Sorin Paliga
University of Bucharest

Working with Old Church Slavonic texts: the simple way from non-standard encoding to unicode encoding

Abstract

In editing Old Church Slavonic (hereafter OCS) texts there are several issues to be solved.

The first refers to the former use of non-standard, non-unicode fonts, which consisted of replacing the Latin characters by the specific OCS characters. This means such a text cannot be displayed if that specific font, often of bad quality, is not installed. The solution seems simple enough: a script, which behaves like a find-replace sequence. After such a replacement, the old font is replaced by a new, good quality font, e.g. Dilyana or Method Std.

The second issue refers to the keyboard layouts (hereafter keylayouts), as the current keylayouts installed with both Windows and OS X do not allow to type all the specific OCS chars. The solution is a dedicated keylayout, for both Cyrillic and Glagolitic, for OS X and Windows.

Using a find-replace sequence also allows to automatically convert Cyrillic to Glagolitic, and vice-versa.

The presentation aims at clarifying some practical aspects, and to show how the author has solved such issues.

Preliminary thoughts

The author of this paper is no expert in Old Church Slavonic (hereafter OCS) texts. Nevertheless, over years, he has encountered some issues — some of them known to all those implied in the laborious activity of editing OCS — which will be briefly analysed here. According to the conventions gradually adopted by various scholars in the modern age, Glagolitic texts were transcribed to Cyrillic, and that was the norm with practically all the Glagolitic documents published in the Modern times.

In the late 1980's and beginning of the 1990's, the computer began to radically change our life, and OCS texts could not avoid this situation, as the texts had to be prepared for print in a computer, and then sent to the printing house as 'camera-ready copies'. In the next phase, these texts may not be printed at all, but uploaded and downloaded as electronic texts, usually in PDF format or being read in a web browser outright.

Two problems gradually occurred: 1. the fonts, which had to include the specific OCS (initially Cyrillic only, Glagolitic characters, hereafter chars, were added later); 2. the keyboard layouts (hereafter keylayouts) had to allow an easy, rational access to the letters and symbols used in the OCS texts. In the first phase of problem #1, the solution was improvised: as the fonts could not include more than 256 chars, including some system commands, the OCS letters and symbols were included in this range by removing the initial chars. They were, therefore, 'Latin fonts'. A company like Linguist Software had such fonts, not only for OCS, and accompanying keylayouts for writing such texts, including various linguistic transcriptions, e.g.

The Indo-European reconstructed root for the numeral '100' was *k̂ṃt-óm* and *k̂ṃt-ā*.

