# ZLG/GUI 图形用户界面

## 1.1 概述

GUI 为 Graphics User Interface 的简写，即图形用户界面，这是用于提高人机交互友好性、易操作性的计算机程序，它是建立在计算机图形学基础上的产物。图形用户界面是当今计算机技术的重大成就之一，它极大地方便了非专业用户的使用，人们不再需要死记硬背大量的命令，而是通过窗口、菜单方便地进行操作。

随着嵌入式系统的日益发展，32 位嵌入式处理器及图形显示设备的广泛应用，目标产品对 GUI 的需求越来越多。由于嵌入式系统的资源有限，所以对 GUI 的要求是可载剪的，高速度的。ZLG/GUI 是占用资源小、使用方便的嵌入式系统简易的图形用户界面软件。ZLG/GUI 提供了最基本的画点、线、圆形、圆弧、椭圆形、矩形、正方形、填充等功能，较高级的接口功能有 ASCII 显示、汉字显示、图标显示、窗口、菜单等，支持单色、灰度、伪彩、真彩等图形显示设备。

## 1.2 ZLG/GUI 的文件

在这节里将统一介绍 ZLG/GUI 的接口函数及可用资源。接口函数是按其功能分类的，并且分别编写到不同的文件中，如下所示：

| | | |
|---|---|---|
| 基本图形操作函数 | —— | GUI_BASE.C |
| 显示颜色管理函数 | —— | GUI_STOCKC.C |
| 颜色转换操作函数 | —— | CONVERTCOLOR.C |
| 5×7ASCII 码字库及显示函数 | —— | FONT5_7.C |
| 8×8ASCII 码字库及显示函数 | —— | FONT8_8.C |
| 24×32 数字库及显示函数 | —— | FONT24_32.C |
| 单色图形及汉字显示函数 | —— | LOADBIT.C |
| 图标菜单、下拉菜单操作函数 | —— | MENU.C |
| 窗口操作函数 | —— | WINDOW.C |

其它重要文件说明如下：

CONFIG.H——用于声明常用宏，包含所有项目所用的头文件。(方便项目的管理)

GUI_CONFIG.H——用于配置 ZLG/GUI。(用于裁剪 ZLG/GUI)

FONT_MACRO.H——定义字节点阵宏。(用于定义字体点阵数据)

## 1.3 ZLG/GUI 的配置

在 ZLG/GUI 的 GUI_CONFIG.H 文件中进行 ZLG/GUI 的配置，功能配置说明如下：

- **GUI_LineWith_EN**

画有宽度的直线函数 GUI_LineWith()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_CircleX_EN**

画圆函数 GUI_Circle()、GUI_CircleFill()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_EllipseX_EN**

画椭圆函数 GUI_Ellipse()、GUI_EllipseFill()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_FloodFill_EN**

填充函数 GUI_FloodFill()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。当使用填充函数时，可以定义 DOWNP_N、UPP_N 宏来设置向上及向下折点个数，这两个宏用于定义保存向上及向下折点数据的数组大小。

- **GUI_ArcX_EN**

画圆弧函数 GUI_Arc4()、GUI_Arc 使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_Pieslice_EN**

扇形函数 GUI_Pieslice()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_WINDOW_EN**

窗口管理函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。窗口管理函数如 GUI_WindowsDraw()、GUI_WindowsHide()、GUI_WindowsClr()等。由于窗口管理函数使用到 5×7 ASCII 字体显示，所以必需同时设置 FONT5x7_EN 为 1。

- **GUI_MenuIco_EN**

图标菜单操作函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。图标菜单操作函数如 GUI_MenuIcoDraw() 、 GUI_Button49x14() 、 GUI_Button_OK() 、 GUI_Button_Cancle()、GUI_Button_OK1()、GUI_Button_Cancle1()等。

- **GUI_MenuDown_EN**

下拉菜单操作函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。下拉菜单操作函数如 GUI_MMenuDraw() 、 GUI_MMenuSelect() 、 GUI_MMenuNSelect() 、 GUI_SMenuDraw ()、GUI_SMenuSelect ()、GUI_SMenuHide ()等。

- **FONT5x7_EN**

5×7 ASCII 码字库及显示函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。5×7 ASCII 码字符显示函数如 GUI_PutChar()、GUI_PutString()、GUI_PutNoStr()等。

- **FONT8x8_EN**

8×8 ASCII 码字库及显示函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。8×8 ASCII 码字符显示函数如 GUI_PutChar8_8 ()、GUI_PutString8_8 ()、GUI_PutNoStr8_8 ()等。

