Regexp matching with \p{Upper} and \p{Lower} for EUC-JP doesn’t work.

12/22/2015 08:16 AM - matsui (Kimihito Matsui)

Status: Rejected
Priority: Normal
Assignee: matsui (Kimihito Matsui)
Target version: ruby 2.2.2p95 (2015-04-13 revision 50295) [x86_64-darwin14]

Description
U+FF21 (Ａ, FULLWIDTH LATIN CAPITAL LETTER A) and U+00c0 (À, LATIN CAPITAL LETTER A WITH GRAVE) is Uppercase_Letter so it should match and return 0 in following case but this returns 1.

```
ruby -e 'puts "\uFF21A".encode("EUC-JP") =~ Regexp.compile("\\p{Upper}".encode("EUC-JP"))' # => 1
ruby -e 'puts "\u00C0A".encode("EUC-JP") =~ Regexp.compile("\\p{Upper}".encode("EUC-JP"))' # => 1
```

This also happens in lower case matching.

```
ruby -e 'puts "\uFF41a".encode("EUC-JP") =~ Regexp.compile("\\p{Lower}".encode("EUC-JP"))' # => 1
```

In Unicode encoding it works as follows.

```
ruby -e 'puts "\uFF21A" =~ Regexp.compile("\\p{Upper}".encode("EUC-JP"))' # => 0
```

Looks like EUC-JP \p{Upper} and \p{Lower} regex is limited to ASCII characters.

Related Issues:
Related to Ruby master - Feature #13770: Can't create valid Cyrillic-named cl... Closed

History
#1 - 12/22/2015 08:21 AM - matsui (Kimihito Matsui)
- Description updated

#2 - 12/22/2015 08:45 AM - matsui (Kimihito Matsui)
- Description updated

#3 - 06/13/2016 09:37 AM - naruse (Yui NARUSE)
- Status changed from Open to Rejected

Ruby doesn't have case tables for non Unicode encodings.

And EUC-JP is legacy encoding, I don't think such encoding should be extended.

#4 - 06/14/2016 09:32 AM - duerst (Martin Dürst)
Some additional comments following up on the committers' meeting yesterday:

There are many single-byte non-Unicode encodings that have case tables.

Checking the paper versions of the standards in question, À (LATIN CAPITAL LETTER A WITH GRAVE) exists in JIS X 0212-1990 at position (区点) 10-2, and in JIS X 0213-2004 at position 9-23 on the first plane (面). JIS X 0213-2004 is the version I have at hand, but that character didn't change from the -2000 version.

Checking the actual encoding of À in EUC-JP in Ruby shows the following:

```
$ ruby -e 'puts "\u00C0".encode("EUC-JP").b.inspect'
"\x8F\xA4\xA2"
```

This is clearly the JIS X 0212-1990 version, using SS3 (0x8F) to switch to the JIS X 0212 plane at G3. The 1990 version of JIS X 0212 is the first one, so the À character didn't exist in EUC-JP before.
- Related to Feature #13770: Can't create valid Cyrillic-named class/module added