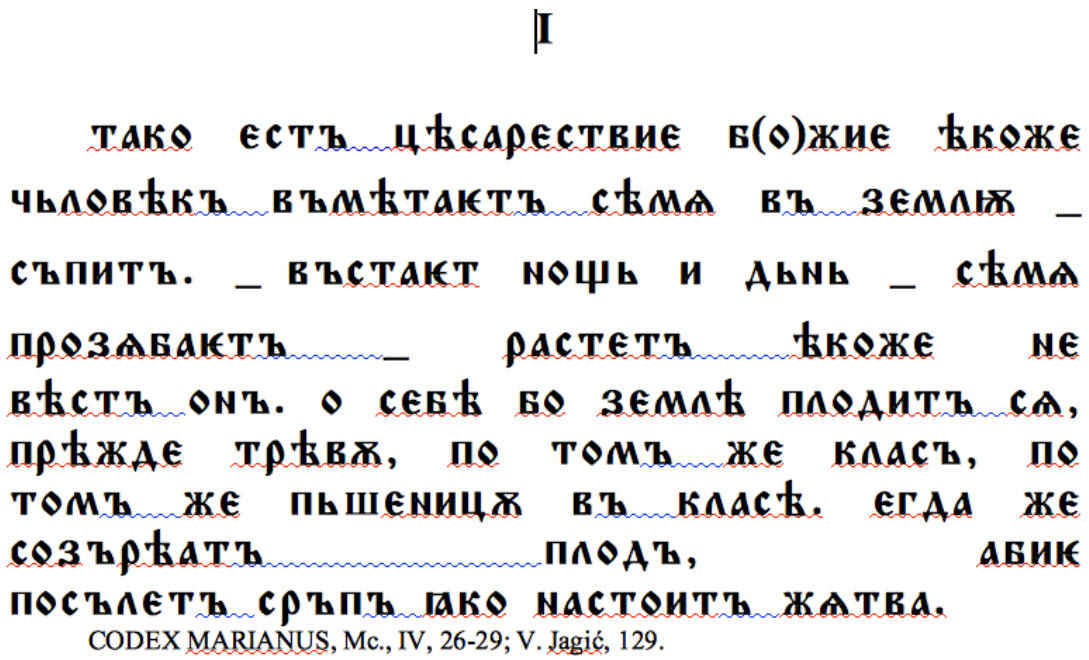
The interim solution in the 1990's was therefore to generate fonts with OCS chars by using the existing encoding, and by replacing the existing Latin chars and symbols by OCS chars. The keyboard layouts were either the existing ones, e.g. for Russian or Bulgarian, with some chars available on other keys. More experienced users could build their keylayouts or some companies, like Linguist Software, were selling their fonts together with the keylayouts.

In the late 1990's and after 2000, a new phase began: the gradual implementation of Unicode conventions and blocks, which led to gradually bringing more coherence in the use of Unicode blocks also by enlarging the limit of fonts to more than 256 chars. As the label *unicode* shows, it means that every char in (virtually) any language must have a unique code (encoding). This led to also including Cyrillic, initially modern Cyrillic (including Cyrillic used for noting non-Slavic languages spoken in the former Soviet Union), then OCS. This implied the additions of some chars, not used in modern Cyrillic and Glagolitic. There still are various symbols used in the OCS texts, only some of them included in the unicode blocks, others to be used in the so-called 'private use area' (hereafter PUA).

The situation may be labelled as ‘coherent and sufficient’ for editing most OCS texts, at least in a simplified form, i.e. by removing occasional symbols used in the these texts. This has been the norm in modern times anyway. Of course, PUA may include various symbols, which may be accessed by key combinations or by inserting them via a utility, available in both macOS and Windows. As the present author is familiar with macOS mainly, the references are to this operating system (hereafter OS, sometimes also labelled ‘platform’), but the situations described are similar or identical on any platform.

Case study #1

One problem refers now to the use of old fonts (the ones current in the 1990’s, but still used), which do not have a rigorous unicode encoding. If we have found in our archives an older document, the picture of which is this one (fortunately we have also kept a print, right?)



CODEX MARIANUS, Mc., IV, 26-29; V. Jagić, 129.

but, alas, we do not know what font was used to write it, when opening it now, we may get something like this:

tako est[chsarestvie b(o)jie hkoje ç]lovhk[v[mhtaét[shmê v[zemlò _ s[pit[. _ v[staét no=] i d]n] _ shmê prozêbaét[_ rastet[hkoje ne vhst[on[. o sebh bo zemlh plodit[sê, prhjde trhvô, po tom[je klas[, po tom[je p]§enicô v[klash. egda je soz[rhat[plod[, abié pos[let[sr[p[áko nastoit[jêtva.

We may loosely identify an OCS text, but what to do if we do not have the original font any more. If it was Method (most probably, which circulated for free on many CD’s with pirated software ad fonts), then a solution is quite simple or, perhaps better phrased, not very difficult: to convert the initial interim encoding to the current unicode encoding. If this is the

case, by using a script I developed for the application Nisus, then the result is the following, if using font Dilyana (© Ralph Cleminson).

