The nbaseprt package

Harald Harders
harald.harders@gmx.de

Version v0.11, 2004/12/14; printed August 20, 2012

Abstract

This package prints integer numbers in different bases (octal, decimal, hexadecimal, binary) similarly to the `numprint` package. But here, the number of digits within one group depends on the base.

This version of `nbaseprt.sty` is a BETA VERSION. The main command `\nbaseprint` will stay stable but all configuration commands and the output of `\nbaseprint` may change in future. Please give me feedback what can be improved and if the abbreviations for the different number bases are correct.

Contents

Copyright

1 Load the package

To use this package place

```
\usepackage{nbaseprt}
```

in the preamble of your document. The `nbaseprt` package calls the `numprint` package and parses all package options to it. Please read the documentation for the `numprint` package for details. If you want to use both the `numprint` and the `nbaseprt` package either load `numprint` first given all options, e.g.,

```
\usepackage[autolanguage,nosepfour]{numprint}
\usepackage{nbaseprt}
```

or only load the `nbaseprt` package, giving it the options for `numprint`:

```
\usepackage[autolanguage,nosepfour]{nbaseprt}
```

2 Print numbers

`\nbaseprint` Numbers are printed using the `\nbaseprint{`⟨*number*⟩`}` command. Which number base is used is determined by parsing the ⟨*number*⟩.

The type can be given by preceding the number by "`0x`", "`0o`", "`0d`", or "`0b`" (or the uppercase characters) for hexadecimal, octal, decimal, or binary numbers, respectively. For example,

```
$\nbaseprint{0x1A0E3F}$, $\nbaseprint{0o377377}$,
$\nbaseprint{0d192314}$, $\nbaseprint{0b11010110}$
```

Alternatively, hexadecimal and octal numbers can be given by appending "h", "H", "o", or "O":

```
$\nbaseprint{1A0E3Fh}$, $\nbaseprint{377377o}$
```

If neither is given, the number defaults to decimal.

The format of the printed numbers is similar to the possible input formats. By default, the numbers are preceded by "`0x`", "`0o`", "`0d`", or "`0b`", e.g.

0x 1A 0E 3F, 0o 377 377, 0d 192 314, 0b 1101 0110

`\nbaseposttext` You can change this by using `\nbaseposttext`. This leads to

1A 0E 3F h, 377 377 o, 192 314 d, 1101 0110 b

`\nbasepretext` You can switch back to the default behaviour using `\nbasepretext` or by using `\nbaseposttext` inside a group.

If you want to print negative numbers the sign may be written before or after "`0x`", "`0o`", "`0d`", or "`0b`". Some examples:

```
$\nbaseprint{-0x1A0E3F}$, $\nbaseprint{0o-377377}$,
$\nbaseprint{0d+-192314}$, $\nbaseprint{0b\pm 11010110}$
```

which lead to

0x −1A 0E 3F, 0o −377 377, 0d ±192 314, 0b ±1101 0110

In the printout, the sign always is written after the base-specific string. (is this correct?)

3 Customization

3.1 Padding a number on the left side

\nplpadding
\npnolpadding

Sometimes it is desireable to have a number of a fixed length with the missing digits filled with a character (mostly the character "o", so this is the default). This can be achieved calling `\nplpadding[`⟨*character*⟩`]{`⟨*digits*⟩`}` borrowed from the `numprint` package. For example,

```
\nplpadding{6}%
$\nbaseprint{0xA03E}$, $\nbaseprint{0o1234}$
```

leads to "0x 00 A0 3E, 0o 001 234"

`\npnolpadding` switches padding off.

4 International support

`nbaseprt` uses the thousand separator from `numprint`. Since this package uses the German "`\,`" by default `nbaseprt` does this, too. Using the package option `autolanguage` this can be fixed. If you are using this option without the `babel` package the settings are switched to English at `\begin{document}`: separator "`,`". If using `babel` the separator is changed automatically when switching to a supported language.

If you do not want to use the `autolanguage` option you may use the `numprint` command `\npthousandsep` command to change the separator.

5 Print aligned numbers in tabulars

Sorry, not programmed, yet.

A Lists of options and commands

This section contains lists of all package options resp. available commands. Items that belong together and may be exclusive are printed in groups together.

