第4章 文本显示

使用μC/GUI 显示字体是很容易的。仅仅需要很少的的函数知识就能让我们在任何有效的字体当中进行文本书字，然后显示在任何一个位置。我们首先对显示字体进行简短的介绍，然后是分别对所用的函数进行更详细的说明。

4.1 基本函数

为了在LCD上显示文本，可以简单地调用函数GUI_DispString()，把你所希望显示的文本作为其参数，例如：

```
GUI_DispString("Hello world!");
```

上面的代码将会在当前文本坐标显示文本“Hello world!”。

然而，正如你所看到的，有很多函数用于显示不同字体的文本或都在不同的坐标显示文本。另外，它不仅能写字符串，而且能写十进制数，十六进制数和二进制数用于显示。即使图形显示通常是以字节为导向，文本能够定位在显示屏上的任何像素上，不仅仅是按字节定位。

控制字符

控制字符是一个小于32的字符代码。控制字符被定义为ASCII码的一部分。μC/GUI忽略所有除了下表所列出的以外的控制字符：

字符代码	ASCII代码	“C”	含义
10	LF	\n	换行，改变当前文本坐标到下一行，即： X=0； Y += 字体-距离（单位：像素）（如函数GUI_GetFontDistY()所讨论的那样）
13	CR	\r	回车，改变当前文本坐标到当前行的开始处，即：x=0

控制字符LF的用法在字符串中非常方便。换行能将一个字符串拆开几部分，这样，只需要调用一个函数就能将这个字符串就能变成几行显示。

在一个选定坐标放置文本

这个功能可能通过调用函数GUI_GotoXY()来实现，如下面例子所示：

```
GUI_GotoXY(10,10);                  // 设置坐标（以像素为单位）
GUI_DispString("Hello world!");     // 显示文本
```

4.2 文本API

下表列出了与文本处理相关的函数，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
显示文本的函数	
GUI_DispChar()	在当前坐标显示单个字符
GUI_DispCharAt()	在指定坐标显示单个字符
GUI_DispChars()	按指定重复次数显示一个字符
GUI_DispString()	在当前坐标显示字符串
GUI_DispStringAt()	在指定坐标显示字符串
GUI_DispStringAtCEOL()	在指定坐标显示字符串，并清除到行末
GUI_DispStringInRect()	在指定矩形区域内显示字符串
GUI_DispStringLen()	在当前坐标显示指定字符数量的字符串
选择文本绘图模式	
GUI_SetTextMode()	设置文本绘图模式
选择文本对齐方式	
GUI_GetTextAlign()	返回当前文本对齐模式
GUI_SetLBorder()	设置换行后的左边界
GUI_ SetTextAlign()	设置文本对齐模式
设置当前文本坐标	
GUI_GotoX()	设置当前X坐标
GUI_GotoXY()	设置当前X、Y坐标
GUI_GotoY()	设置当前Y坐标
找回当前文本坐标	
GUI_GetDispPosX()	返回当前X坐标
GUI_GetDispPosY()	返回当前Y坐标
清除视窗或其部分的函数	
GUI_Clear()	清除活动视窗（如果背景是活动视窗，则是清除整个屏幕）
GUI_DispCEOL()	清除从当前坐标到行末的显示内容

4.3 显示文本的函数

GUI_DispChar()

描述

在当前视窗使用当前字体在当前文本坐标处显示单个字符。

函数原型

void GUI_DispChar(U16 c);

参 数	含 意
c	显示的字符

附加信息

这是显示字符的基本函数。所有其它显示函数（GUI_DispCharAt()，GUI_DispString()等）都要调用这个函数输出单个字符。

字符是否有效取决于所选择的字体，如果在当前字体中该字符无效，则不会有任何显示。

范例

在屏幕上显示一个大写“A”：

```
GUI_DispChar('A');
```

相关主题

GUI_DispChars()，GUI_DispCharAt()

GUI_DispCharAt()

描述

在当前视窗使用当前字体在指定坐标处显示单个字符。

函数原型

void GUI_DispCharAt(U16 c, I16P x, I16P y);

