29 \DeclareFontShape{T1}{dozch}{m}{n}{<-> fdzr8r }{}
```

```

30 \DeclareFontShape{T1}{dozch}{bx}{n}{<-> fdzb8r }{}
31 \DeclareFontShape{T1}{dozch}{m}{sl}{<-> fdzs8r }{}
32 \DeclareFontShape{T1}{dozch}{bx}{sl}{<-> fdzbo8r }{}
33 \DeclareFontShape{T1}{dozch}{m}{it}{<-> fdzso8r }{}
34 \DeclareFontShape{T1}{dozch}{bx}{it}{<-> fdzbi8r }{}
35 %\newcommand\doz[1]{\fontfamily{dozch}\selectfont #1}}
36 \iftypeone%
37 \def\doz#1{\fontfamily{dozch}\fontencoding{T1}\selectfont #1}}%
38 \else%
39 \def\doz#1{\fontfamily{dozch}\fontencoding{OT1}\selectfont #1}}%
40 \fi%
41 \newcommand\x{\TextOrMath{\protect\doz{X}}{X}}%
42 \newcommand\ey{\TextOrMath{\protect\doz{E}}{E}}%
43 \DeclareSymbolFont{dozens}{OT1}{dozch}{m}{n}
44 \DeclareMathSymbol{X}{\mathord}{dozens}{88}
45 \DeclareMathSymbol{E}{\mathord}{dozens}{69}

```

Put in some additional code for the tally marks.

```

46 \newcommand\tally[1]{%
47 \usefont{OT1}{dozch}{m}{n}\selectfont{#1}%
48 }%

```

Then we define our command which will produce the dozenal numbers from decimal sources. This algorithm was taken directly from the publicly available archives of comp.text.tex, where it was posted by the well-known and redoubtable David Kastrup. We also define the `\dozens` command, a simplified `\basexii` (which, in fact, depends utterly upon `\basexii`), just to make it easy for everyone.

```

49 \def\basexii#1#2#3{\ifcase\numexpr(#1)\relax
50 0\or1\or2\or3\or4\or5\or6\or7\or8\or9\or#2\or#3\else
51 \expandafter\basexii\expandafter{\number\numexpr((#1)-6)/12}{#2}{#3}\expandafter\basexii\expand
52 \newcommand\dozens[1]{\basexii{#1}{x}{e}}

```

Now that we can convert numbers *to* dozenal, let's set it up so that we can convert them *from* dozenal. This is pretty ugly stuff here, because it's mostly straight \TeX (and e-TeX) without higher-level conveniences, and because it's an expansion nightmare. If we didn't have to account for `\x` and `\e` because included in such strings, it's fairly easy; but it took a great deal of troubleshooting to make it work with them.

First, we work up some macros to help us count the characters in an argument (the same, more or less, as is used in the `basicarith` package, adapted from those by "Florent" at tex-and-stuff.blogspot.com); then, we convert to decimal with `\basex`.

```

53 \newcount\b@charcount
54 \def\gobblechar{\let\char= }
55 \def\assignthencheck{\afterassignment\checknil\gobblechar}
56 \def\countunlessnil{%
57 \ifx\char\nil \let\next=\relax%
58 \else%
59 \let\next=\auxcountchar%
60 \advance\b@charcount by1%

```

