前言 2
第一章 生成字库.....3
第二章 字符点阵预览及编辑.....4
第三章 字库文件.....5
一．文件结构.....5
二．图文详解.....6
三．如何 show 一个字符？ 9
第四章 字符集.....10

前言

一．FontMaker 简介

Fontmaker 是一个可**快速**转换**任意字体，不同点阵，非等宽**的点阵字库工具（由矢量字库转点阵字库）。主要应用在所有需要点阵文字显示的嵌入式系统中。

二．注解

MBCS: Multi Byte Charset，即本地字符集（多字节字符集）。
Unicode: 即统一编码（宽字节字符集）
CJK： China，Japan，Korea，即中（简体，繁体）日韩字符集。
非等宽：每个字符的显示宽度不等。 如： 字符'i'，'M'。
等宽： 每个字符的显示宽度相等。 如： CJK。
DPI: Dots per inch 的缩写，即每英寸的点数。

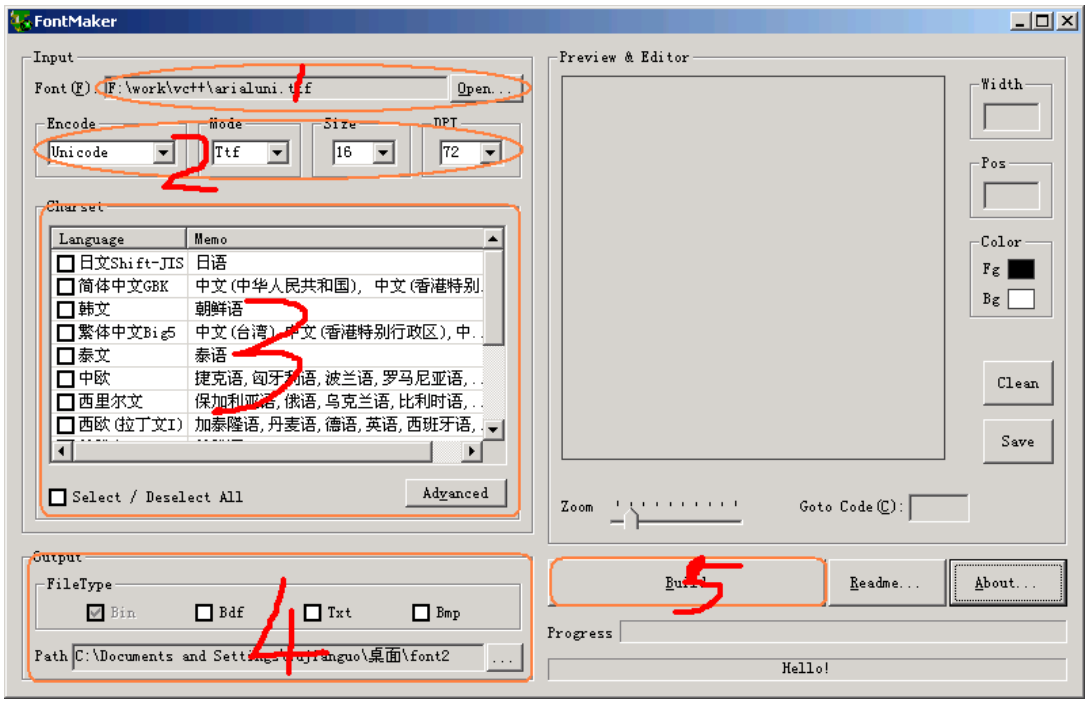
三．信息反馈

为了能够更快更好的完善该工具，以便于大家在使用中达到事半功倍的效果。欢迎您的意见和反馈信息，我们都将慎重考虑、酌情处理，及时给以回复。

联系人： Nancy Office
QQ： 9599598
MSN: wujianguo19@hotmail.com

第一章 生成字库

一． 如何生成字库？



如上图示，分 5 步。

1. **选择矢量字库文件（*.ttf）或 bdf 格式文件（.bdf）.**

   A． 先将 c:\windows\fonts 目录下的字体文件拷贝（ctrl+c）出来,然后本软件即可选择。

   B． 建议去网上找一个 ArialUni.ttf 字库（也可找本人提供，字符非常全面）。

   下载地址： http://ishare.iask.sina.com.cn/f/4942778.html

2. **选择编码格式，转换模式，字体大小，DPI.**

3. **选择需要支持的字符集。**