参 数	含 意
c	显示的字符
x	写到客户窗口经X轴坐标（以像素为单位）
y	写到客户窗口经Y轴坐标（以像素为单位）

附加信息

显示字符的左上角在指定的（X,Y）坐标。
使用函数GUI_DispChar()写字符。

如果在当前字体中该字符无效，则不会有任何显示。

范例

在屏幕左上角显示一个大写“A”：

```
GUI_DispCharAt('A',0,0);
```

相关主题

GUI_DispChar()，GUI_DispChars()

GUI_DispChars()

描述

在当前视窗使用当前字体在当前文本坐标显示一个字符，并指定重复显示的次数。

函数原型

```
void GUI_DispChars(U16 c, int Cnt);
```

参　数	含　意
c	显示的字符
Cnt	重复的次数（0≤Cnt≤32767）

附加信息

使用函数GUI_DispChar()写字符。
如果在当前字体中该字符无效，则不会有任何显示。

范例

在屏幕上显示一行“**********”：

```
GUI_DispChars('*', 30);
```

相关主题

GUI_DispChar()，GUI_DispCharAt()

GUI_DispString()

描述

在当前视窗的当前坐标，使用当前字体显示作为参数的字符串。

函数原型

```
void GUI_DispString(const char GUI_FAR *s);
```

参　数	含　意
s	显示的字符串

附加信息

字符串包括控制字符“\n”。该控制字符把当前文本坐标移到下一行的开始处。

范例

在屏幕上显示“Hello world”及在下一行显示“Next line”：

```
GUI_DispString("Hello world");       // 显示文本
GUI_DispString("\nNext line");       // 显示文本
```

相关主题

GUI_DispStringAt(), GUI_DispStringAtCEOL(), GUI_DispStringLen()

GUI_DispStringAt()

描述

在当前视窗，使用当前字体在指定坐标显示作为参数的字符串。

函数原型

```
void GUI_DispStringAt(const char GUI_FAR *s, int x, int y);
```

参　数	含　意
s	显示的字符串
x	写到客户视窗的X轴坐标（以像素为单位）

y	写到客户视窗的Y轴坐标（以像素为单位）

范例

在屏幕上坐标（50,20）处显示“Position 50,20”

```
GUI_DispStringAt("Position 50,20", 50, 20);       // 显示文本
```

相关主题

GUI_DispString(), GUI_DispStringAtCEOL(), GUI_DispStringLen()

GUI_DispStringAtCEOL()

描述

该函数使用的参数与GUI_DispStringAt()完全一致。它也执行同样的操作：在指定坐标显示所给出的字符串。但是，完成这步操作后，它会调用GUI_DispCEOL函数清除本行剩下部分内容直至行末。如果字符串覆盖了其它的字符串，同时该字符串长度比原先的字符串短的时候，使用该函数就会很方便。

GUI_DispStringInRect()

描述

在当前视窗，使用当前字体在指定坐标显示作为参数的字符串。

函数原型

```
void GUI_DispStringInRect ( const char GUI_FAR *s,
                            const GUI_RECT *pRect,
                            int Align);
```

参　数	含　意
s	显示的字符串
pRect	写像素的客户窗口的矩形区域
Align	垂直对齐：GUI_TA_TOP，GUI_TA_BOTTOM，GUI_TA_VCENTER； 水平对齐：GUI_TA_LEFT，GUI_TA_RIGHT，GUI_TA_HCENTER；

范例

在当前视窗的水平及垂直对中的坐标显示字“Text”：

```
GUI_RECT rClient;
GUI_GetClientRect(&rClient);
GUI_DispStringInRect("Text", &rClient, GUI_TA_HCENTER | GUI_TA_VCENTER);
```

附加信息

如果指定的矩形太小，文本会被裁剪。

相关主题

GUI_DispString()，GUI_DispStringAtCEOL()，GUI_DispStringLen()

GUI_DispStringLen()

描述

在当前视窗，使用当前字体在指定坐标显示作为参数的字符串，指定显示字符的数量。

函数原型

```
void GUI_DispStringLen(const char GUI_FAR *s, int Len);
```

参　数	含　意
s	显示的字符串，应该以一个“\0”作为8位字符排列的结束标记。允许用NULL作为参数。
Len	显示的字符数量

附加信息

如果字符串的字符少于指定的数量，则用空格填满。如果多于指定的数量，则显会显示指定数量的字符。

文本信息可能以不同语言显示（自然长度会不一样）时，该函数十分有用，但是只有某些字符能够显示。

相关主题

GUI_DispString()，GUI_DispStringAt()，GUI_DispStringAtCEOL()