A.1 Package options

`nbaseprt` supports all options of the `numprint` package. In this list, only the ones that are new or have a different meaning are listed.

The default values are marked by *.

`np` — Define the shortcuts `\np` for `\numprint` and `\nbp` for `\nbaseprint`.

A.2 Commands

Commands that begin with `\np` are borrowed from `numprint`. Here, the new commands and `numprint` commands that have a special meaning for `nbaseprt` are listed here.

`\npaddplus`	Add a plus to a number without a sign.
`\npnoaddplus`	Don't do that.
`\nbp`	Shortcut for `\nbaseprint` (only available with package option `np`).
`\nbaseprint`	Typesets a number (the package's main command).
`\npthousandsep`	Change the separator between the digit groups.
`\nplpadding`	Declare up to how many digits the number will be padded at the lefthand side.
`\npnolpadding`	Switch off padding.
`\nbaserpretext`	Switches on to precede the number by "ox", "oo", or "od".
`\nbaserposttext`	Switches on to append "h", "o", or nothing to the number.

B To do

- Add table support.
- Better customization for the pre and the post text.
- Parse the argument for invalid numbers.
- Proof output format of numbers.

C The implementation

Heading of the package:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{nbaseprt}
  [2004/12/14  v0.11  Print numbers with numerical bases (HH)]
```

Warning, that this is a beta version.

```
\typeout{^^J*****************************************************************}
\PackageWarningNoLine{nbaseprt}{This version of nbaseprt.sty is a BETA
  VERSION.\MessageBreak
  The main command \string\nbaseprint\space will stay stable
  but\MessageBreak
  all configuration commands and the output of\MessageBreak
  \string\nbaseprint\space may change in future.\MessageBreak
  Please give me feedback what can be improved and if\MessageBreak
  the abbreviations for the different number bases are\MessageBreak
  correct}
\typeout{*****************************************************************^^J}
```

Pass all unknown options to `numprint.sty` to avoid conflicts when loading `numprint` seperately.

```
\DeclareOption{np}{%
  \newcommand*\nbp{\nbaseprint}%
  \PassOptionsToPackage{np}{numprint}%
}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{numprint}}
\ProcessOptions\relax
```

Load package `numprint` because `nbaseprt` shares some commands with it.

```
\RequirePackage{numprint}
```

Easier if-then clauses.

```
\RequirePackage{ifthen}
```

Dummy definitions to get an error in case of incompatibility with other packages.

```
\newcommand*\nbprt@base{}
\newcommand*\nbprt@number{}
```

`\nbprt@testbase` Test which numeric base is used in the argument of `\nbaseprt`.

```
\newcommand*\nbprt@testbase{}
\def\nbprt@testbase#1#2#3\@empty{%
```

Test if number begins with "`0x`", "`0X`", or "`"`" which all mean that it is given as hexadecimal number.

```
  \ifthenelse{\equal{#1#2}{0x}\or\equal{#1#2}{0X}\or\equal{#1}{"}}{%
    \gdef\nbprt@base{hex}%
```

Store the rest of the argument as number in `\nbprt@number`.

```
    \ifthenelse{\equal{#1}{"}}{%
      \edef\nbprt@number{#2#3}%
    }{%
      \edef\nbprt@number{#3}%
    }%
  }{%
```

Test if number begins with "`0o`", "`0O`", or "`'`" which all mean that it is given as octal number.

```
    \ifthenelse{\equal{#1#2}{0o}\or\equal{#1#2}{0O}\or\equal{#1}{'}}{%
      \gdef\nbprt@base{oct}%
```

Store the rest of the argument as number in `\nbprt@number`.

```
      \ifthenelse{\equal{#1}{'}}{%
        \edef\nbprt@number{#2#3}%
      }{%
        \edef\nbprt@number{#3}%
      }%
    }{%
```

Test if number begins with "`0d`" or "`0D`" which means that it is given as decimal number.

```
      \ifthenelse{\equal{#1#2}{0d}\or\equal{#1#2}{0D}}{%
        \gdef\nbprt@base{dec}%
```

Store the rest of the argument as number in `\nbprt@number`.

