Typesetting

CYRILLIC NUMERALS

with ConTEXt MkIV

A Module

The latest Version can be found at `https://bitbucket.org/phg/`.
Mail bugs and fixes or complaints and suggestions to megas.kapaneus@gmail.com.

CONTENT

INTRODUCTION 2

FUNCTIONALITY 4

2.1 Setup 4

2.2 Predefined Commands 8

2.3 Command Derivation 10

USAGE AND PRECAUTIONS 11

3.1 Counters 11

3.2 Font Issues 12

LICENSE 13

REFERENCES 14

ѧ INTRODUCTION

The ConTEXt format comes with a collection of conversion routines for different number systems that are specified in the files `core-con.mkiv` and `core-con.lua`. Cyrillic numerals, however, are not part of this collection. The aim of the module at hand is to provide means of handling Cyrillic numbers and make them seamlessly integrate with the existing interface for number conversion.

If you are familiar with Cyrillic numbers, you might choose to skip the rest of this section and instead continue in medias res with the description on the module's usage in the section on "Functionality".

The Cyrillic numeral system, like the alphabet it is based on, originated from the Greek numerals and thus continues many features of the latter.[1] As with the Roman number system, there are no genuine glyphs reserved for numerals, instead numbers are represented by strings of letters from the ordinary alphabet, organized in a peculiar way; both systems also have the base (10) in common. However, unlike the Roman system Cyrillic numbers are positional, meaning that the numerical value of a digit depends on its location relative to the other digits. The first nine digits, in ascending order, are: "ѧ", "ѩ", "ѫ", "ѧ", "ѥ", "ѕ", "ꙁ", "ꙁ", and "ѳ". As you might have noticed, this series does not correspond to the first nine glyphs of the Cyrillic alphabet (ѧ ѩ ѫ ѧ ѥ ж ѕ ꙁ), but rather to the order of the original Greek letters from which they were derived (α β γ δ ε

n	1	2	3	4	5	6	7	8	9
$n \cdot 10^0$	ѧ	ѩ	ѫ	ѧ	ѥ	ѕ	ꙁ	ꙁ	ѳ
$n \cdot 10^1$	ι	ϰ	ѧ	ϻ	ν	ѯ	ο	π	ϙ
$n \cdot 10^2$	ρ	ҁ	τ	ϒ	ϕ	χ	ѱ	ѡ	ц

Table 1 Number values of the Cyrillic alphabet

[1] Thorough examinations of how Cyrillic numbers were used in praxi are hard to find. The best that the couple of bookshelfs dedicated to grammar in a local department of Slavonic languages has to offer appears to be Žolobov (2006). Another work, Trunte (2005), although it follows a less descriptive but rather instructional approach, deserves mentioning as well.

ϛ ζ η ϑ, with the character ϛ at position № 6 representing “stigma”). The inherited order of numerical values was kept, essentially trading backward compatibility for simplicity. The two other sets of letters that represent multiples of ten and hundred are listed in Table 1. The digits are written in descending order, beginning with the most significant one. The numbers from 11 to 19 follow a different rule for their order mimicks the spoken language, which means that the less significant digit precedes the more significant one (аі, ві, гі, ді, єі, ѕі, зі, иі, ді). There are no glyphs to represent zeros, so they are simply left out. For example, in the Cyrillic system the number 42 is written as мв; the lack of a distinct zero sign causes 402 to have two digits as well, but the character representing the digit 4 gets chosen from the hundreds set: ѵв.

The rules so far don’t allow for numbers above 999. To compensate for the lack of additional letters, greater numbers are represented by the same glyphs (their value being padded by 1000). There are two ways to avoid confusion: Each digit may be prefixed with a special character, the thousands sign ҂. That way, 42 is still written мв, but 42 000 becomes ҂м҂в, and their sum ҂м҂вмв. But if numbers become bigger, the high digits can take alternative ornate forms: ҁ҃ – instead of ҂ҁ – for 100 000, and ҉ for 1 000 000.

In an environment where punctuation was at best minimal and interword spacing a luxury, numbers of this kind tend to be confused with text. Therefore, a Cyrillic number can have additional markers. Dots are used as delimiters before and after a number: ·мв·. Additionally, a number may be indicated by the titlo, which may span its whole length or just parts of it: м҃в.

The Cyrillic number module combines all the above mentioned aspects into one handy command generator, relying on Lua for the conversion routine and METAPOST for the titlo placement. With all options in one place, it is trivial to create and maintain different conversion settings for different purposes.