4.4 选择文本的绘制模式

通常，在当前文本坐标，使用所选择的字体，在选择视窗中以正常文本模式定入文本。正常文本意思是指，文本覆盖已经显示的任何东西，在这种情况下，在字符屏蔽中被设定的位在屏幕上被设定。在这种模式下，活动的位使用前景色写，而非活动的位用背景色写。然而，在一些场合，需要改变这些默认的行为。为了这个目的，μC/GUI提供四种标识（一种默认加上三种修改值），它们可以组合使用：

正常文本

文本可能正常显示，此时模式标识应指定为GUI_TEXTMODE_NORMAL 或 0。

反转文本

文本反转显示，模式标识应指定为GUI_TEXTMODE_REVERSE。通常在黑色上显示白色变成在白色上显示的黑色。

透明文本

透明文本意思是文本写在已经在屏幕上可见的任何东西上面。不同的是，屏幕上原有的内容仍然能够看得见，与正常文本相比，背景色被擦除了。

模式标识指定为GUI_TEXTMODE_TRANS表示显示透明文本。

异或文本

通常情况下，用白色绘制的（实际字符）显示是反相的。如果背景颜色是黑色，效果与正常模式（正常文本）是一样的。如果背景是白色，输出与反转文本一样。如果你使用彩色，一个反相的像素由下式计算：

新像素颜色 = 颜色的值 - 实际像素颜色 - 1

透明反转文本

作为透明文本，它不覆盖背景，作为反转文本，文本显示是反转的。

文本通过指定标识GUI_TEXTMODE_TRANS |GUI_TEXTMODE_REVERSE来实现这种效果。

范例

显示正常，反转，透明，异或及透明反转文本：

```
GUI_SetFont(&GUI_Font8x16);
GUI_SetFont(&GUI_Font8x16);
GUI_SetBkColor(GUI_BLUE);
GUI_Clear();
GUI_SetPenSize(10);
GUI_SetColor(GUI_RED);
GUI_DrawLine(80, 10, 240, 90);
GUI_DrawLine(80, 90, 240, 10);
GUI_SetBkColor(GUI_BLACK);
GUI_SetColor(GUI_WHITE);
GUI_SetTextMode(GUI_TM_NORMAL);
GUI_DispStringHCenterAt("GUI_TM_NORMAL", 160, 10);
GUI_SetTextMode(GUI_TM_REV);
GUI_DispStringHCenterAt("GUI_TM_REV", 160, 26);
GUI_SetTextMode(GUI_TM_TRANS);
GUI_DispStringHCenterAt("GUI_TM_TRANS" , 160, 42);
GUI_SetTextMode(GUI_TM_XOR);
GUI_DispStringHCenterAt("GUI_TM_XOR" , 160, 58);
GUI_SetTextMode(GUI_TM_TRANS | GUI_TM_REV);
GUI_DispStringHCenterAt("GUI_TM_TRANS | GUI_TM_REV", 160, 74);
```

下图为上面范例程序执行结果的的屏幕截图

GUI_SetTextMode()

描述

按照指定的参数设置文本模式。

函数原型

```
int GUI_SetTextMode(int TextMode);
```

参　数	含　意
TextMode	设置的文本模式，可以是文本模式标识的任意组合

参数TextMode允许的数值（可以用“OR（或）”进行组合）

GUI_TEXTMODE_NORMAL	设置正常文本，这是默认的设置，该数值等同于0
GUI_TEXTMODE_REVERSE	设置反转文本
GUI_TEXTMODE_TRANSPARENT	设置透明文本
GUI_TEXTMODE_XOR	反相显示的文本

返回值

选择的文本模式

范例

屏幕上坐标（0,0）处显示“The value is”，设置文本模式为反转模式，再将其设回正常模式：

```
int i = 20;
GUI_DispStringAt("The value is", 0, 0);
GUI_SetTextMode(GUI_TEXTMODE_REVERSE);
GUI_DispDec(20, 3);
GUI_SetTextMode(GUI_TEXTMODE_NORMAL);
```

4.5 文本对齐的选择

GUI_GetTextAlign()