тако естъ цѣсарєствиє б(о)жиє ѣкоже ѹьловѣкъ въмѣтаєтъ сѣма въ зємлѭ съпитъ. въстаєт нощь и дьнь сѣма прозѧбаєтъ растєтъ ѣкоже нє вѣстъ онъ. о сєбѣ бо зємлѣ плодитъ сѧ, прѣжде трѣвѫ, по томъ же класъ, по томъ же пьшєницѫ въ класѣ. єгда же созърѣатъ плодъ, абиє посълєтъ сръпъ ѩко настоитъ жатва.

Or, by using font Bukyvede (© Sebastian Kempgen):

тако естъ цѣсарєствиє б(о)жиє ѣкоже ѹьловѣкъ въмѣтаєтъ сѣма въ зємлѭ съпитъ. въстаєт нощь и дьнь сѣма прозѧбаєтъ растєтъ ѣкоже нє вѣстъ онъ. о сєбѣ бо зємлѣ плодитъ сѧ, прѣжде трѣвѫ, по томъ же класъ, по томъ же пьшєницѫ въ класѣ. єгда же созърѣатъ плодъ, абиє посълєтъ сръпъ ѩко настоитъ жатва.

or Lazov font:

тако єстъ цѣсарєствнє б(о)жнє ѣкоже ѹьловѣкъ въмѣтаєтъ сѣма въ зємлѭ съпнтъ. въстаєт нощь н дьнь сѣма прозѧбаєтъ растєтъ ѣкоже нє вѣстъ онъ. о сєбѣ бо зємлѣ плодитъ сѧ, прѣжде трѣвѫ, по томъ же класъ, по томъ же пьшєнницѫ въ класѣ. єгда же созърѣатъ плодъ, абнє посълєтъ сръпъ ѩко настоитъ жатва.

The script used looks like this (the beginning):

```
Require Application Version "3.1"

# work on a copy of the current document
$doc = Document.active
If ! $doc
     Prompt "No document to transliterate."
     Exit
End
$doc = $doc.copy

# Construct the map from old ASCII letters to new Unicode code
points.
# Each pair should have a line that takes the form:
# $map{'X'} = 'Y'
```

```
# Where X is the old character, and Y is the new. As an
example, we could specify
# that an asterisk (*) be converted to ideological zero (〇)
like so:
#
# We could instead use the Unicode code point value if we
wanted:
# $map{'*'} = 0x3007
#
# All these pairs should be added after the following line:
$map = Hash.new
$map{'ⰰ'} = 'ⱁ'
$map{'ⰱ'} = 'ⱑ'
$map{'ⰲ'} = 'ⱂ'
$map{'ⰳ'} = 'ⱃ'
$map{'ⰴ'} = 'ⱄ'
$map{'ⰵ'} = 'ⱅ'
```

In the text above, the Cyrillic letters on the left are in font Method, while on the right it is font Dilyana. If the old font is not available, then we may get a result like this:

```
$map{'a'} = 'ⱁ'
$map{'A'} = 'ⱑ'
$map{'b'} = 'ⱂ'
$map{'B'} = 'ⱃ'
$map{'v'} = 'ⱄ'
$map{'V'} = 'ⱅ'
```

Converting to Glagolitic

It is a most frequent situation to have an original Glagolitic document converted to Cyrillic, which was the norm in the editions of various OCS texts. It is also the case of *Codex Marianus*. If the document is updated to unicode encoding, it is simple enough to have its Glagolitic original look. The fragment chosen as an example would look like this (font Dilyana). To date, very few fonts include the Glagolitic block in their repertoire, one such

example is, beside Dilyana, Bukyvede (© Sebastian Kempgen) and the Google font Noto Sans Glagolitic.