҂ FUNCTIONALITY

҂.1 SETUP

The module is initialized as any other:

```
\usemodule[cyrillicnumbers]
```

Once the module code is loaded, the setup command provides means to configure all the functionality it offers.

```
\setupcyrnum [.⁝.] [..,.=.,..]
             Optional
1   Identifier

2   command          =  \...#1
    dots             =  no yes
    dotsymbol        =  Text
    penwidth         =  Dimension
    preferhundredk   =  no yes
    titlo            =  mp font
    titlocolor       =  Identifier
    titlolocation    =  middle final first
    titlomode        =  Number
    titlospan        =  Number
```

dots

dotsymbol

Let's walk through the options one by one. As was hinted in the introduction a common praxis is to delimit Cyrillic numbers with dots. Dot placement is enabled or disabled by setting the `dots` key to *yes* or *no* respectively. The `dotsymbol` key allows the user to supply a delimiter of Eir own choice; it defaults to the character "·" (unicode U+00B7). If a font doesn't contain a glyph for this code point or for whatever reason another symbol is required, the solution will look like this: `\setupcyrnum[dots=yes,dotsymbol=\cdot]`. The result of this modification looks as follows: ·ꙋв·. As the dot can be an arbitrary symbol, it could be replaced by the asterisk: *ꙋв*, or – even weirder – the hash character ("Gartenzaun"): #ꙋв#.

In order to notate the sixth decimal digit (for multiples of one hundred million) there are two mutually excluding options: either the hundred thousands sign or the thousands prefix may be employed. Thus, the setup key `preferhundredk` determines which one will be chosen. If set to *yes*, then it's going to be the houndred thousands sign, else the regular thousands sign: ҂ӱ҃ = ҁ + ҁѱ. (As is obvious from this example, the visual quality of the houndred thousands sign, which is a separate glyph, depends on the font used.)

preferhundredk

Not every font contains proper glyphs for the entire Cyrillic unicode range, in fact every dedicated font for one Cyrillic alphabet – Russian, say – might not contain all the characters needed to represent every Cyrillic numeral. This is the result of the historical development the respective scripts went through. This process usually lead to the elimination of several glyphs at different stages of the development. For example the Russian alphabet experienced one significant reduction of letters at the hand of emperor Peter I[2] and another one later in 1917 as a consequence of – not only – the revolution.[3] Thus, chances are that in order to represent Cyrillic numbers, which rely on a superset of the modern Russian alphabet, another font needs to be chosen.[4] The Cyrillic Numbers module provides a hook for this kind of customization: you may define your own font switching macro and assign it to the `command` key of the setup. Suppose you decide to typeset your numbers using the Paratype Serif font.[5]

command

```
%%% 1. Load the module and the font.
\usemodule         [cyrillicnumbers]
\usetypescriptfile [type-paratype.mkiv]

%%% 2. Define the font and a command that switches to it.
\definetypeface [numberfont] [rm]      [serif]
                [paratype]   [default] []
```

[2] For an overview cf. http://www.paratype.ru/e-zine/issue04/peter1/peter1a.htm or just google "гражданский шрифт".

[3] Cf. http://ru.wikipedia.org/wiki/Реформа_русской_орфографии_1918_года.

[4] Font fallbacks are another option if the substitutes match the main font typographically; see Hagen/Hoekwater (2011), pp. 97–99.

[5] A typescript can be found on the ConTEXt mailing list: http://archive.contextgarden.net/message/20110105.204326.d0228ca7.en.html. Keep in mind that ParaType Serif itself is not a particularly suitable number font as it does not contain all required glyphs.

Figure 1 Titlo as a matter of taste.

```
\unexpanded\define[1]\numbercommand{%
  \begingroup
    \language[ru]%
    \setupbodyfont[numberfont]%
    #1%
  \endgroup%
}

%%% 3. Hook the command into our converter.
\setupcyrnum [dots=yes,command=\numbercommand]

%%% 4. Use as needed.
\starttext
Normal text \cyrnum{42} \dots
\stoptext

%%% 5. ????

%%% 6. Profit!!!!!
```

Naturally, the `command` may do anything that fits inside a one-argument macro, from coloring (·мв·) to case manipulation (МВ).

titlo

The key `titlo` controls the placement and, if applicable, placement method of the titlo. The two valid method identifiers are *mp* and *font*, everything else will be interpreted as *no*, e. g. the titlo will be omitted. The latter method, *font*, takes the titlo glyph as supplied by the font file (code point U+483). The main drawback of this solution is that to my knowledge the font titla are designed to span a single char only. As the titlo belongs to the class of combining characters, in the stream of unicode glyphs it will be appended to the character above which it is placed. Thus, multi-digit Cyrillic numbers (i. e. essentially any number with two or more non-zero digits) won't be represented in an optimal way: мв. There is a further option, `titlolocation`, which controls the placement of a font specific titlo. The three possible values specify a position *first*: ѓДсмв, *middle*: ѓДсмв, and *final*: ѓДсмв. (For even character counts, the *middle* argument will pick one place to the right of the exact middle.)

titlolocation

titlospan	result
1	҂Ѵ҂КҀДСМВ
2	҂Ѵ҂КҀДСМВ
3	҂Ѵ҂КҀДСМВ
4	҂Ѵ҂КҀДСМВ
5	҂Ѵ҂КҀДСМВ
6	҂Ѵ҂КҀДСМВ

Figure 2 Different titlo spans.

