# 第7章 字体

随μC/GUI 一起提供的普通字体大部分是标准字体。实际上，你或许会发现这些字体对于你的应用已完全足够。对单独的字体，想了解更详细的情况，请参考第 25 章：标准字体，这一章描述了μC/GUI 中所有的字体，及这些字体在屏幕上显示时的所有字符。

μC/GUI 支持 ASCII，ISO 8859-1 及 Unicode。通常，μC/GUI 按 8 位字符进行编译，允许最大为 256 的不同的字符代码，32 之前的编码除外，这部分字符作为控制字符保留。字符是否有效取决所选择的字体（即该字体是否包括有所需的字符）。

# 7.1 有效字体

当前μC/GUI 版本提供 4 种字体：等宽位图字体，比例位图字体，带有 2bpp（bit/pixel）用于建立反混淆信息的比例位图字体，带有 4bpp（位/像素）用于建立反混淆信息的反混淆字体。（更多关于抗锯齿字体的信息请参考第 15 章：“抗锯齿”）

所有的字体会与你的应用相连接，而字体的选择在 GUIConf.h 中定义。我们推荐编译所有的字体并将它们作为一个库模块进行连接，或者将所有的字体工程文件放入一个你能与你的应用相连接的库当中。这种方法可以让你确定那些你在应用中需要的字体被真正连接。字体转换器（在一本独立的手册中描述）用于创建附加的字体。

为了能在你的应用中使用一种字体，你必须要做到下面几点：

- 字体在与μC/GUI 规范相兼容的“C”文件，工程文件或库这三种文件中任一种当中。
- 字体文件与你的应用链接。
- 字体的描述要包含在 GUIConf.h 中（这很必要，这是为了避免由于没有声明的外部常量而产生的编译警告）

## 增加字体

一旦你连接过一个如上面所描述的字体文件，将要连接的字体声明为一个外部常量 GUI_FONT，如下面范例所显示的那样：

## 范例

```
extern const GUI_FONT GUI_FontNew;
int main (void)
{
   GUI_Init () ;
   GUI_Clear();
   GUI_SetFont(&GUI _FontNew);
   GUI_DispString("Hello world\n");
   return 0;
}
```

## 选择字体

μC/GUI 提供不同的字体，总会有其中的一种被选中。可以通过调用函数 GUI_SetFont()

改变所选择的字体，该函数选择字体用于伴随当前任务的文字输出。

如果在你的应用中没有字体被选择，则使用默认字体。该默认值由 GUIConf.h 配置，可以进行修改。你应该确认默认字体是你的应用中真正用到的字体，因为默认字体会与你的应用连接，因此可能会耗尽 ROM 存储空间。

### µC/GUI 的兼容性

老版本的µC/GUI 使用一个不同的字体概念，字体在一个字体表中列出，它们在表格中的位置通过一个整数选择。由于缺乏灵活性，这种概念改变了。新的概念较原来的概念提高了一个等级，字体标识符（例如 F6x8）依然有效。

## 7.2 字体API

下表列出了与字体处理相关的函数，在各自的类型中按字母顺序进行排列。函数的详细描述后面列出。

| 函　数 | 说　明 |
|---|---|
| **字体的选择** | |
| GUI_GetFont() | 返回当前选择字体的指针 |
| GUI_SetFont() | 设置当前字体 |
| **字体相关函数** | |
| GUI_GetCharDistX() | 返回当前字体中指定字符的宽度（X 轴，以像素为单位） |
| GUI_GetFontDistY() | 返回当前字体 Y 轴方向间距 |
| GUI_GetFontInfo() | 返回一个包含字体信息的结构 |
| GUI_GetFontSizeY() | 返回当前字体的高度（Y 轴，以像素为单位） |
| GUI_GetStringDistX() | 返回一个使用当前字体的文本的 X 轴尺寸 |
| GUI_GetYDistOfFont() | 返回一个特殊字体的 Y 轴间距 |
| GUI_GetYSizeOfFont() | 返回一个特殊字体的 Y 轴尺寸 |
| GUI_IsInFont() | 估计一个指定的字符是否在一种特殊字体里面 |