4. **设置输出文件类型，及输出路径。**

5. **点击 build 按钮，转换生成字库。**

补充说明：

A． 若您选择的是 MBCS 编码格式，则会根据您选择的字符集数产生相应份数的上述文件。

B． 若您选择的是 UNICODE 编码格式，不论您选多少个字符集，都只会输出一份上述文件。

C． 若生成字库失败，则有可能您选择的字体文件(*.TTF)原本就不包含该字符集的字符信息。 比如：宋体中不存在韩文字符，即用宋体生成的字库无法支持韩文显示。

参考办法： 用系统自带的字符映射表进行参照， 其字符映射表中可选择不同的字体、字符集（点击“ 高级查看 ”）。

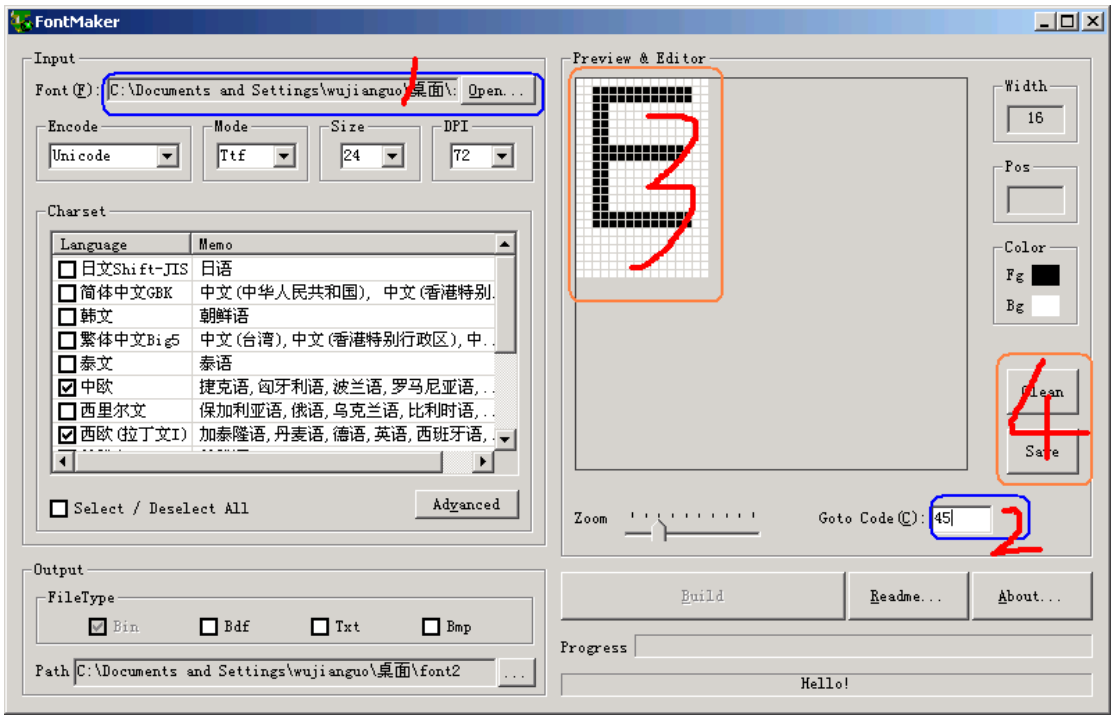
开启字符映射表的方法：

1）以命令方式运行开启，点击**开始**菜单-->选择**运行**-->键入"**charmap**"回车即可。

2）以菜单方式开启，点击**开始**菜单-->**所有程序**-->**附件**-->**系统工具**--->**字符映射表**

第二章 字符点阵预览及编辑

一．如何预览一个字符的点阵信息？



如上图示，分 2 步。

1. **打开一个点阵字库文件（*.bin）**

   将会自动打开并分析出其编码类型，点阵大小，包含的字符集等信息。

2. **根据编码格式（Mbcs, Unicode）输入相应编码（十六进制）。**

   例如：

   a． 字库编码格式为 Mbcs，则输入内码编码。以“ 建 ”字为例，在 2 处输入“ BDA8 ”回车确认即可。

   b． 字库编码格式为 Unicode， 则输入 Unicode 码。 同以“ 建 ”字为例，在 2 处输入“ 5EFA ”回车确认即可。

补充说明： 如某字符不能预览，则表示该字库中不包含此字符。

二．如何简单编辑字符点阵？

1. 将光标移到字符预览区域（3），用**鼠标左键单击描点，右键单击清点。**
2. Clean 与 Save 按钮（4），分别表示**清除**和**更新保存**当前字符的点阵信息。