- **FONT24x32_EN**

24×32 数字库及显示函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。24×32 数字字符显示函数如 GUI_PutChar24_32()。

- **GUI_PutHZ_EN**

汉字显示函数 GUI_PutHZ()使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。

- **GUI_LoadPic_EN**

单色图形显示函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。单色图形显示函数如 GUI_LoadPic()、GUI_LoadPic1()。

- **CONVERTCOLOR_EN**

颜色转换操作函数使能控制，设置为 1 时函数有效，为 0 或其它值时函数禁止。颜色转换操作函数如 GUI_Color2Index_565()、GUI_Color2Index_555()、GUI_Color2Index_444()、GUI_Color2Index_332 等等。

## 1.4 ZLG/GUI 函数手册

ZLG/GUI 函数表如表 1～表 11 所列。以下函数原型中，uint8 已定义为 unsigned char，uint32 已定义为 unsigned int。

表 1 硬件驱动层接口函数(LCDDRIVE.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_Initialize(void) | 无 | 初始化 GUI(缓冲区及 LCM) |
| void GUI_FillSCR(TCOLOR dat) | dat 填充的颜色值数据 | 全屏填充 |
| Void GUI_ClearSCR(void) | 无 | 清屏 |
| uint8 GUI_Point(uint32 x, uint32 y,<br>TCOLOR color) | x，y 点的坐标<br>color 显示颜色 | 画点(返回值为 1 时表示操作成功，为 0 表示失败) |
| int GUI_ReadPoint(uint32 x, uint32 y,<br>TCOLOR *ret); | x，y 点的坐标<br>ret 保存变量的指针 | 读取指定点的颜色(返回值为 0 时表示操作失败) |
| void GUI_HLine(uint32 x0, uint32 y0,<br>uint32 x1,<br>TCOLOR color) | x0，y0 起点坐标<br>x1 水平线终点 x 值<br>color 显示颜色 | 画水平线 |
| void GUI_RLine(uint32 x0, uint32 y0,<br>uint32 y1,<br>TCOLOR color) | x0，y0 起点坐标<br>y1 垂直线终点 y 值<br>color 显示颜色 | 画垂直线 |
| int GUI_CmpColor(TCOLOR color1,<br>TCOLOR color2) | color1 颜色值 1<br>color2 颜色值 2 | 比较两个颜色值是否相等，相等返回 1，否则返回 0 |
| void GUI_CopyColor(TCOLOR *color1,<br>TCOLOR color2) | color1 目标变量指针<br>color2 原颜色值 | 颜色值复制，即*color1= color2 |

**可用资源：**LCDDRIVE.H **中定义宏** TCOLOR、GUI_LCM_XMAX **及** GUI_LCM_YMAX，TCOLOR **为颜色数据类型**，GUI_LCM_XMAX **和** GUI_LCM_YMAX **为** LCM **点像素数**。

表 2 基本图形操作函数(GUI_BASE.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_Line(uint32 x0, uint32 y0,<br>uint32 x1, uint32 y1,<br>TCOLOR color) | x0，y0 起点坐标<br>x1，y1 终点坐标<br>color 显示颜色 | 画任意两点之间的直线 |
| void GUI_LineWith(uint32 x0, uint32 y0,<br>uint32 x1, uint32 y1,<br>uint8 with,<br>TCOLOR color) | x0，y0 起点坐标<br>x1，y1 终点坐标<br>with 线宽大小<br>color 显示颜色 | 画具有线宽的任意两点之间的直线(with 值最大为 50) |
| void GUI_LineS(uint32 const *points,<br>uint8 no,<br>TCOLOR color) | *points 顶点坐标值指针<br>no 顶点数<br>color 显示颜色 | 画多边形(即多个顶点之间的连续连线) |
| void GUI_Rectangle(uint32 x0, uint32 y0,<br>uint32 x1, uint32 y1,<br>TCOLOR color) | x0，y0 矩形左上角坐标<br>x1，y1 矩形右下角坐标<br>color 显示颜色 | 画矩形 |
| void GUI_RectangleFill(uint32 x0, uint32 y0,<br>uint32 x1, uint32 y1,<br>TCOLOR color) | x0，y0 矩形左上角坐标<br>x1，y1 矩形右下角坐标<br>color 填充颜色 | 画填充矩形 |
| void GUI_Square(uint32 x0, uint32 y0,<br>uint32 with,<br>TCOLOR color) | x0，y0 正方形左上角坐标<br>with 边长<br>color 显示颜色 | 画正方形 |