ⱅⰰⰽⱁ ⰵⱄⱅⱏ ⱌⱑⱄⰰⱃⰵⱄⱅⰲⰻⰵ ⰱⱁⰶⰻⰵ ⱑⰽⱁⰶⰵ ⱍⱏⰾⱁⰲⱑⰽⱏ ⰲⱏⰿⱑⱅⰰⰵⱅⱏ ⱄⱑⰿⱔ ⰲⱏ ⰸⰵⰿⰾⱙ ⱄⱏⱂⰻⱅⱏ. ⰲⱏⱄⱅⰰⰵⱅ ⱀⱁⱋⱏ ⰻ ⰴⱏⱀⱏ ⱄⱑⰿⱔ ⱂⱃⱁⰸⱔⰱⰰⰵⱅⱏ ⱃⰰⱄⱅⰵⱅⱏ ⱑⰽⱁⰶⰵ ⱀⰵ ⰲⱑⱄⱅⱏ ⱁⱀⱏ. ⱁ ⱄⰵⰱⱑ ⰱⱁ ⰸⰵⰿⰾⱑ ⱂⰾⱁⰴⰻⱅⱏ ⱄⱔ, ⱂⱃⱑⰶⰴⰵ ⱅⱃⱑⰲⱘ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⰽⰾⰰⱄⱏ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⱂⱏⱎⰵⱀⰻⱌⱘ ⰲⱏ ⰽⰾⰰⱄⱑ. ⰵⰳⰴⰰ ⰶⰵ ⱄⱁⰸⱏⱃⱑⰰⱅⱏ ⱂⰾⱁⰴⱏ, ⰰⰱⰻⰵ ⱂⱁⱄⱏⰾⰵⱅⱏ ⱄⱃⱏⱂⱏ ⱑⰽⱁ ⱀⰰⱄⱅⱁⰻⱅⱏ ⰶⱔⱅⰲⰰ.

Font Bukyvede

ⱅⰰⰽⱁ ⰵⱄⱅⱏ ⱌⱑⱄⰰⱃⰵⱄⱅⰲⰻⰵ ⰱⱁⰶⰻⰵ ⱑⰽⱁⰶⰵ ⱍⱏⰾⱁⰲⱑⰽⱏ ⰲⱏⰿⱑⱅⰰⰵⱅⱏ ⱄⱑⰿⱔ ⰲⱏ ⰸⰵⰿⰾⱙ ⱄⱏⱂⰻⱅⱏ. ⰲⱏⱄⱅⰰⰵⱅ ⱀⱁⱋⱏ ⰻ ⰴⱏⱀⱏ ⱄⱑⰿⱔ ⱂⱃⱁⰸⱔⰱⰰⰵⱅⱏ ⱃⰰⱄⱅⰵⱅⱏ ⱑⰽⱁⰶⰵ ⱀⰵ ⰲⱑⱄⱅⱏ ⱁⱀⱏ. ⱁ ⱄⰵⰱⱑ ⰱⱁ ⰸⰵⰿⰾⱑ ⱂⰾⱁⰴⰻⱅⱏ ⱄⱔ, ⱂⱃⱑⰶⰴⰵ ⱅⱃⱑⰲⱘ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⰽⰾⰰⱄⱏ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⱂⱏⱎⰵⱀⰻⱌⱘ ⰲⱏ ⰽⰾⰰⱄⱑ. ⰵⰳⰴⰰ ⰶⰵ ⱄⱁⰸⱏⱃⱑⰰⱅⱏ ⱂⰾⱁⰴⱏ, ⰰⰱⰻⰵ ⱂⱁⱄⱏⰾⰵⱅⱏ ⱄⱃⱏⱂⱏ ⱑⰽⱁ ⱀⰰⱄⱅⱁⰻⱅⱏ ⰶⱔⱅⰲⰰ.