第三章 字库文件

一．文件结构

字库文件指的是通过 tool 转换出来的 bin 文件(**低字节序**)。通常由以下几部分组成。

1. 文件头

指的是文件的前 16 个字节（BYTE），描述信息如下结构：

```
typedef struct tagFontLibHeader{
    BYTE      magic[4];   //'U'('S', 'M'), 'F', 'L', X---Unicode(Simple or MBCS), X: Version
    DWORD dwFileSize;                    /* File total size */
    BYTE     nSection;    /* total sections */
    BYTE     YSize;       /* height of font    */
    WORD    wCpFlag;     /* codepageflag  每个 bit 位表示一个字符集。即最多可表示 16 个字符集。
*/
    WORD    nTotalChars;    /*  总的有效字符数  */
    char              reserved[2];
} FL_Header, *PFL_Header;
```

2. 段信息

只针对 Unicode 编码有效，占字节数：nSection*sizeof(FL_SECTION_INF)。结构如下：

```
struct tagFlSectionInfo{
        WORD    First;          /* first character                  */
        WORD    Last;           /* last character                   */
        DWORD OffAddr;          /*  指向的是当前 SECTION 包含的  UFL_CHAR_INFO 第一个字符
信息的起始地址  */
    } FL_SECTION_INF, *PFL_SECTION_INF;
```

3． 检索表

只针对 Unicode 字库 和 非等宽的 Mbcs 有效。不包含 Mbcs 的 CJK (简中、繁中、日文、韩文)，因这些都将等宽处理。

```
typedef struct tagUflCharInfo{
    DWORD     OffAddr   : 26;       //  当前字符点阵数据的起始地址
    DWORD     Width     : 6;        //  字符点阵的像素的宽度(  目前最大支持点阵  < 64)
    } UFL_CHAR_INDEX;
```

4．点阵数据

即当前字库中所有包含字符的点阵数据集合。数据存储方式为：横向高到底位存储。如： 10110011 00011010 即为 B3. 1A

补充说明： （Y：包含 / N：不包含）

	文件头	段信息	检索表	点阵数据
Mbcs 等宽 （CJK）	Y	N	N	Y
Mbcs 非等宽 (拉丁文)	Y	N	Y	Y
Unicode	Y	Y	Y	Y

二．图文详解

A. MBCS-等宽 （CJK）

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 46 4C 11 50 FE 03 00 00 10 02 00 F2 1F 00 00 ; MFL.P?.....?..
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 30 00 18 00 0C 00 04 00 00 00 00 00 00 00 ; ..0.............
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

。。。

解析如下：

1）．文件头。

蓝底标记部分，即前 16 Byte，如下图示。

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 46 4C 11 50 FE 03 00 00 10 02 00 F2 1F 00 00 ; MFL.P?.....?..
```

4D 46 4C 11 -- 标识头，判断是否为合法的字库文件。

4D = 'M'， 表示该文件为 MBCS 编码格式的字库文件。

46 = 'F'， 4C = 'L'

11 表示该字库文件版本信息为： Version 1.1

50 FE 03 00 -- 文件总长度 ，即文件大小为：0x3fe50

00 -- 是否包含检索表。 0-标识无检索表。

10 -- 字体高度 （宽高都以像素为单位），即表示 16 点阵。

02 00 -- 选择的字符集标志位。即 0x0002（00000000 00000010），根据字符集序，故得出当前选择为： 简中字符集。

F2 1F -- 有效点阵字符数。 即表示有 0x1ff2 个字符的点阵数据。

00 00 -- 预留字节

2)．点阵数据。

除去文件头 16 Byte 外，其它数据都是纯字符点阵数据。

因为 GB2312（简体中文） 的首个字符为：0xA1A1， 它的点阵数据起始地址为 0x10（除去文件头），数据长度为：((字体高度+7) /8) * 字体高度 = ((16+7)/8)*16=32

故从 0x10 开始连续取 32 字节，即为字符 0xA1A1 的点阵数据。如下图示：

