Evaluating Fonts for use in Multi-Lingual Documents

Frank Oberle: November 2016

This is primarily of interest to those who have experienced unwanted and occasionally bizarre font substitutions when creating multi-lingual documents. Such errant substitutions often occur when using more than one Script – a second alphabet if you will [1] – within a single document. Word processors such as LibreOffice Writer have a CTL (Complex Text Layout) feature that permits a user to explicitly define a separate font for a "second" Language that uses a "complex" Script but, even in such cases, it is not uncommon for that font to be replaced either. [2] A common cause for such substitutions is the use of one or more incorrectly formatted font files.

A little history is helpful in understanding why this may occur: in the not-too-distant past, we were restricted to a limited subset of two Scripts at a time (for example: mixing Latin's a, b, c and Thai's ก ข ฃ). Support for more than two such sets simultaneously, however, was rarely possible. Support for the use of multiple Languages – sorting, hyphenation, spell-checking, grammar-checking, and so forth – was similarly limited.

The adoption of Unicode (specifically through its UTF-8 format) now permits computer operating systems to freely intermingle well over 100,000 characters and symbols, and recent advances in font technology, generally identified by some form of the term "Open Type" means that – in theory at least – additional specialized software [3] is no longer necessary to properly place the variety of accents, tone marks and vowels used by many Languages, nor to correctly combine and reorder neighboring characters as many Scripts require.

In order to benefit from these advances, however, one critical requirement is the availability of up-to-date and properly formatted fonts that *not only contain the characters needed, but that also include any instructions required for the glyph manipulations referred to in the previous paragraph* – in other words, "Open-Type-capable" fonts.

There are web postings suggesting that all we need to do is locate and use fonts with an .otf extension, but this is at best a very misleading suggestion.[4] Here's the truth – in the form of a few logical propositions:

1. Fonts with full Open Type capabilities can have either .ttf or .otf extensions. Really.
   The primary difference between the two extensions, by the way, is that .ttf fonts use Quadratic Bézier splines, while .otf fonts use Cubic Bézier splines (as older PostScript Type 1 fonts did).[5]
2. BUT: Not all fonts with a .ttf extension have Open Type capabilities, particularly older ones!
3. Fonts with a .ttf extension that report no Open Type capabilities may be outdated and no longer useful!

If, for example, we need to combine Greek, Thai, and Hindi in a single document, we would ideally need – aside from normal stylistic considerations of course – to be able to ***easily*** find a font or font family that: a) contains an aesthetically compatible collection of all the characters needed and: b) correctly ***reports*** which Languages and Scripts it supports, i.e. a font with Open Type capabilities. In practice, accomplishing this can be quite tedious!

So why the unwanted substitutions? Extremely few fonts contain all of the glyphs defined in the Unicode standard (after all, Coptic and Cuneiform symbols – to give just two examples – are not commonly required). Because of this, operating systems and individual applications often use libraries that will find any missing glyph in another font and transparently replace it. If multiple potential replacements are located, the most suitable substitution is chosen by matching a variety of characteristics such as style, weight, etc. But: (there's always a "but," isn't there?)

1 There isn't necessarily a one-to-one correspondence between Script and Language. A Script is a collection of glyphs representing characters and symbols and may often often be used by more than one Language; a particular Language may use all or part of a particular Script. Most western Languages use the Latin Script - each with a slightly different collection of characters. Some Languages use different Scripts in different contexts (e.g. Kazakh can be written in Cyrillic Script or in (a form of) Arabic Script; Serbian uses either Latin or Cyrillic Scripts.)

2 Since there is no indication that it has done so, here are two methods to determine if Writer (for example) has actually replaced a specified font: 1) choose "Save as" an .fodt file and then open that in a text editor; you can see what font is actually used by examining the styles. 2) choose "Export as" a .pdf file; many pdf readers will have a menu option to list the fonts contained in the document.