## 7.3 一种字体的选择

### GUI_GetFont()

#### 描述

返回当前选择字体的指针。

**函数原型**

```
const GUI_FONT * GUI_GetFont(void)
```

## GUI_SetFont()

**描述**

设置用于文字输出的字体。

**函数原型**

```
const GUI_FONT * GUI_SetFont(const GUI_FONT * pNewFont) ;
```

| 参数 | 含义 |
|---|---|
| pFont | 所选择及使用字体的指针 |

**返回值**

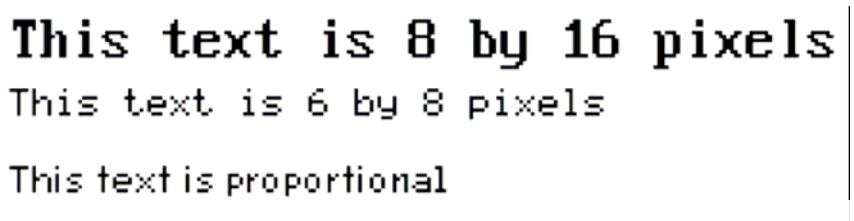
返回先前所选择字体的指针，这样你可以在稍后一点恢复原先使用的字体。

**范例**

用 3 种不同尺寸显示样本文字，然后恢复原先的字体：

```
void DispText(void)
{
   const GUI_FONT GUI_FLASH* OldFont=GUI_SetFont(&GUI_Font8x16);
   GUI_DispStringAt("This text is 8 by 16 pixels", 0, 0);
   GUI_SetFont(&GUI_Font6x8);
   GUI_DispStringAt("This text is 6 by 8 pixels", 0, 20);
   GUI_SetFont(&GUI_Font8);
   GUI_DispStringAt("This text is proportional", 0, 40);
   GUI_SetFont(OldFont);                    // 恢复字体
}
```

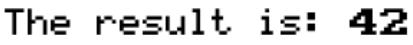
**上面范例程序运行结果的屏幕截图：**




用不同的字体显示文字和数值：

```
GUI_SetFont(&GUI _Font6x8);
GUI_DispString("The result is:");        // 显示文字
GUI_SetFont(&GUI_Font8x8);
GUI_DispDec(42,2);                       // 显示数值
```

**上面范例程序运行结果的屏幕截图：**




## 7.4 字体相关函数

### GUI_GetCharDistX()

**描述**

返回当前选择字体中用于显示一个指定字符的宽度（X 轴，以像素为单位）

**函数原型**

```
int GUI_GetCharDistX(U16 c);
```

| 参数 | 含义 |
|---|---|
| c | 需计算宽度的字符 |

### GUI_GetFontDistY()

**描述**

返回当前选择字体 Y 轴方向间距

**函数原型**

```
int GUI_GetFontDistY(void);
```

**附加信息**

Y 轴方向间距是一个以像素为单位在两个文字相邻的线之间的垂直距离。返回值是当前选择字体入口 Y 轴方向距离数值。该返回值对于比例字体及等宽字体都有效。

## GUI_GetFontInfo()

**描述**

计算指向一个特殊字体的 GUI_FONTINFO 结构的指针。

**函数原型**

```
void GUI_GetFontInfo(const GUI_FONT*pFont, GUI_FONTINFO* pfi);
```

| 参数 | 含义 |
|---|---|
| pFont | 指向字体的指针 |
| pfi | 指向一个 GUI_FONTINFO 结构的指针 |

**附加信息**

GUI_FONTINFO 结构的定义如下：

```
typedef struct
{
  U16 Flags;
}GUI_FONTINFO;
```

变量标志的成员使用以下数值：

```
GUI_FONTINFO_FLAG_PROP
GUI_FONTINFO_FLAG_MONO
GUI_FONTINFO_FLAG_AA
GUI_FONTINFO_FLAG_AA2
GUI_FONTINFO_FLAG_AA4
```