Font Noto Sans Glagolitic

ⱅⰰⰽⱁ ⰵⱄⱅⱏ ⱌⱑⱄⰰⱃⰵⱄⱅⰲⰻⰵ ⰱⱁⰶⰻⰵ ⱑⰽⱁⰶⰵ ⱍⱏⰾⱁⰲⱑⰽⱏ ⰲⱏⰿⱑⱅⰰⰵⱅⱏ ⱄⱑⰿⱔ ⰲⱏ ⰸⰵⰿⰾⱙ ⱄⱏⱂⰻⱅⱏ. ⰲⱏⱄⱅⰰⰵⱅ ⱀⱁⱋⱏ ⰻ ⰴⱏⱀⱏ ⱄⱑⰿⱔ ⱂⱃⱁⰸⱔⰱⰰⰵⱅⱏ ⱃⰰⱄⱅⰵⱅⱏ ⱑⰽⱁⰶⰵ ⱀⰵ ⰲⱑⱄⱅⱏ ⱁⱀⱏ. ⱁ ⱄⰵⰱⱑ ⰱⱁ ⰸⰵⰿⰾⱑ ⱂⰾⱁⰴⰻⱅⱏ ⱄⱔ, ⱂⱃⱑⰶⰴⰵ ⱅⱃⱑⰲⱘ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⰽⰾⰰⱄⱏ, ⱂⱁ ⱅⱁⰿⱏ ⰶⰵ ⱂⱏⱎⰵⱀⰻⱌⱘ ⰲⱏ ⰽⰾⰰⱄⱑ. ⰵⰳⰴⰰ ⰶⰵ ⱄⱁⰸⱏⱃⱑⰰⱅⱏ ⱂⰾⱁⰴⱏ, ⰰⰱⰻⰵ ⱂⱁⱄⱏⰾⰵⱅⱏ ⱄⱃⱏⱂⱏ ⱑⰽⱁ ⱀⰰⱄⱅⱁⰻⱅⱏ ⰶⱔⱅⰲⰰ.

Writing (typing text) in Cyrillic and Glagolitic

An issue which seems ignored so far is the answer to the question: how can we write (type) an OCS text, be it written in Cyrillic or Glagolitic? The keylayouts included in the OS's regularly include keylayouts for the modern Slavic and non-Slavic languages using Cyrillic, which can only partially cover the needs for 'archaic Cyrillic' (as OCS texts are sometimes labelled) and never the needs for Glagolitic. Of course, I refer to those keylayouts compatible with unicode encoding, and not to be used with the old generation of adapted fonts (in fact, I do not know whether there is one for Glagolitic too).

A second possibility is to use an OCR application, e.g. Readiris. In my tests, it works OK with the modern Slavic languages based on modern Cyrillic, but not with OCS texts. I could not test the last versions of this application or, possibly, another application built for recognizing OCS texts. It may be done, though.

So said, several years ago I built a keylayout for writing (typing) OCS texts, for both Cyrillic and Glagolitic, to be used in macOS. I started from a Cyrillic QWERTY keylayout (sometimes labelled Russian QWERTY, which roughly corresponds to a Serbian Cyrillic keylayout too), i.e. from an existing keylayout built for those who currently use the Latin alphabet, while switching to such a keylayout means it will preserve the correspondence of chars Latin ~ Cyrillic as much as possible, e.g. a is a, b is б, v is в, and h is ч, which is a visual approximation, not a phonetic one, of course.

You may also note that current fonts, even if including the OCS block, will look like modern Cyrillic. The fragment chosen as an example will in fact look in a current font like this, note that ҇а is substituted with the char in Bradley font:

тако єстъ цѣсарєствиє б(о)жиє ѣкожє чьловѣкъ въмѣтаютъ сѣма въ зємлю҇ съпитъ. въстаєт нощь и дьнь сѣма прозабаютъ растєтъ ѣкожє нє вѣстъ онъ. о сєбѣ бо зємлѣ плодитъ са҇, прѣждє трѣвы҇, по томъ жє класъ, по томъ жє пьшєницы҇ въ класѣ. єгда жє созърѣатъ плодъ, абиє посълєтъ сръпъ ҇ако настоитъ жа҇тва.