3 … and this includes the patronizingly named "CTL" support in some word processors.

4 Actually, it's just wrong. See http://superuser.com/questions/96390/difference-between-otf-open-type-or-ttf-true-type-font-formats for a description that, for me at least, has proven to match all my observations.

5 Some sources suggest that if a font lacks a digital signature – part of the OTF specification – it cannot legitimately use an .otf suffix

Even the best of these utilities are subject to the old data processing maxim “Garbage in = Garbage out” – the infamous “GIGO” syndrome. More fonts than you might expect don’t report their capabilities completely or correctly; older fonts in particular had no need to do so. If your chosen font fails to report its capabilities correctly (or at all), it may fall victim to these no-longer-mysterious replacements!

As for the “ideal need” postulated above, many applications exist – from simple utilities to full-blown font editors – that look within single fonts. But very few will look through several at once[6]; none have an option to restrict their output to just what support is reported for specific Script(s).

Hence, the development of the primitive (but hopefully useful) shell script included at the end of this paper. Rather than relying on what the fonts report to various utilities, it traverses through all of the fonts to locate those containing a representative set of characters from the Scripts of interest, and only then uses standard utilities to determine what those fonts report. The resulting output is then used to further evaluate any potentially suitable fonts, as well to identify fonts that might just need to be replaced, repaired, or discarded.

The `FindFont` script is run from the command line. It is *not* “comprehensive”, but there are enough Language and Script examples in its main *case* section that it shouldn’t be very difficult to add the aforementioned Coptic and Cuneiform definitions should such a need arise. In order to determine which installed fonts have full support for Greek, Thai, and Hindi (our earlier example), the command “`FindFont greek Thai hindi`” will provide a list, and include the level of support reported by each font. “`FindFont farsi Laotian`” is another example.

The script is heavily commented, so a little reading will suffice to clarify what it’s doing and what options are easily changed (e.g. where the script looks for the fonts). The only potentially confusing part is how the `CharMap` variable – a regular expression – for each of the Languages/Scripts is constructed. Regular expressions are certainly well documented (although not always consistently implemented); it is the layout of the targets to be matched that may not be intuitive, so it may help to describe what `$CharMap` is intended to match in a little more detail.

Each modern font should contain a bitmap – a set of single bits representing each of Unicode’s variety of glyphs; a “1” bit indicates the glyph is contained in the font while a “0” means that it isn’t. Simple enough. Surprisingly (to me anyway), a number of fonts on my system that actually contain glyphs/characters were not reported.

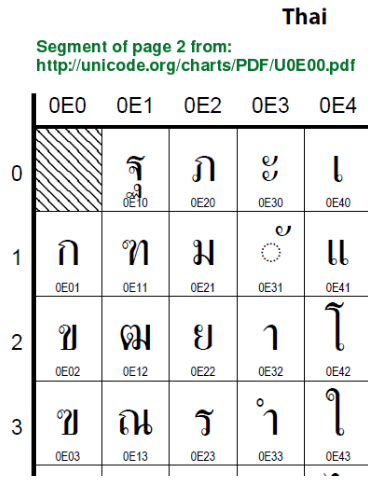
The fc-query utility, used by FindFont to extract the bitmaps, returns one or more lines, each consisting of an offset value followed by eight (8) thirty-two bit words arranged in four bytes each. Here is an example of one such line:

```
    000e: fffffffe 87ffffff 0fffffff 00000000 fef02596 3bffecae 33ff3f5f 00000000
```

This line displays the 256 bits representing the presence or absence of Unicode glyphs/characters in slots `0x0e00` through `0x0eff`. Although these 32-bit words are not numeric values, they are nonetheless laid out as if they were.