```

61 \fi%
62 \ifx\char\backslashadvance\b@charcount by-1\fi%
63 \next%
64 }%
65 \def\auxcountchar{%
66 \afterassignment\countunlessnil\gobblechar%
67 }%
68 \def\countchar#1{\def\xx{#1}\b@charcount=0\expandafter\auxcountchar\xx\nil}
69 %end Florent code
70 \def\gobble#1{%
71 \newcount\@numdigs%
72 \newcount\@decmult%
73 \newcount\@loopindex\@loopindex=1%
74 \newcount\@decnum\@decnum=0%
75 \newcount\@digit\@digit=1%
76 \def\basex#1{%
77 \countchar{#1}\@numdigs=\b@charcount%
78 \@decmult=1\@loopindex=1%
79 \loop\ifnum\@loopindex<\@numdigs%
80 \multiply\@decmult by12%
81 \advance\@loopindex by1%
82 \repeat%
83 \@loopindex=0%
84 \def\listen{x}\def\iselv{e}\def\isquote{\backslash}%
85 \edef\@doznum{\detokenize{#1}\relax}%
86 \@decnum=0%
87 \loop\ifnum\@loopindex<\@numdigs%
88 \def\@firstchar{\expandafter\@car\@doznum\nil}%
89 \if\@firstchar\backslashchar
90 \edef\@doznum{\@removefirstdig{\@doznum}}%
91 \else
92 \if\@firstchar\listen\def\@firstchar{10}\fi
93 \if\@firstchar\iselv\def\@firstchar{11}\fi
94 \advance\@loopindex by1%
95 \@digit=\@firstchar%
96 \edef\@doznum{\@removefirstdig{\@doznum}}%
97 \multiply\@digit by\@decmult%
98 \advance\@decnum by\@digit%
99 \divide\@decmult by12%
100 \fi
101 \repeat
102 \the\@decnum%
103 }
104 \def\@removefirstdig#1%
105 \expandafter\expandafter\expandafter\gobble\expandafter\string#1%
106 }

```

Now, of course, we simply redefine all the counters. This covers only those counters included in the basic L^AT_EX document classes, however, so if you've written your own, you'll need to redefine them yourself.

This first bit ensures that the counters are redefined even if the command `\mainmatter` is not defined. We have to do this outside of the `\g@addto@macro` below; otherwise, in documents where `\mainmatter` is defined but not used, the counters will not be redefined. This way, they’re redefined in all cases.

This also takes care of ensuring that the counters are only redefined if the “nocounters” options was *not* specified.

```

107 \ifundefined{nocounters}{%
108 \ifundefined{c@page}{}{%
109 \renewcommand\thepage{\basexii\value{page}}{\x}{\e}}
110 \ifundefined{c@footnote}{}{%
111 \renewcommand\thefootnote{%
112 \basexii\value{footnote}}{\x}{\e}}
113 \ifundefined{c@part}{}{%
114 \renewcommand\thepart{%
115 \basexii\value{part}}{\x}{\e}}
116 \ifundefined{c@subparagraph}{}{%
117 \renewcommand\thesubparagraph{%
118 \basexii\value{subparagraph}}{\x}{\e}}
119 \ifundefined{c@paragraph}{}{%
120 \renewcommand\theparagraph{%
121 \basexii\value{paragraph}}{\x}{\e}}
122 \ifundefined{c@equation}{}{%
123 \renewcommand\theequation{%
124 \basexii\value{equation}}{\x}{\e}}
125 \ifundefined{c@figure}{}{%
126 \renewcommand\thefigure{%
127 \basexii\value{figure}}{\x}{\e}}
128 \ifundefined{c@table}{}{%
129 \renewcommand\thetable{%
130 \basexii\value{table}}{\x}{\e}}
131 \ifundefined{c@table}{}{%
132 \renewcommand\thempfootnote{%
133 \basexii\value{mpfootnote}}{\x}{\e}}
134 \ifundefined{c@enumi}{}{%
135 \renewcommand\theenumi{%
136 \basexii\value{enumi}}{\x}{\e}}
137 \ifundefined{c@enumii}{}{%
138 \renewcommand\theenumii{%
139 \basexii\value{enumii}}{\x}{\e}}
140 \ifundefined{c@enumiii}{}{%
141 \renewcommand\theenumiii{%
142 \basexii\value{enumiii}}{\x}{\e}}
143 \ifundefined{c@enumiv}{}{%
144 \renewcommand\theenumiv{%
145 \basexii\value{enumiv}}{\x}{\e}}
146 \ifundefined{c@chapter}{%
147 \renewcommand\thesection{%
148 \basexii\value{section}}{\x}{\e}}
149 \renewcommand\thesubsection{%

```