**范例**

获得 GUI_Font6x8 字体的信息。计算后，FontInfo.Flags 包含 GUI_FONTINFO_FLAG_MONO 标志。

```
GUI_FONTINFO FontInfo;
GUI_GetFontInfo(&GUI_Font6x8, &FontInfo);
```

## GUI_GetFontSizeY()

**描述**

返回当前选择字体的高度（Y 轴，以像素为单位）

**函数原型**

```
int GUI_GetFontSizeY(void);
```

**附加信息**

返回值是当前选择字体入口 Y 轴方向大小数值。该值小于或等于通过执行 GUI_GetFontDistY()获得的返回值——Y 轴方向间距。

该返回值对于比例字体及等宽字体都有效。

## GUI_GetStringDistX()

**描述**

返回在当前选择的字体中用于显示一个指定字符串的 X 轴尺寸。

**函数原型**

```
int GUI_GetStringDistX(const char GUI_FAR *s);
```

| 参数 | 含义 |
| --- | --- |
| s | 字符串的指针 |

## GUI_GetYDistOfFont()

### 描述

返回一种特殊字体的 Y 轴方向间距。

### 函数原型

```
int GUI_GetYDistOfFont(const GUI_FONT* pFont);
```

| 参数 | 含义 |
| --- | --- |
| pFont | 字体的指针 |

### 附加信息

参考 GUI_GetFontDistY()。

## GUI_GetYSizeOfFont()

### 描述

返回一种特殊字体的 Y 轴方向尺寸。

### 函数原型

```
int GUI_GetYSizeOfFont(const GUI_FONT* pFont);
```

| 参数 | 含义 |
| --- | --- |
| pFont | 字体的指针 |

### 附加信息

参考 GUI_GetFontSizeY()

## GUI_IsInFont()

### 描述

估计一个指定的字符是否在一种特殊字体里面

**函数原型**

```
char GUI_IsInFont(const GUI_FONT*pFont, U16 c);
```

| 参数 | 含义 |
|---|---|
| pFont | 字体的指针 |
| c | 搜索的字符 |

附加信息

如果指针 pFont 设置为 0，则使用当前选择字体。

**范例**

估计字体 GUI_FontD32 是否包含“X”：

```
if (GUI_IsInFont(&GUI_FontD32, 'X') == 0)
{
   GUI_DispString("GUI_FontD32 does not contains 'X'");
}
```

# 7.5 字符设置

## ASCII

µC/GUI 支持所有的 ASCII 字符。下表是从 32 到 127 共 96 个字符：

| Hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2x | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4x | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5x | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6x | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7x | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |

很不幸，因为 ASCII 立足于美国用于信息互换的标准，它为美国的需要而制定。它不包括用于欧洲语言的任何特殊的字符，诸如 Ä，Ö，Ü，á，à 等等。没有单独的标准适合这些 ASCII 字符的“欧洲扩展”。现在已经有几个不同的标准，其中一个用于互联网并且被大多数 Windows 应用程序所接受的标准是 ISO8859-1，一个 ASCII 字符的扩展集。

## ISO 8859-1 西文、拉丁文字符设置

µC/GUI 支持 ISO 8859-1，字符的定义如下表如示：

| 代码 | 描述 | 字符 |
|---|---|---|
| 160 | non-breaking space | |
| 161 | inverted exclamation | ¡ |
| 162 | cent sign | ¢ |
| 163 | pound sterling | £ |
| 164 | general currency sign | ¤ |
| 165 | yen sign | ¥ |
| 166 | broken vertical bar | ¦ |
| 167 | section sign | § |
| 168 | umlaut (dieresis) | ¨ |
| 169 | copyright | © |
| 170 | feminine ordinal | ª |
| 171 | left angle quote, guillemotleft | « |
| 172 | not sign | ¬ |
| 173 | soft hyphen | |
| 174 | registered trademark | ® |
| 175 | macron accent | ¯ |
| 176 | degree sign | ° |
| 177 | plus or minus | ± |
| 178 | superscript two | ² |
| 179 | superscript three | ³ |
| 180 | acute accent | ´ |
| 181 | micro sign | µ |
| 182 | paragraph sign | ¶ |
| 183 | middle dot | · |
| 184 | cedilla | ¸ |
| 185 | superscript one | ¹ |
| 186 | masculine ordinal | º |
| 187 | right angle quote, guillemot right | » |
| 188 | fraction one-fourth | ¼ |
| 189 | fraction one-half | ½ |
| 190 | fraction three-fourth | ¾ |
| 191 | inverted question mark | ¿ |
| 192 | capital A, grave accent | À |
| 193 | capital A, acute accent | Á |