```
        \edef\nbprt@number{#3}%
      }{%
```

Test if number begins with "`0b`" or "`0B`" which means that it is given as decimal number.

```
        \ifthenelse{\equal{#1#2}{0b}\or\equal{#1#2}{0B}}{%
          \gdef\nbprt@base{bin}%
```

Store the rest of the argument as number in `\nbprt@number`.

```
          \edef\nbprt@number{#3}%
        }{%
```

If none of the above is the case the number defaults to decimal.

```
          \def\nbprt@base{dec}%
          \edef\nbprt@number{#1#2#3}%
```

But there are also other possibilities to mark the number as hexadecimal or octal, by appending "h", "H", "o", or "O". These tests are performed by seperate macros.

```
          \nbprt@ishex#1#2#3h\@empty\@empty
          \nbprt@isHex#1#2#3H\@empty\@empty
          \nbprt@isoct#1#2#3o\@empty\@empty
          \nbprt@isOct#1#2#3O\@empty\@empty
        }%
      }%
    }%
  }%
```

Test for a sign before the number.

```
  \expandafter\nbprt@testsign\nbprt@number\@empty\@empty\@empty
```

Reset `\nbprt@string` that holds the number in formatted form.

```
  \def\nbprt@string{}%
```

Reset the counters that help formatting the number.

```
  \@tempcnta=0
  \@tempcntb=0
```

Parse the number, done by `\nbprt@parsenum`.

```
  \expandafter\nbprt@parsenum\nbprt@number\@empty
```

If left padding is switched on, add the leading characters to gain the specified length. See `\nbprt@parsenum` for explanation of the algorithm.

```
  \whiledo{\the\@tempcntb<\nprt@lpaddigits}{%
    \ifnum\@tempcnta=\csname nbprt@digitgroup@\nbprt@base\endcsname\relax
      \edef\nbprt@string{\nprt@separator@before\nbprt@string}%
      \@tempcnta=0
    \fi
    \edef\nbprt@string{\nprt@lpadchar\nbprt@string}%
    \advance\@tempcntb 1
    \advance\@tempcnta 1
  }%
```

Print the text that marks the base of the number before the number itself.

```
  \ifnbprt@pretext
    \csname nbprt@pretext@\nbprt@base\endcsname
    \nbprt@presep
  \fi
```

Print the sign (use routine of `numprint`).

```
  \nprt@printsign{mantissa}\nbprt@sign\@empty
```

Print the modified number with separators.

```
  \nbprt@string
```

Print the text that marks the base of the number after the number itself.

```
  \ifnbprt@pretext
  \else
    \nbprt@postsep
    \csname nbprt@posttext@\nbprt@base\endcsname
  \fi
}
```

\nbprt@testsign

```
\def\nbprt@testsign#1#2#3\@empty{%
% ``#1'', ``#2'', ``#3'':
  \nprt@IfCharInString{#1}{\nprt@signlist}{%
    \edef\nbprt@number{#2#3}%
    \edef\nbprt@sign{#1}%
    \ifx\nbprt@sign\nprt@plus@test
      \def\nprt@tmp{#2}%
      \ifx\nprt@tmp\nprt@minus@test
        \edef\nbprt@sign{+-}%
        \edef\nbprt@number{#3}%
      \fi
    \else
      \ifx\nbprt@sign\nprt@plusminus@test
        \edef\nbprt@sign{+-}%
      \fi
    \fi
  }{%
    \edef\nbprt@number{#1#2#3}%
  }%
}
```

Test if the number is marked as hexadecimal by appending an "h".

```
\def\nbprt@ishex#1h#2\@empty{%
```

If #2 is h, the number has ended with an h because this macro has been called with an appended h in addition to the h that is the last character of the number.

```
  \ifthenelse{\equal{#2}{h}}{%
```

Set the base and redefine the number.

```
    \def\nbprt@base{hex}%
    \edef\nbprt@number{#1}%
  }{}%
}
```

Test if the number is marked as hexadecimal by appending an "H".

```
\def\nbprt@isHex#1H#2\@empty{%
  \ifthenelse{\equal{#2}{H}}{%
    \def\nbprt@base{hex}%
    \edef\nbprt@number{#1}%
  }{}%
}
```

Test if the number is marked as octal by appending an "o" or an "O".