As an example of how this affects creation of suitable reguar expressions, assume that we wish to locate a font whose bitmap indicates support for the Thai alphabet. The Unicode standard defines “0E00-0E7F” [7] as the plane for Thai Script.[8] Not all slots in that plane are assigned, however, as can be seen in the partial segment of the chart on the right, which shows that no glyph assigned to `0x0e00`. It is also true that no characters are assigned by the standard to the ranges “`0x0e3b-0x0e3e`” or “`0x0e5c-0x0e7f`” although both are reserved for future use by the Thai Script.

What we want, therefore, is to locate a bit map segment in a row that a) begins with “`000e:`”, and b) contains 1 bits in each of the defined character positions.


Thai

Segment of page 2 from:
http://unicode.org/charts/PDF/U0E00.pdf

	0E0	0E1	0E2	0E3	0E4
0	▨	ฐ 0E10	ภ 0E20	ะ 0E30	เ 0E40
1	ก 0E01	ฑ 0E11	ม 0E21	◌ั 0E31	แ 0E41
2	ข 0E02	ฒ 0E12	ย 0E22	า 0E32	โ 0E42
3	ฃ 0E03	ณ 0E13	ร 0E23	ำ 0E33	ใ 0E43

6 Of those that do, Fontaine is the most comprehensive I’m aware of – its one drawback for many users is that it is only available as source code, and requires compilation: see https://sourceforge.net/projects/fontaine/files/latest/download for more information.

7 See http://unicode.org/charts/, where you can view or download official Unicode charts, look up code points by number, etc. It is important to note that the output of fc-query does not use capital letters for hex characters, so the regular expressions use only small letters.

8 Observant readers will note that this example line also encompasses Laotian, since the Unicode standard places that Script in the “0E80-0EFF” plane. In this example, but Thai and Laotian portion of the font’s bitmaps are contained within in a single line. It isn’t always so easy.

Furthermore, we need to ignore the irrelevant bit values in the row, which could be either 1 or 0. Complicating this somewhat is the fact that we need to effectively translate each nybble (single hex character) into its four component bits. It bears repeating that these are not values: the hex nybble “`8`” is not a value of 8, nor does it represent a “required-ignore-ignore-ignore” (1-0-0-0, since a hex “`8`” is a binary `1000`) sequence but a 0-0-0-1 “ignore-ignore-ignore-required” sequence: the bits in each nybble are read from right to left! An example will illustrate how this works for the output line containing the bitmap for Thai Script given earlier.

Recall that we need to eliminate `0xe00` from consideration, but insure that `0x0e01` through `0x0e03` contain a 1 value. The first nybble must therefore be a “required-required-required-ignore” (1-1-1-0) sequence. Carrying this further, the next twenty-eight bits (seven nybbles), representing `0xe04` through `0xe1f` must all be set to “1” as well. Thus the regular expression that will find a potentially valid matching line in the fc-query output would begin with:

“`000e:[[:space:]]fffffffe`”

But we should continue with the remainder of the desired pattern. According to the Unicode chart, we need “1” bits in positions `0x0e20` through `0x0e3a` and in `0x0e3f`. The twenty-four bits (six nybbles) from `0x0e2a` *down to* `0x0e20` can be represented as `ffffff` but remember: these are the *rightmost* characters of the second word.

The second nybble from the left in the second word, representing just three required glyphs (`0xe3a` *down to* `0xe38` – we don’t care about the undefined position `0x3b` so we can ignore it) means an ignore-required-required-required sequence (0-1-1-1, a hex “`7`”) so the hex nybble “`7`” needs to be added.