```
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

同理如下即为字符 0xA1A2 的点阵数据。

```
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040h: 00 00 30 00 18 00 0C 00 04 00 00 00 00 00 00 00
```

由于等宽，故所有字符的点阵数据长度都为：((字体高度+7) /8) * 字体高度 = ？

B. MBCS-非等宽 （拉丁文）

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 46 4C 11 70 1A 00 00 01 10 80 00 DA 00 00 00 ; MFL.p.....€.?..
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000070h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000090h: 10 04 00 10 20 04 00 10 30 04 00 18 40 04 00 24 ; .... ...0...@..$
000000a0h: 60 04 00 24 80 04 00 38 A0 04 00 2C C0 04 00 0C ; `..$€..8?.,?..
000000b0h: D0 04 00 14 E0 04 00 14 F0 04 00 18 00 05 00 24 ; ?..?..?.....$
```

…

```
00000410h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000420h: 00 40 40 40 40 40 40 40 40 00 40 40 00 00 00 00 ; .@@@@@@@@.@@....
00000430h: 00 48 48 48 48 00 00 00 00 00 00 00 00 00 00 00 ; .HHHH..........
00000440h: 00 00 09 00 09 00 12 00 FF 80 12 00 34 00 24 00 ; ........  €..4.$.
```

解析如下：

1）. 文件头

蓝底标记部分，即前 16 Byte，如下图示。

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 46 4C 11 70 1A 00 00 01 10 80 00 DA 00 00 00 ; MFL.p.....€.?..
```

4D 46 4C 11 -- 标识头，判断是否为合法的字库文件。

4D = 'M'， 表示该文件为 MBCS 编码格式的字库文件。

46 = 'F'， 4C = 'L'

11 表示该字库文件版本信息为： Version 1.1

70 1A 00 00 -- 文件总长度，即表示文件总长度为： 0x1A70

01 -- 是否包含检索表。 1-标识有检索表

10 -- 字体高度，表示字体大小为 16 点阵。

80 00 -- 选择的字符集标志位。即 0x0080（00000000 10000000），根据字符集序，故得出当前选择为： 西欧字符集。

DA 00 --有效点阵字符数。 即表示有 0xDA 个字符的点阵数据。

00 00 -- 预留字节

2）. 检索表

从 00000010h 开始，每 4 个字节表示一个字符的检索信息， 且从字符 0x0 开始。故空格字符（' '，编码为 0x20）的检索信息（文件头的长度+字符编码*4 = 0x10 + 0x20 *4 = 0x90， 即 000000090h）为：**10 04 00 10**，即得出一个 32 位数为： 0x10000410（十六进制） --- （00010000 00000000 00000100 00010000）.

高 6 位，表示当前字符的宽度。 故得出 000100 -- 4 （字库宽度为 4 ）

低 26 位，表示当期字符的点阵数据的偏移地址。故得出 00 00000000 00000100 00010000 -- 0x410（点阵信息的起始地址为 0x410)

3)．点阵数据

由于空格字符的起始地址为 0x410，且数据长度为：((字体宽度+7) /8) * 字体高度 = ((4+7)/8)*16 = 16.

故取如下 16 字节，即为空格字符的点阵数据。

```
00000410h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

同理如下即为字符'!' 的点阵信息。

```
00000420h: 00 40 40 40 40 40 40 40 00 40 40 00 00 00 00
```

C.Unicode （重点）

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 55 46 4C 11 3C 52 00 00 03 18 A0 00 0F 01 00 00 ; UFL.<R....?....
00000010h: 20 00 D2 06 28 00 00 00 01 0E 5B 0E F4 1A 00 00 ;  .?(.....[.?..
00000020h: 0C 20 22 21 60 1C 00 00 BC 20 00 18 D4 20 00 1C ; . "!`...?..?..
00000030h: EC 20 00 24 1C 21 00 34 4C 21 00 34 7C 21 00 54 ; ?.$.!.4L!.4|!.T
00000040h: C4 21 00 40 F4 21 00 14 0C 22 00 20 24 22 00 20 ; ?.@?...". $".
```