```
\def\nbprt@isoct#1o#2\@empty{%
  \ifthenelse{\equal{#2}{o}}{%
    \def\nbprt@base{oct}%
    \edef\nbprt@number{#1}%
  }{}%
}
\def\nbprt@isOct#1O#2\@empty{%
  \ifthenelse{\equal{#2}{O}}{%
    \def\nbprt@base{oct}%
    \edef\nbprt@number{#1}%
  }{}%
}
```

`\nbprt@parsenum` Parses the given number and generates the formatted string in `\nbprt@string`, working recursively. `#1` is the first character in the left number, `#2` is the rest.

```
\def\nbprt@parsenum#1#2\@empty{%
```

If `#2` is not `\@empty` call `\nbprt@parsenum` recursively to parse the number backwards.

```
  \ifthenelse{\equal{#2}{\@empty}}{}{%
    \expandafter\nbprt@parsenum#2\@empty
  }%
```

Test if `\@tempcnta` has reached the number of digits that are printed as group for the given number base (stored in `\nbprt@digitgroup@`⟨`\nbprt@base`⟩).

```
  \ifnum\@tempcnta=\csname nbprt@digitgroup@\nbprt@base\endcsname\relax
```

Precede the formatted number by the separator `\nprt@separator@before`, taken from `numprint.sty`.

```
    \edef\nbprt@string{\nprt@separator@before\nbprt@string}%
```

Reset the number of handled characters in this group.

```
    \@tempcnta=0
  \fi
```

Precede the formatted number by the current character while forcing uppercase hexadecimal numbers.

```
  \edef\nbprt@string{%
    \uppercase{\ifmmode\mathrm{#1}\else#1\fi}%
    \nbprt@string}%
```

Count this digit for the current group (`\@tempcnta`) and for the total number of digits (`\@tempcntb`).

```
  \advance\@tempcntb 1
  \advance\@tempcnta 1
}
```

`\nbasepretext` Provide a command that switches to marking the numbers before the number itself.

```
\newif\ifnbprt@pretext
\newcommand*\nbasepretext{\nbprt@pretexttrue}
```

`\nbaseposttext` Provide a command that switches to marking the numbers after the number itself.

```
\newcommand*\nbaseposttext{\nbprt@pretextfalse}
```

Provide the commands that print the text before or after the number.

```
\def\nbprt@pretext@hex{0\ifmmode\mathrm{x}\else x\fi}%
\def\nbprt@pretext@oct{0\ifmmode\mathrm{o}\else o\fi}%
\def\nbprt@pretext@dec{0\ifmmode\mathrm{d}\else d\fi}%
\def\nbprt@pretext@bin{0\ifmmode\mathrm{b}\else b\fi}%
\def\nbprt@presep{\,}%
\def\nbprt@posttext@hex{\ifmmode\mathrm{h}\else h\fi}%
\def\nbprt@posttext@oct{\ifmmode\mathrm{o}\else o\fi}%
\def\nbprt@posttext@dec{\ifmmode\mathrm{d}\else d\fi}%
\def\nbprt@posttext@bin{\ifmmode\mathrm{b}\else b\fi}%
\def\nbprt@postsep{\,}%
```

By default, use the marker before the number.

```
\nbasepretext
```

Define how many numbers are grouped together, depending on the number base.

```
\def\nbprt@digitgroup@hex{2}%
\def\nbprt@digitgroup@oct{3}%
\def\nbprt@digitgroup@dec{3}%
\def\nbprt@digitgroup@bin{4}%
```

`\nbaseprint` Define the man command `\nbaseprint` which takes the printed number as mandatory argument.

```
\DeclareRobustCommand*\nbaseprint[1]{%
```

First, expand the number to allow to use macros in the argument.

```
  \edef\nbprt@number{#1}%
```

Test if the number begins with a sign.

```
  \def\nbprt@sign{}%
  \expandafter\nbprt@testsign\nbprt@number\@empty\@empty\@empty
```

Call `\nbprt@testbase` which tests for the number base and prints the number.

```
  \expandafter\nbprt@testbase\nbprt@number\@empty\@empty\@empty
}
```

Change History

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.

O

P

R

T

U

W