To complete the beginning of the second word, we include a required-ignore-ignore-ignore sequence (1-0-0-0, a hex “`8`”) for the `0x0e3f` character. The regular expression now looks like:

“`000e:[[:space:]]fffffffe[[:space:]]87ffffff`”

Finally, to complete our entire regular expression pattern, we note that the final required characters (from `0xe5b` *down to* `0xe40`) occupy exactly 28 bits of the third word’s 32 bits (96-65); these make up, as should be apparent by now, the rightmost seven nybbles of that third word. The regular expression looking for complete coverage of the defined Unicode plane for Thai now looks like:

“`000e:[[:space:]]fffffffe[[:space:]]87ffffff[[:space:]]0fffffff`”

What follows is a more graphic representation of the Thai portion of the Unicode plane:

```
      ┌─32─31─30─29────── the bits within each 4 bit Nybble (hex character) are read from right to left ──────────
      │┌──────────28─27─26─25───── the 4 bit Nybbles within each 32 bit Word are read from right to left ─────
      ││┌──────────────────────24─23─22─21───── but the Words are read from left to right ──────────
      │││┌─────────────────────────────────────20─19─18─17──────────────────────────────
      ││││┌──────────────────────────────────────────────16─15─14─13─────────────────────
      │││││┌───────────────────────────────────────────────────────12─11─10─9────────────
      ││││││┌──────────────────────────────────────────────────────────────8─7─6─5──────
      │││││││┌── in this example, the Nybble ‘e’ of the first 32-bit word represents bits: 1 1 1 0 (hexadecimal ‘e’) ────4─3─2─1──
      ││││││││                                                                      │
      ││││││││ ││││││││ ││││││││ ││││││││ ... ││││││││                               └─ 0x0e03, 0x0e02, 0x0e01 are set but 0x0e00 is not!
000e: fffffffe 87ffffff 0fffffff 00000000 ... 00000000  (0x000e00 is 3584: Bit 1, therefore, represents 0x0e00)
      bits 32-1  bits 64-33  bits 96-65  bits 128-97  . . .  bits 256-224  (each line has 256 bit positions arranged in 8 32-bit Words)
```

A more detailed examination of this line is presented on the following page and includes Laotian Script as well as Thai – which both occupy the `000e:` line) to show how the string

“`000e:[[:space:]01-9a-f]\{37\}fef02596[[:space:]]3bffecae[[:space:]]33ff3f5f`”

was derived as the regular expression to confirm the reporting of the Laotian Script bitmap in a font. A final example, showing an fc-query output line (`0005:`) covering multiple Unicode planes (Cyrillic Supplement U0500-U052F, Armenian (U0530-U058F), and Hebrew (U0590-U05FF); note that Yiddish also uses characters from the Hebrew Script plane, as does Biblical (or chanted) Hebrew, which uses even more glyphs in the plane!

Following the latter example chart is the complete text of the FindFont shell script. Copy it from this pdf into a separate text file, make it executable, and experiment, changing variables as required. Feedback is welcome!

Interpreting a representative line (00e0:…) from the “fc-query [[path/]fontname]” utility