| | | |
|---|---|---|
| 194 | capital A, circumflex accent | Â |
| 195 | capital A, tilde | Ã |
| 196 | capital A, dieresis or umlaut mark | Ä |
| 197 | capital A, ring | Å |
| 198 | capital A, diphthong (ligature) | Æ |
| 199 | capital C, cedilla | Ç |
| 200 | capital E, grave accent | È |
| 201 | capital E, acute accent | É |
| 202 | capital E, circumflex accent | Ê |
| 203 | capital E, dieresis or umlaut mark | Ë |
| 204 | capital I, grave accent | Ì |
| 205 | capital I, acute accent | Í |
| 206 | capital I, circumflex accent | Î |
| 207 | capital I, dieresis or umlaut mark | Ï |
| 208 | Eth, Icelandic | Ð |
| 209 | N, tilde | Ñ |
| 210 | capital O, grave accent | Ò |
| 211 | capital O, acute accent | Ó |
| 212 | capital O, circumflex accent | Ô |
| 213 | capital O, tilde | Õ |
| 214 | capital O, dieresis or umlaut mark | Ö |
| 215 | multiply sign | × |
| 216 | capital O, slash | Ø |
| 217 | capital U, grave accent | Ù |
| 218 | capital U, acute accent | Ú |
| 219 | capital U, circumflex accent | Û |
| 220 | capital U, dieresis or umlaut mark | Ü |
| 221 | capital Y, acute accent | Ý |
| 222 | THORN, Icelandic | Þ |
| 223 | sharp s, German (s-z ligature) | ß |
| 224 | small a, grave accent | à |
| 225 | small a, acute accent | á |
| 226 | small a, circumflex accent | â |
| 227 | small a, tilde | ã |
| 228 | small a, dieresis or umlaut mark | ä |
| 229 | small a, ring | å |
| 230 | small ae diphthong (ligature) | æ |
| 231 | cedilla | ç |
| 232 | small e, grave accent | è |

| | | |
|---|---|---|
| 233 | small e, acute accent | é |
| 234 | small e, circumflex accent | ê |
| 235 | small e, dieresis or umlaut mark | ë |
| 236 | small i, grave accent | ì |
| 237 | small i, acute accent | í |
| 238 | small i, circumflex accent | î |
| 239 | small i, dieresis or umlaut mark | ï |
| 240 | small eth, Icelandic | ð |
| 241 | small n, tilde | ñ |
| 242 | small o, grave accent | ò |
| 243 | small o, acute accent | ó |
| 244 | small o, circumflex accent | õ |
| 245 | small o, tilde | õ |
| 246 | small o, dieresis or umlaut mark | ö |
| 247 | division sign | ÷ |
| 248 | small o, slash | ø |
| 249 | small u, grave accent | ù |
| 250 | small u, acute accent | ú |
| 251 | small u, circumflex accent | û |
| 252 | small u, dieresis or umlaut mark | ü |
| 253 | small y, acute accent | ý |
| 254 | small thorn, Icelandic | þ |
| 255 | small y, dieresis or umlaut mark | ÿ |

### Unicode

Unicode 是最终的字符编码，它是一个基于 ASCII 和 ISO8859-1 的国际标准，UNICODE 要求 16 位的字符，因为所有的字符都有它们的固有码。目前，已经定义了超过 30,000 个不同的字符。不过，不是所有的字符图像能在μC/ GUI 中定义。定义这些附加的字符是使用者的工作。请联系 Micrium 公司，或你的发行人，因为我们可能有你所需要的字符。