By using a font like Bradley we may have a better result:

тако єстъ цѣсарєствиє б(о)жиє ѣкоже чьловѣкъ въмѣтаютъ сѣма въ зємлю҇ съпитъ. въстаєт нощь и дьнь сѣма прозабаютъ растєтъ ѣкоже нє вѣстъ онъ. о сєбѣ бо зємлѣ плодитъ са҇, прѣжде трѣвы҇, по томъ жє класъ, по томъ жє пьшєницы҇ въ класѣ. єгда жє созърѣатъ плодъ, абиє посълєтъ сръпъ ҇ако настоитъ жа҇тва.

Such a keylayout uses as many as possible specific chars at the so-called 'zero-level', i.e. without pressing any additional key. Of course, this cannot cover all the char inventory

required, so dead keys are necessary or, at least, the use of the option/alt key.

It may be argued that we do not need, in fact, keylayouts for OCS texts, as we do not generally write (type) OCS documents, we simply prepare them for publication. This is true and, ideally, the outright solution would be OCR and direct conversion of an OCS text directly into an open, editable document. To date, I have not identified such a simple and outfight solution, but it is perfectly possible. Other colleagues may have their positive experience with this. Even so, preparing such texts for publication implies a certain adaptation and simplification of the original, by removing irrelevant details, symbols etc. which of course belong to the flavor of the original document, but usually removed in modern editing, especially when we deal with manuals of OCS when the author is rather preoccupied with presenting OCS grammar and structure in an easy, simplified and gradual way.

Reverting to the problem of keylayouts allowing to write OCS texts, as an author of these two keylayouts (for OCS Cyrillic and Glagolitic), I think that they cover most needs. They are open to improvements and they may be easily adapted to other mapping of the specific chars in order to fit the current habits of the user, e.g. following the specific mapping of keylayouts in Russia or Bulgaria, for example. In macOS, this may be quite easily done with the application UKELELE (free download). OCS texts do have numerous non-standardized symbols and chars. These may be included in the PUA area, and they may be included in a given text by enhancing the existing keylayouts in UKELELE or, alternatively, by picking them with the mouse from a list of chars.

Many years ago, Gé van Gasteren promptly created a Windows verion of OCS Cyrillic. Reportedly he would have been open to creating a Windows version for Glagolitic too, but he had to feedback to the already created one, therefore he assumed, with much justification, that nobody needs it.

A general view

After including OCS blocks in the Unicode encoding, which meant, in several steps, adding chars to the existing ones, most problems have been solved, i.e.:

– Several fonts, notably Dilyana, Bukyvede and Method Std. may be used for editing OCS texts in a modern, easy-to-read form. Glagolitic chars are included in the repertoire of Dilyana, Bukyvede and Google Noto Glagolitic Font, perhaps in other fonts too. This proves sufficient for current use, as both fonts allow easy-to-read Glagolitic fonts, allowing thus editing manuals of OCS, including by using Glagolitic script.

– If OCR cannot be used for old documents, then a solution — at least for the purpose of compiling dictionaries and manuals of OCS – then several keylayouts are available. One set is mine, for both Cyrillic and Glagolitic to run on macOS. Gé van Gesteren has created a Windows version for Cyrillic only. They are available on my webpages at the University of Bucharest under *Software Resources*, including van Gesteren's version for Windows. Other alternatives are on the Kodeks website (Sebastian Kempgen).

– Newer fonts and/or newer generations of fonts may include certain specific symbols in the PUA area. For these, the existing keylayouts may be enhanced in order to easily use these chars too. In macOS, this may be easily done with UKELELE. In Windows, Microsoft's MSKLC may probably work as well (not tested personally, I have not used

Windows for years).

Addendum

The following screenshots show most of the available chars for both Cyrillic and Glagolitic in my keyboard layouts discussed above. There are some other chars, like symbols are diacritical marks, which are also available at the Option/Alt level.