...

```
000000b0h: 44 26 00 40 74 26 00 44 BC 26 00 44 04 27 00 40 ; D&.@t&.D?.D.'.@
000000c0h: 34 27 00 3C 64 27 00 4C AC 27 00 44 F4 27 00 1C ; 4'.<d'.L?.D?..
```

...

```
00002700h: 00 00 00 00 00 00 3F FC 3F FC 30 00 30 00 30 00 ; ....?.???.0.0.
00002710h: 30 00 30 00 3F F8 3F F8 30 00 30 00 30 00 30 00 ; 0.0.???.0.0.0.
00002720h: 30 00 30 00 3F FC 3F FC 00 00 00 00 00 00 00 00 ; 0.0.???.......
00002730h: 00 00 00 00 00 00 3F F8 3F F8 30 00 30 00 30 00 ; ......???.0.0.
00002740h: 30 00 30 00 3F F0 3F F0 30 00 30 00 30 00 30 00 ; 0.0.???.0.0.0.
```

解析如下：

1）．文件头（蓝色框选部分）。 前 16 Byte

55 46 4C 11 -- 标识头，判断是否为合法的字库文件。

55 = 'U'， 表示该文件为 Unicode 编码格式的字库文件。

46 = 'F'， 4C = 'L'

11 表示该字库文件版本信息为： Version 1.1

3C 52 00 00 -- 文件总长度

03 -- 包含几个 Section。03 即表示分为 3 段。

18 -- 字体高度．即表示字体大小为 24（0x18）点阵。

A0 00 -- 选择的字符集标志位。即 0x00A0（00000000 10100000），根据字符集序， 故得出当前选择为： 中欧+西欧字符集。

0F 01 -- 有效点阵字符数。 即表示有 0x10F 个字符的点阵数据。

00 00 -- 预留字节

2）．段信息 （粉红色框选部分）。 nSection * sizeof(FL_SECTION_INF) = 3 * 8 = 24Byte

此文件分 3 段，解析如下：

Section1.

20 00 -- First character，即此段首字符为 0x20

D2 06 -- Last character，即此段尾字符为 0x6D2

28 00 00 00 -- 指向的是首字符（0x20）的字符信息（即检索表）的起始地址，即为 0x28.

Section2.

01 0E -- First character，即此段首字符为 0xE01

5B 0E -- Last character，即此段尾字符为 0xE5B

F4 1A 00 00 -- 指向的是首字符（0xE01）的字符信息（即检索表）的起始地址，即为 0x1AF4.

Section3.

0C 20 -- First character，即此段首字符为 0x200C

22 21 -- Last character，即此段尾字符为 0x2122

60 1C 00 00 -- 指向的是首字符（0x200C）的字符信息（即检索表）的起始地址，即为 0x1C60.

3）. 检索表 （绿色下划线标记部分）

（(Section[0].Last - Section[0].First + 1) * sizeof(UFL_CHAR_INDEX)+ …+ (Section[n-1].Last - Section[n-1].First + 1) * sizeof(UFL_CHAR_INDEX)）

如何快速找到一个字符的字符信息（检索表）？

a. 先比较确定该字符属于哪一段。

b. 精确计算其起始地址。 （起始地址 = 字符的 unicode 码 - Section[x].First ） * sizeof(UFL_CHAR_INDEX) + Section[x].OffAddr）

4). 点阵数据（蓝色下划线标记部分）

例如：字符大写字母‘ E ’(0x45)。

a) 读取字库文件头，获取字库的基本信息。

b) 确定属第几段。 因为 0x45 >= 0x20 && 0x45 <= 0x6D2, 故可确定其在第 **1** 段。

c) 计算点阵信息的起始地址。 起始地址=（0x45-0x20）*4+0x28 = 0xBC 故可得出点阵信息：04 27 00 40, 即为 0x40002704 （01000000 00000000 00100111 00000100）。

高 6 位，表示当前字符的宽度。故得出 010000—16，即当前字符的宽度为 16.