描述

返回当前文本对齐模式。

函数原型

```
int GUI_GetTextAlign(void);
```

GUI_SetLBorder()

描述

设置在当前视窗换行后的左边界

函数原型

```
void GUI_SetLBorder(int x)
```

参　数	含　意
x	新的左边界（以像素为单位，0表示视窗左边界）

GUI_SetTextAlign()

描述

设置文本对齐模式，用于当前视窗的字符串输出。

函数原型

```
int GUI_SetTextAlign(int TextAlign);
```

参　数	含　意
TextAlign	设定的文本对齐模式。可以是水平和垂直对齐标志的组合。

参数TextAlign允许的数值（水平和垂直标识以“OR”组合）

水平对齐	
GUI_TA_LEFT	X轴方向左对齐（默认）
GUI_TA_HCENTER	X轴方向对中
GUI_TA_RIGHT	X轴方向右对齐（默认）
垂直对齐	
GUI_TA_TOP	在字符Y轴方向顶部对齐（默认）
GUI_TA_VCENTER	在Y轴方向对中
CUI_TA_BOTTOM	在字体Y轴底部线像素对齐

返回值

所选择的文本对齐模式

附加信息

GUI_SetTextAllign()不影响以源于GUI_DispChar的字符输出函数。

范例

在坐标（100,100）处显示数值1234，采用对中模式：

```
GUI_SetTextAlign(GUI_TA_HCENTER | GUI_TA_VCENTER);
GUI_DispDecAt(1234,100,100,4);
```

4.6 设置当前文本坐标

每个任务都有一个当前文本坐标，该坐标以视窗的原点（通常是(0,0)）为参考，如果调用了文本输出函数，下一个字符会写在这个坐标上。初始化时，该坐标是（0,0），即当前视窗的左上角。在三个函数可能用来设置当前文本坐标。

GUI_GotoXY(), GUI_GotoX(), GUI_GotoY()

描述

设置当前写文本的坐标。

函数原型

```
char GUI_GotoXY(int x, int y);
char GUI_GotoX(int x);
```

```
char GUI_GotoY(int y);
```

参　数	含　意
x	新的X轴坐标（以像素为单位，0为视窗左边界）
y	新的Y轴坐标（以像素为单位，0为视窗顶部边界）

返回值

通常为0。如果返回数值非0，则当前文本坐标超出视窗范围（到了右边或下边），这样紧接着的写操作可能被忽略。

附加信息

GUI_GotoXY()对当前视窗文本坐标的X坐标和Y坐标两部分同时设置。
GUI_GotoX()只对当前视窗文本坐标的X坐标部分进行设置，Y坐标保持不变。
GUI_GotoY()只对当前视窗文本坐标的Y坐标部分进行设置，X坐标保持不变。

范例

在屏幕上坐标（20,20）处显示“(20,20)”：

```
GUI_GotoXY(20,20)
GUI_DispString("The value is");
```

4.7 找回当前文本坐标

GUI_GetDispPosX()

描述

返回当前X坐标

函数原型

```
int GUI_GetDispPosX(void);
```

GUI_GetDispPosY()

描述

返回当前Y坐标

函数原型

```
int GUI_GetDispPosY(void);
```

4.8 清除一个视窗或其部分的函数

GUI_Clear()

描述

清除当前视窗。

函数原型

```
void GUI_Clear(void);
```

附加信息

如果没有定义视窗，当前视窗为整个显示区。这样的话，整个显示区都会被清除。

范例

在屏幕上显示“Hello world”，等待1秒种，然后清除显示内容：

```
GUI_DispStringAt("Hello world", 0, 0);        // 显示文本
GUI_Delay(1000);                              // 等待1秒钟（非μC/GUI部分）
GUI_Clear();                                  // 清屏
```

GUI_DispCEOL()

描述

清除当前视窗（或屏幕）从当前文本坐标到行末显示区域的内容，高度为当前字体高度。

函数原型

```
void GUI_DispCEOL(void);
```

范例

在屏幕上显示“Hello world”，等待1秒钟，然后在同步坐标显示“Hi”，代替原先的显示的字符：

```
GUI_DispStringAt("Hello world", 0, 0);          // 显示文本
Delay(1000);
GUI_DispStringAt("Hi", 0, 0);
GUI_DispCEOL();
```