Thai and Lao Scripts as defined in the Unicode Standard (Bit position: +3584 offset)

```
Thai Script         f ── 1 ย  1 ม  1 ภ  1 ฟ  0e1f-0e1c (bits  32- 29)
 Unicode            f ── 1 ผ  1 ป  1 บ  1 น  0e1b-0e18 (bits  28- 25)
  Bit Map           f ── 1 ท  1 ถ  1 น  1 ธ  0e17-0e14 (bits  24- 21)
                    f ── 1 ณ  1 ฒ  1 ฑ  1 ฐ  0e13-0e10 (bits  20- 17)
  Unicode           f ── 1 ฏ  1 ฎ  1 ญ  1 ฌ  0e0f-0e0c (bits  16- 13)
Standard v9         f ── 1 ซ  1 ช  1 ฉ  1 จ  0e0b-0e08 (bits  12-  9)
x0e00-x0e7f         f ── 1 ง  1 ฆ  1 ฅ  1 ค  0e07-0e04 (bits   8-  5)
                    e ── 1 ฃ  1 ข  1 ก  0    0e03-0e00 (bits   4-  1)
Glyphs Used         8 ── 1฿   0    0    0    0e3f-0e3c (bits  64- 61)
 e001-e03a          7 ── 0    1 อฺ  1 อู  1 อุ  0e3b-0e38 (bits  60- 57)
 e03f-e05b          f ── 1 อื  1 อึ  1 อี  1 อิ  0e37-0e34 (bits  56- 53)
                    f ── 1 ำ  1 า  1 อั  1 ะ  0e33-0e30 (bits  52- 49)
                    f ── 1 ฯ  1 ฮ  1 อ  1 ฬ  0e2f-0e2c (bits  48- 45)
                    f ── 1 ห  1 ส  1 ษ  1 ศ  0e2b-0e28 (bits  44- 41)
                    f ── 1 ว  1 ฦ  1 ล  1 ฤ  0e27-0e24 (bits  40- 37)
                    f ── 1 ร  1 ย  1 ม  1 ภ  0e23-0e20 (bits  36- 33)
                    0 ── 0    0    0    0    0e5f-0e5c (bits  96- 93)
                    f ── 1๛   1๚   1 ๙  1 ๘  0e5b-0e58 (bits  92- 89)
                    f ── 1 ๗  1 ๖  1 ๕  1 ๔  0e57-0e54 (bits  88- 85)
                    f ── 1 ๓  1 ๒  1 ๑  1 ๐  0e53-0e50 (bits  84- 81)
                    f ── 1๏   1 อ๎  1 อํ  1 อ์  0e4f-0e4c (bits  80- 77)
                    f ── 1 อ๋  1 อ๊  1 อ้  1 อ่  0e4b-0e48 (bits  76- 73)
                    f ── 1 อ็  1 ๆ  1 ๅ  1 ไ  0e47-0e44 (bits  72- 69)
                    f ── 1 ใ  1 โ  1 แ  1 เ  0e43-0e40 (bits  68- 65)
                      ── 0    0    0    0    0e7f-0e7c (bits 128-125)
                      ── 0    0    0    0    0e7b-0e78 (bits 124-121)
                      ── 0    0    0    0    0e77-0e74 (bits 120-117)
                      ── 0    0    0    0    0e73-0e70 (bits 116-113)
                      ── 0    0    0    0    0e6f-0e6c (bits 112-109)
                      ── 0    0    0    0    0e6b-0e68 (bits 108-105)
                      ── 0    0    0    0    0e67-0e64 (bits 104-101)
                      ── 0    0    0    0    0e63-0e60 (bits 100- 97)

      000e: fffffffe 87ffffff 0fffffff 00000000 00000000 00000000 00000000 00000000
 Hex Range: :00-1f   :20-3f   :40-5f   :60-7f   :80-9f   :a0-bf   :c0-df   :e0-ff