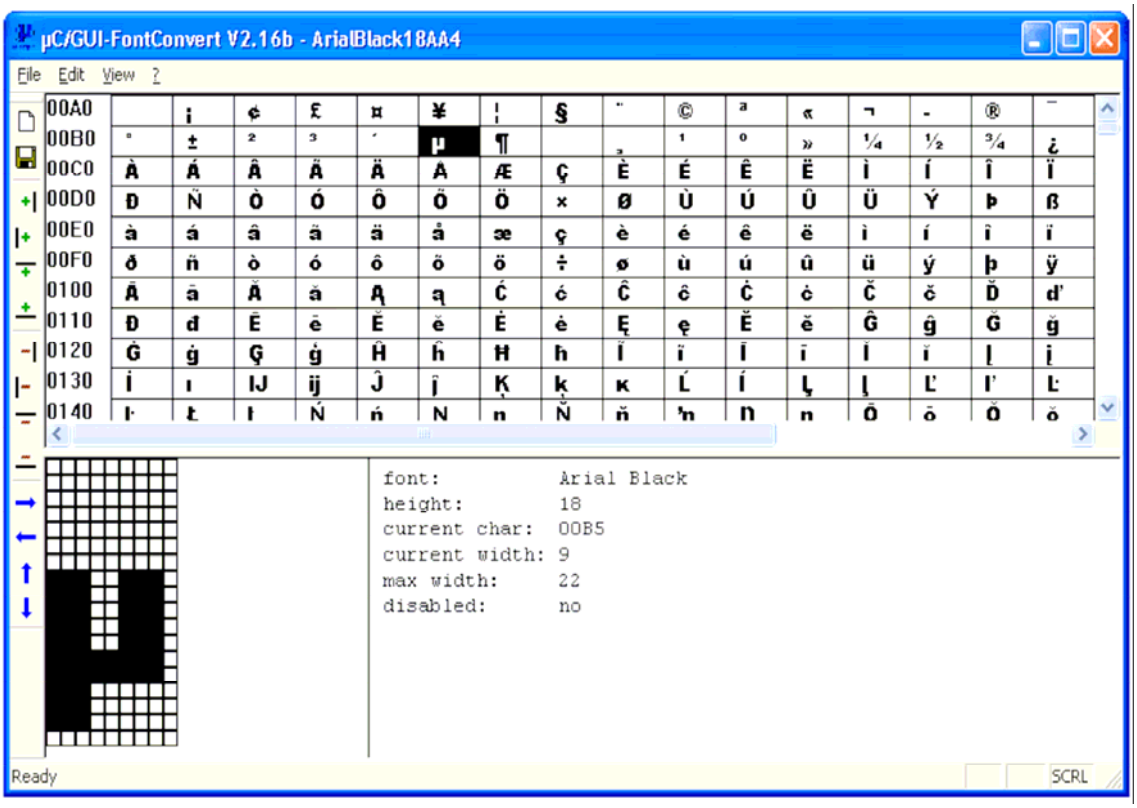
## 7.6 字体转换器

能用于μC/GUI 的字体必须在“C”中定义为 GUI_FONT 结构。该结构或者由这些结构引用的与之相当的字体数据，可能相当大。手工产生这些字体时耗时巨大且效率很低。因此我们推荐使用字体转换器，它能够从字体自动产生“C”文件。

字体转换器是一个简单的 Windows 程序。你只需在程序载入一种 Windows 安装字体，如果有需要则对其进行编辑，然后将其保存为“C”文件。该“C”文件可以进行编译，你需要

的字体就可以随μC/GUI 一道在屏幕上显示。

默认情况下字符代码 0x00～0x1F 及 0x80～0x9F 是被禁止的。下面为字体转换器截入一种字体的范例的屏幕截图：




字体转换器在一份单独的资料中描述，你可以与 Micrium 公司联系以获得这份资料。