An alternative to the font titlo is provided by the *mp* variant which uses METAPOST to actually draw a titlo above the string of digits. Not only does this titlo cover the entire numeral, it also comes in a variety of drawing routines. At the moment there are nine more or less different titlos you may choose from as demonstrated in figure 1. These can be enabled on via the `titlomode` key. (Observant users will have recognized mode 8 as the old Rubl' sign: ₽.[6]) The range of digits to be covered by the titlo can be customized by passing the parameter `titlospan` an integer. The default value of 3 results in the titlo spanning at maximum the least significant three digits, because these won't be prefixed by a thousands sign. If the user wants the numeral to be covered as a whole, E can simply pass the value *all*. Beware that the dimensions of the titlo are proportional to the width of the numeral. Therefore, sufficiently wide (in terms of non-zero digits) numbers will cause the titlo to shrink horizontally as seen in figure 2. E. g. for the single digit number ҂Ѧ the titlo even exceeds the character it sits on, while it does not entirely cover the five digits plus two thousand signs of ҂ЍҀДТКѦ. When using the *mp* titlo the color of this element can be chosen separately by passing a valid color identifier to the `titlocolor` key. The following example code demonstrates the colorization and drawing facilities.

```
\usemodule [cyrillicnumbers]
\setupbodyfont [computer-modern-unicode]
\setupcyrnum [
  titlo=mp,
  titlocolor=blue,
  titlospan=all,
  titlomode=7,
]

\starttext
\startlines
```

titlomode,titlospan

titlocolor

6 Cf. http://ru.wikipedia.org/wiki/Рубль##.D0.92_.D0.98.D0.BC.D0.BF.D0.B5.D1.80.D0.B0.D1.82.D0.BE.D1.80.D1.81.D0.BA.D0.BE.D0.B9_.D0.A0.D0.BE.D1.81.D1.81.D0.B8.D0.B8.

```
\cyrnum                                    {42}
\cyrnum   [titlocolor=red,titlomode=9] {141213}
\cyrnum [titlocolor=green,titlomode=2] {271828}
\cyrnum  [titlocolor=cyan,titlomode=4] {314159}
\stoplines
\stoptext
```

penwidth

The METAPOST option also comes with a key , which rather obviously determines the width of the pen that is used when drawing a titlo. Finding the optimal width can involve a lot of testing on the user's side; as a rule, the greater the font size, the wider the pen should be. Refer to table 1 for a demonstration of different values for this parameter.

1	0.600pt	·а·	4201	2.600pt	·дса·
2	0.800pt	·в·	4202	2.800pt	·дсв·
3	1.000pt	·г·	4203	3.000pt	·дсг·
4	1.200pt	·д·	4204	3.200pt	·дсд·
5	1.400pt	·е·	4205	3.400pt	·дсе·
6	1.600pt	·ѕ·	4206	3.600pt	·дсѕ·
7	1.800pt	·з·	4207	3.800pt	·дсз·
8	2.000pt	·и·	4208	4.000pt	·дси·
9	2.200pt	·ѳ·	4209	4.200pt	·дсѳ·
10	2.400pt	·і·	4210	4.400pt	·дсі·

Table 1 Comparison of different values for the parameter `penwidth`.

PREDEFINED COMMANDS

Once the module is loaded, the commands `\cyrnum` and `\cyrnumdrawtitlo` will have been predefined.

`\CYRNUM [..,..=1..,..] {..2..}`

Optional

1 *inherits from* `\setupcyrnum`

2 Content

`\cyrnum` is the default Cyrillic number macro which is fully functional, meaning that besides converting a nonnegative integer into a Cyrillic numeral, it takes a key-value set of options as an optional first argument.

```
\usemodule[cyrillicnumbers]
\starttext

\cyrnum{1}

\cyrnum[titlo=mp,titlomode=4]{42}

\cyrnum{15}

\stoptext
```

Any of the abovementioned settings can be specified in the first argument. As customary with ConTEXt macros, these additional settings are local to one instance. Further calls to the macro won't be affected, unless they are explicitly applied via `\setupcyrnum`,

The use of `titlos` is not restricted to indicating numerals. In addition it is often employed as a kind of emphasis in handwritten text where it is not easy to achieve visual distinction by font switching. Also, the titlo serves as a default marker for abbreviations as in **благо́дѣть** → **бл͠г о́дѣть**.