┌Nybble ──           1111111  22222111 33322222 43333333 44444444 55555554 66666555─Tens ─┐
└Number: ─ 87654321 65432109 43210987 21098765 09876543 87654321 65432109 43210987─Units─┘
┌Grep      0        01          12         23         34         45         56         67 7─Used with─┐
└Counter: 123456789012345678901234567890123456789012345678901234567890123456789012─Grep mask─┘
      000e: 00000000 00000000 00000000 00000000 fef02596 3bffecae 33ff3f5f 00000000

0e9f-0e9c (bits 160-157) 1 ຟ  1 ພ  1 ຝ  1 ຜ  ─┘          Lao Script
0e9b-0e98 (bits 156-153) 1 ປ  1 ບ  1 ນ  0    ─e           Unicode
0e97-0e94 (bits 152-149) 1 ທ  1 ຖ  1 ຕ  1 ດ  ─f            Bit Map
0e93-0e90 (bits 148-145) 0    0    0    0    ─0
0e8f-0e8c (bits 144-141) 0    0    1 ຍ  0    ─2            Unicode
0e8b-0e88 (bits 140-137) 0    1 ຊ  0    1 ຈ  ─5          Standard v9
0e87-0e84 (bits 136-133) 1 ງ  0    0    1 ຄ  ─9          x0e80-x0eff
0e83-0e80 (bits 132-129) 0    1 ຂ  1 ກ  0    ─6
0ebf-0ebc (bits 192-189) 0    0    1 ຽ  1 ກ  ─3          Glyphs Used
0ebb-0eb8 (bits 188-185) 1 ກົ  0    1 ກູ  1 ກ  ─b           e081-e082
0eb7-0eb4 (bits 184-181) 1 ກື  1 ກຶ  1 ກີ  1 ກ  ─f             e084
0eb3-0eb0 (bits 180-177) 1 ຳ  1 າ  1 ກັ  1 ະ  ─f           e087-0e88
0eaf-0eac (bits 176-173) 1 ຯ  1 ຮ  1 ອ  0    ─e          e08a    e08d
0eab-0ea8 (bits 172-169) 1 ຫ  1 ສ  0    0    ─c           e094-e097
0ea7-0ea4 (bits 168-165) 1 ວ  0    1 ລ  0    ─a           e099-e09f
0ea3-0ea0 (bits 164-161) 1 ຣ  1 ຢ  1 ມ  0    ─e           e0a1-e0a3
0edf-0edc (bits 224-221) 0    0    1 ໝ  1 ໜ  ─3          e0a5    e0a7
0edb-0ed8 (bits 220-217) 0    0    1 ໙  1 ໘  ─3           e0aa-e0ab
0ed7-0ed4 (bits 216-213) 1 ໗  1 ໖  1 ໕  1 ໔  ─f           e0ad-e0b9
0ed3-0ed0 (bits 212-209) 1 ໓  1 ໒  1 ໑  1 ໐  ─f           e0bb-e0bd
0ecf-0ecc (bits 208-205) 0    0    1 ກໍ  1 ກ  ─3           e0c0-e0c4
0ecb-0ec8 (bits 204-201) 1 ກ໋  1 ກ໊  1 ກ້  1 ກ  ─f              e0c6
0ec7-0ec4 (bits 200-197) 0    1 ໆ  0    1 ໄ  ─5           e0c8-e0cd
0ec3-0ec0 (bits 196-193) 1 ໃ  1 ໂ  1 ແ  1 ເ  ─f           e0d0-e0d9
0eff-0efc (bits 256-253) 0    0    0    0    ──           e0dc-e0dd
0efb-0ef8 (bits 252-249) 0    0    0    0    ──
0ef7-0ef4 (bits 248-245) 0    0    0    0    ──         The Lao “Kip”
0ef3-0ef0 (bits 244-241) 0    0    0    0    ──         (₭) Currency
0eef-0eec (bits 240-237) 0    0    0    0    ──         symbol is at
0eeb-0ee8 (bits 236-233) 0    0    0    0    ──           0x20ad
0ee7-0ee4 (bits 232-229) 0    0    0    0    ──           (8365d)
0ee3-0ee0 (bits 228-225) 0    0    0    0    ──
```