低 26 位，表示当前字符点阵数据的偏移地址。故得出 00 00000000 00100111 00000100 ---0x2704。 即当前字符点阵数据的偏移地址为 0x2704。

d) 蓝色下划线标记部分,即为大写字母‘ E ’的点阵数据。因为它的字体高度为 24， 宽度为 16，故其点阵数据长度为： 24 * （16+8-1）/8 = 48

三．如何 show 一个字符？

1. Mbcs

A. 等宽 （CJK ）

a. 先读出 FL_Header 信息；

b. 计算出 code 在当前字符集中的索引值（index），然后根据这个 sizeof（FL_Header） + index * (FL_Header.Ysize/8*FL_Header.Ysize)找到 code 的点阵数据；

c. 然后根据 FL_Header.Ysize 与点阵数据即可 show 当前字符。

具体见： MBCS 编码中的简中、繁中、日文和韩文索引值计算方法

B． 非等宽 （拉丁文）

a. 先读出 FL_Header 信息；

b. 根据这个 sizeof（FL_Header） + code * 2 找到 code 的 UFL_CHAR_INDEX 信息；

c. 根据 UFL_CHAR_INDEX 的 OffAddr 再找到当前 code 的点阵数据；

d. 最后根据 FL_Header.Ysize、UFL_CHAR_INDEX.Width 以及点阵数据即可 show 出当前字符。

2. Unicode

a. 先读出 FL_Header 信息；

b. 分析当前字符属第几段。比如在第 n 段，就可根据这个 xxx[n].OffAddr+（code - xxx[n].First）* sizeof（UFL_CHAR_INDEX）找到字符检索信息(UFL_CHAR_INDEX)；

c. 根据 UFL_CHAR_INDEX 的 OffAddr 再找到当前字符的点阵数据；

c. 最后根据 FL_Header.Ysize、UFL_CHAR_INDEX.Width 以及点阵数据即可 show 出当前字符。

第四章 字符集

一． **支持所有 windows 字符集**。如下：

CP932， 日文 Shift-JIS， 如：日语

CP936， 简体中文 GBK， 如：中文(中华人民共和国)，中文(香港特别行政区)，中文(新加坡)

CP949， 韩文， 如：朝鲜语

CP950， 繁体中文 Big5， 如；中文(台湾)，中文(澳门特别行政区)

CP874， 泰文， 如：泰语

CP1250， 中欧， 如：捷克语,匈牙利语,波兰语,罗马尼亚语,克罗地亚语,斯洛伐克语,阿尔巴尼亚语,斯洛文尼亚语,塞尔维亚语(拉丁文)

CP1251， 西里尔文， 如：保加利亚语,俄语,乌克兰语,比利时语,马其顿语(FYROM),哈萨克语,吉尔吉斯语,鞑靼语,蒙古语,阿塞拜疆语,乌兹别克语,塞尔维亚语

CP1252， 西欧(拉丁文 I)， 如：加泰隆语,丹麦语,德语,英语,西班牙语,芬兰语,法语,冰岛语,意大利语,荷兰语,挪威语,葡萄牙语,印度尼西亚语,巴士克语,南非语,法罗语,马来语,斯瓦希里语,加里西亚语,瑞典语

CP1253， 希腊文， 如：希腊语

CP1254， 土耳其文， 如：土耳其语,阿塞拜疆语,乌兹别克语

CP1255， 希伯来文， 如：希伯来语

CP1256， 阿拉伯文， 如：乌都语,波斯语,阿拉伯语(伊拉克,埃及,利比亚,阿尔及利亚,摩洛哥,突尼斯,阿曼,也门,叙利亚,约旦,黎巴嫩,科威特,阿联酋,巴林,卡塔尔)

CP1257， 波罗的海文， 如：爱沙尼亚语,拉脱维亚语,立陶宛语，

CP1258， 越南， 如：越南语

二．字符集顺序

0. 日文，
1. 简中，
2. 韩文，
3. 繁中，
4. 泰文，
5. 中欧，
6. 西里尔，
7. 西欧，
8. 希腊，
9. 土耳其文，
10. 希伯来文，
11. 阿拉伯文，
12. 波罗的海文，
13. 越南文。