`\cyrnumdrawtitlo {.*.}`

`*` Content

This is where the macro `\cyrnumdrawtitlo` comes into play. For instance, designations of things considered "sacred" are highlighted by default in some texts. Because they appear very frequently, they were shortened as well, like **господь** → **г͠ь**.[7]

```
господь   ->\cyrnumdrawtitlo{гь}
```

7 Examples taken from http://commons.wikimedia.org/wiki/Category:Titlo?uselang=uk.

```
благодьть -> \cyrnumdrawtitlo{блг}одьть
```

COMMAND DERIVATION

There is no need to reconfigure the `\cyrnum` macro whenever you intend to deviate from the presets. Instead, special purpose commands can be defined via `\definecyrnum`.

`\definecyrnum [.`$\overset{1}{\cdot}$`.] [..,.`$\overset{2}{=}$`.,..]`

Optional

1 Identifier

2 *inherits from* `\setupcyrnum`

All the options that can be passed to `\definecyrnum` are also valid for derived macros; they inherit the setups of the macros they are derived from. A full example to play with is given in below listing:

```
\usemodule[cyrillicnumbers]
\setupbodyfont[computer-modern-unicode]

\definecyrnum[mynumone][titlo=no,dots=no]

\definecyrnum[mynumtwo][mynumone]
\setupcyrnum [mynumtwo][titlo=mp,titlomode=2,titlocolor=red]

\definecyrnum[mynumthree][mynumtwo]
\setupcyrnum [mynumthree][titlomode=4,dots=yes]

\starttext

\mynumone{42}
\mynumtwo{42}
\mynumthree{42}

\stoptext \endinput
```

Which results in: мв $\overset{\sim}{\text{мв}}$ $\cdot\overset{\sim}{\text{мв}}\cdot$.

҂ USAGE AND PRECAUTIONS

҂.а COUNTERS

The macros created by `\definecyrnum` are generic conversion commands. As such, they can be hooked into any functionality that outputs integers of some sort: document structure elements, page numbers &c. In order to have ConTEXt recognize your personal Cyrillic number macro as a converter you need the macro `\defineconversion`.

```
\usemodule [cyrillicnumbers]
\setupbodyfont [computer-modern-unicode]

%%% 1. Define a number converter.
\definecyrnum [neatsections] [
  dots=yes,
  titlo=mp,
  titlomode=7,
]

%%% 2. Register the converter.
\defineconversion [my_section_conversion] [\neatsections]

%%% 3. Insert it into a structure set.
\definestructureconversionset [my_section_set]
  [numbers,my_section_conversion] [my_section_conversion]

%%% 4. Use the set in your chapter command.
\definehead [mychapter] [chapter]
\setuphead  [mychapter] [
  sectionconversionset=my_section_set,
  page=no,
]

\starttext

\dorecurse{10}{%
  \startmychapter[title=foo]
    \input knuth
  \stopmychapter
}
```

```
\stoptext \endinput
```

ѧ.6 FONT ISSUES

Although not the entire Cyrillic alphabet is needed to represent the numerals, they involve certain characters which are uncommon in contemporary languages. Therefore they are usually omitted in Cyrillic fonts, which leads to the problem of finding an adequate font. A matching superset of Knuth's typeface is *Computer Modern Unicode* which is part of TEXLive and packaged for many distributions.[8] CMU is SIL-OpenFont licensed; it has been used in some of the above examples.

There are alternatives offering glyph shapes that resemble the hand written script of Old Slavonic codices. One of them is the beautiful *BukyVede* which has been used as the main Cyrillic font throughout the text. It was created by the Codex project of the Bamberg University[9] but unfortunately the licensing terms are imprecise (non-free). Its authors offer another font matching the popular Times typeface under a yet more restrictive license.[10]

8 Home: http://cm-unicode.sourceforge.net/; TEXLive: http://tug.org/svn/texlive/trunk/Master/texmf-dist/fonts/opentype/public/cm-unicode/; Arch: http://aur.archlinux.org/packages.php?ID=44029; Debian: http://packages.debian.org/wheezy/fonts-cmu.

9 Home: http://kodeks.uni-bamberg.de/aksl/Schrift/BukyVede.htm.

10 Home: http://kodeks.uni-bamberg.de/aksl/Schrift/RomanCyrillicStd.htm.

LICENSE

REFERENCES

1 HAGEN, Hans and Taco HOEKWATER.
2011. *Fonts in ConTEXt*. Hasselt

2 TRUNTE, Nikolaos H.
2005. Altkirchenslavisch. In *Словѣньскъи ꙗꙁыкъ. Ein praktisches Lehrbuch des Kirchenslavischen in 30 Lektionen. Zugleich eine Einführung in die slavische Philologie.*

3 ŽOLOBOV, O. F.
2006. Čislitel′nye. In IORDANIDI, S.I. and V.B. KRYS′KO, editors, *Istoričeskaâ grammatika drevnerusskogo âzyka*, pages 58–63.