Interpreting a representative line (0005:…) from the “fc-query [[path/]fontname]” utility

Cyrillic (Supp), Armenian, and Hebrew Scripts as defined in the Unicode Standard (Bit position: +1280 offset)

```
                  ──f──(Cyrillic)───────────── 1ԟ  1Ԟ  1ԝ  1Ԝ  051f-051c (bits  32- 29)
Cyrillic          ──f───────────────────────── 1ԛ  1Ԛ  1ԙ  1Ԙ  051b-0518 (bits  28- 25)
Supplement        ──f───────────────────────── 1ԗ  1Ԗ  1ԕ  1Ԕ  0517-0514 (bits  24- 21)
 Unicode          ──f───────────────────────── 1ԓ  1Ԓ  1ԑ  1Ԑ  0513-0510 (bits  20- 17)
  Bit Map         ──f───────────────────────── 1ԏ  1Ԏ  1ԍ  1Ԍ  050f-050c (bits  16- 13)
                  ──f───────────────────────── 1ԋ  1Ԋ  1ԉ  1Ԉ  050b-0508 (bits  12-  9)
  Unicode         ──f───────────────────────── 1ԇ  1Ԇ  1ԅ  1Ԅ  0507-0504 (bits   8-  5)
Standard v9       ──f──(Cyrillic)───────────── 1ԃ  1Ԃ  1ԁ  1Ԁ  0503-0500 (bits   4-  1)
x0500-x052f       ──f─(Armenian)────────────── 1Կ  1Ծ  1Խ  1Լ  053f-053c (bits  64- 61)
                  ──f─(Armenian)────────────── 1Ի  1Ժ  1Թ  1Ը  053b-0538 (bits  60- 57)
___________       ──f─(Armenian)────────────── 1Է  1Զ  1Ե  1Դ  0537-0534 (bits  56- 53)
                  ──e─(Armenian)────────────── 1Գ  1Բ  1Ա  0   0533-0530 (bits  52- 49)
Armenian          ──1─(Cyrillic)────────────── 1ԯ  1Ԯ  1ԭ  1Ԭ  052f-052c (bits  48- 45)
Script            ──1──(Cyrillic)───────────── 1ԫ  1Ԫ  1ԩ  1Ԩ  052b-0528 (bits  44- 41)
 Unicode          ──1──(Cyrillic)───────────── 1ԧ  1Ԧ  1ԥ  1Ԥ  0527-0524 (bits  40- 37)
  Bit Map         ──1──(Cyrillic)───────────── 1ԣ  1Ԣ  1ԡ  1Ԡ  0523-0520 (bits  36- 33)
                  ──f─(Armenian)────────────── 1՟  1՞  1՝  1՜  055f-055c (bits  96- 93)
  Unicode         ──e──(Armenian)───────────── 1՛  1՚  1ՙ  0   055b-0558 (bits  92- 89)
Standard v9       ──7──(etc.)───────────────── 0   1Ֆ  1Օ  1Ք  0557-0554 (bits  88- 85)
x0530-x058f       ──f───────────────────────── 1Փ  1Ւ  1Ց  1Ր  0553-0550 (bits  84- 81)
                  ──f───────────────────────── 1Տ  1Վ  1Ս  1Ռ  054f-054c (bits  80- 77)
Glyphs Used       ──f───────────────────────── 1Ջ  1Պ  1Չ  1Ո  054b-0548 (bits  76- 73)
 0531-0556        ──f───────────────────────── 1Շ  1Ն  1Յ  1Մ  0547-0544 (bits  72- 69)
 0559-055f        ──f───────────────────────── 1Ճ  1Ղ  1Ձ  1Հ  0543-0540 (bits  68- 65)
 0561-0587        ──f───────────────────────── 1տ  1վ  1ս  1ռ  057f-057c (bits 128-125)
 0589-058a        ──f───────────────────────── 1ջ  1պ  1չ  1ո  057b-0578 (bits 124-121)
 058d-058f        ──f───────────────────────── 1շ  1ն  1յ  1մ  0577-0574 (bits 120-117)
                  ──f───────────────────────── 1ճ  1ղ  1ձ  1հ  0573-0570 (bits 116-113)
___________       ──f───────────────────────── 1կ  1ծ  1խ  1լ  056f-056c (bits 112-109)
                  ──f───────────────────────── 1ի  1ժ  1թ  1ը  056b-0568 (bits 108-105)
                  ──f───────────────────────── 1է  1զ  1ե  1դ  0567-0564 (bits 104-101)
                  ──e───────────────────────── 1գ  1բ  1ա  0   0563-0560 (bits 100- 97)

     0005: ffffffff fffe00ff fe7fffff ffffffe 00000000 00000000 00000000 00000000