```

150 \thesection.\basexii{\value{subsection}}{\x}{\e}}
151 \renewcommand\thesubsubsection{%
152 \thesubsection.\basexii{\value{subsubsection}}{\x}{\e}}
153 }{
154 \renewcommand\thechapter{%
155 \basexii{\value{chapter}}{\x}{\e}}
156 \renewcommand\thesection{%
157 \thechapter.\basexii{\value{section}}{\x}{\e}}
158 \renewcommand\thesubsection{%
159 \thesection.\basexii{\value{subsection}}{\x}{\e}}
160 \renewcommand\thesubsubsection{%
161 \thesubsection.\basexii{\value{subsubsection}}{\x}{\e}}
162 }

```

Finally, if the `\mainmatter` command is used, we need to make sure that it doesn't mess up our numbering scheme.

```

163 \@ifundefined{mainmatter}{-}{%
164 \g@addto@macro\mainmatter{%
165 \@ifundefined{c@page}{-}{%
166 \renewcommand\thepage{\basexii{\value{page}}{\x}{\e}}}
167 \@ifundefined{c@footnote}{-}{%
168 \renewcommand\thefootnote{\basexii{\value{footnote}}{\x}{\e}}}
169 \@ifundefined{c@part}{-}{%
170 \renewcommand\thepart{\basexii{\value{part}}{\x}{\e}}}
171 \@ifundefined{c@subparagraph}{-}{%
172 \renewcommand\thesubparagraph{%
173 \basexii{\value{subparagraph}}{\x}{\e}}}
174 \@ifundefined{c@paragraph}{-}{%
175 \renewcommand\theparagraph{%
176 \basexii{\value{paragraph}}{\x}{\e}}}
177 \@ifundefined{c@equation}{-}{%
178 \renewcommand\theequation{%
179 \basexii{\value{equation}}{\x}{\e}}}
180 \@ifundefined{c@figure}{-}{%
181 \renewcommand\thefigure{%
182 \basexii{\value{figure}}{\x}{\e}}}
183 \@ifundefined{c@table}{-}{%
184 \renewcommand\thetable{%
185 \basexii{\value{table}}{\x}{\e}}}
186 \@ifundefined{c@table}{-}{%
187 \renewcommand\thempfootnote{%
188 \basexii{\value{mpfootnote}}{\x}{\e}}}
189 \@ifundefined{c@enumi}{-}{%
190 \renewcommand\theenumi{%
191 \basexii{\value{enumi}}{\x}{\e}}}
192 \@ifundefined{c@enumii}{-}{%
193 \renewcommand\theenumii{%
194 \basexii{\value{enumii}}{\x}{\e}}}
195 \@ifundefined{c@enumiii}{-}{%
196 \renewcommand\theenumiii{%

```

```

197 \basexii{\value{enumiii}}{\x}{\e}}
198 \@ifundefined{c@enumiv}{\}%
199 \renewcommand\theenumiv{%
200 \basexii{\value{enumiv}}{\x}{\e}}
201 \@ifundefined{c@chapter}{
202 \renewcommand\thesection{%
203 \basexii{\value{section}}{\x}{\e}}
204 \renewcommand\thesubsection{%
205 \thesection.\basexii{\value{subsection}}{\x}{\e}}
206 \renewcommand\thesubsubsection{%
207 \thesubsection.\basexii{\value{subsubsection}}{\x}{\e}}
208 }{
209 \renewcommand\thechapter{%
210 \basexii{\value{chapter}}{\x}{\e}}
211 \renewcommand\thesection{%
212 \thechapter.\basexii{\value{section}}{\x}{\e}}
213 \renewcommand\thesubsection{%
214 \thesection.\basexii{\value{subsection}}{\x}{\e}}
215 \renewcommand\thesubsubsection{%
216 \thesubsection.\basexii{\value{subsubsection}}{\x}{\e}}
217 } % end if it's defined
218 }
219 }
220 }{} % end redefinition of counters block

```

And that's the end. Thanks for reading, folks; please email me with any suggestions or improvements.

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.

B		E		T	
\basex 3	\e 2	\tally 4
\basexii 2			\typeone 3
D		N		X	
\dozens 2	\nocounters 3	\x 2