 Hex Range: :00-1f   :20-3f   :40-5f   :60-7f   :80-9f   :a0-bf   :c0-df   :e0-ff
┌Nybble ──           1111111  22222111 33322222 43333333 44444444 55555554 66666555─Tens ┐
└Number: ─ 87654321 65432109 43210987 21098765 09876543 87654321 65432109 43210987─Units┘
┌Grep      0         01         12         23         34         45         56         67 7─Used with┐
└Counter: 123456789012345678901234567890123456789012345678901234567890123456789012─Grep mask┘
     0005: 00000000 00000000 00000000 00000000 fffe2596 3bffecae ffff001f 001f07ff

  059f-059c (bits 160-157)  1    1    1    1   ─┘                        Armenian
  059b-0598 (bits 156-153)  1    1    1    1   ─f┘                       Script
  0597-0594 (bits 152-149)  1    1֔   1    1   ──f┘                      __Continued_
  0593-0590 (bits 148-145)  1    1֓   1֒   0   ───e┘
  058f-058c (bits 144-141)  0    0    0    0   ────0┘
  058b-0588 (bits 140-137)  0    0    0    0   ─────0┘                   Hebrew
  0587-0584 (bits 136-133)  1և   1ֆ   1օ   1ք  ──────f┘                   Script
  0583-0580 (bits 132-129)  1փ   1ւ   1ց   1ր  ───────f┘                   Unicode
  05bf-05bc (bits 192-189)  0    0    0    0   ──0┘                          Bit Map
  05bb-05b8 (bits 188-185)  0    0    0    0   ───0┘
  05b7-05b4 (bits 184-181)  0    0    0    0   ────0┘                     Unicode
  05b3-05b0 (bits 180-177)  0    0    0    0   ─────0┘                   Standard v9
  05af-05ac (bits 176-173)  1    1    1    1   ──────f┘                  x0590-x05ff
  05ab-05a8 (bits 172-169)  1    1    1    1   ───────f┘
  05a7-05a4 (bits 168-165)  1    1    1    1   ────────f┘                Glyphs Used
  05a3-05a0 (bits 164-161)  1֣   1֢   1֡   1   ─────────f┘               0591-05af
  05df-05dc (bits 224-221)  1ן   1מ   1ם   1ל  ──f┘                      Cantillation
  05db-05d8 (bits 220-217)  1כ   1ך   1י   1ט  ───f┘                       Marks
  05d7-05d4 (bits 216-213)  1ח   1ז   1ו   1ה  ────f┘
  05d3-05d0 (bits 212-209)  1ד   1ג   1ב   1א  ─────f┘                   05d0-05ea
  05cf-05cc (bits 208-205)  0    0    0    0   ──────0┘                  Alphabet
  05cb-05c8 (bits 204-201)  0    0    0    0   ───────0┘
  05c7-05c4 (bits 200-197)  0    0    0    0   ────────0┘                05f0-05f2
  05c3-05c0 (bits 196-193)  0    0    0    0   ─────────0┘               Yiddish
  05ff-05fc (bits 256-253)  0    0    0    0   ───0┘                       Digraphs
  05fb-05f8 (bits 252-249)  0    0    0    0   ────0┘
  05f7-05f4 (bits 248-245)  0    0    0    1״  ─────*┘                   *¹ 0 or 1
  05f3-05f0 (bits 244-241)  1׳   1ײ   1ױ   1װ  ──────*┘                  *² 0,7,8 or f
  05ef-05ec (bits 240-237)  0    0    0    0   ───────0┘
  05eb-05e8 (bits 236-233)  0    1ת   1ש   1ר  ────────7┘                05f3-05f4
  05e7-05e4 (bits 232-229)  1ק   1צ   1ץ   1פ  ─────────f┘               Additional
  05e3-05e0 (bits 228-225)  1ף   1ע   1ס   1נ  ──────────f┘              Punctuation
```

*¹: 7=Yiddish Digraphs only; 8=Add'l punctuation only; 0=Neither; f=Both.

*²: 1=Yiddish Digraphs; 0=No Yiddish Digraphs