**接上表**

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_Circle(uint32 x0, uint32 y0,<br>uint32 r,<br>TCOLOR color) | x0，y0 圆形的圆心坐标<br>r 圆形的半径<br>color 显示颜色 | 画圆形 |
| void GUI_CircleFill(uint32 x0, uint32 y0,<br>uint32 r,<br>TCOLOR color) | x0，y0 圆形的圆心坐标<br>r 圆形的半径<br>color 填充颜色 | 画填充圆形 |
| void GUI_Ellipse(uint32 x0, uint32 x1,<br>uint32 y0, uint32 y1,<br>TCOLOR color) | x0，x1 椭圆形的 x 坐标<br>y0，y1 椭圆形的 y 坐标<br>color 显示颜色 | 画椭圆形(x0、x1 分别为椭圆最左和最右的点的 x 坐标；y0、y1 分别为椭圆最上和最下的点的 y 坐标) |
| void GUI_EllipseFill(uint32 x0, uint32 x1,<br>uint32 y0, uint32 y1,<br>TCOLOR color) | x0，x1 椭圆形的 x 坐标<br>y0，y1 椭圆形的 y 坐标<br>color 填充颜色 | 画填充椭圆形(x0、x1 分别为椭圆最左和最右的点的 x 坐标；y0、y1 分别为椭圆最上和最下的点的 y 坐标) |
| void GUI_FloodFill(uint32 x0, uint32 y0,<br>TCOLOR color) | x0，y0 指定填充点坐标<br>color 填充颜色 | 图形填充(在 GUI_CONFIG.H 中配置向上及向下折点个数) |
| void GUI_Arc4(uint32 x, uint32 y,<br>uint32 r,<br>uint8 angle,<br>TCOLOR color) | x，y 圆弧圆心坐标<br>r 圆弧半径<br>angle 圆弧角度<br>color 显示颜色 | 画 1/4 圆弧(angle 为 1～4，即 0～90 度、90～180 度、180～270 度、270～360 度) |
| void GUI_Arc(uint32 x, uint32 y,<br>uint32 r,<br>uint32 stangle,<br>uint32 endangle,<br>TCOLOR color) | x，y 圆弧圆心坐标<br>r 圆弧半径<br>stangle 圆弧起始角度<br>endangle 圆弧终止角度<br>color 显示颜色 | 画任意角度圆弧(stangle、endangle 为 0～359 度) |
| void GUI_Pieslice(uint32 x, uint32 y,<br>uint32 r,<br>uint32 stangle,<br>uint32 endangle,<br>TCOLOR color) | x0，y0 扇形圆心坐标<br>r 扇形半径<br>stangle 扇形起始角度<br>endangle 扇形终止角度<br>color 显示颜色 | 画扇形(stangle、endangle 为 0～359 度) |

**可用资源**：GUI_BASIC.H **中定义数据结构** PointXY，**结构变量中包含点的坐标变量** x、y，**数据类型均为** uint32，**如程序清单** 1 **所示**。

**程序清单** 1 **点数据结构定义**

```
/* 定义坐标数据结构 */
typedef  struct
         {  uint32  x;                    // x 坐标变量
            uint32  y;                    // y 坐标变量
         }
         PointXY;
```

表 3 显示颜色管理函数(GUI_STOCKC.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_SetColor(TCOLOR color1, TCOLOR color2) | color1 前景色<br>color2 背景色 | 设置前景色及背景色 |
| void GUI_GetBackColor(TCOLOR *bakc) | *bakc 保存变量的指针 | 读取背景色的值 |
| void GUI_GetDispColor(TCOLOR *bakc) | * bakc 保存变量的指针 | 读取前景色的值 |
| void GUI_ExchangeColor(void) | 无 | 交换前景色与背景色 |

表 4 窗口管理函数(WINDOWS.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint8 GUI_WindowsDraw(WINDOWS *win) | *win 窗口句柄 | 显示窗口 |
| uint8 GUI_WindowsHide(WINDOWS *win) | *win 窗口句柄 | 消隐窗口 |
| void GUI_WindowsClr(WINDOWS *win) | *win 窗口句柄 | 清屏窗口(即清屏窗口用户区域) |

窗口数据结构 WINDOWS 如程序清单 2 所示。

程序清单 2 窗口数据结构

```
/* 定义窗口数据结构 */
typedef  struct
{  uint32  x;              // 窗口位置(左上角的 x 坐标)
   uint32  y;              // 窗口位置(左上角的 y 坐标)

   uint32  with;           // 窗口宽度
   uint32  hight;          // 窗口高度

   uint8   *title;         // 定义标题栏指针 (标题字符为 ASCII 字符串，最大个数受窗口限制)
   uint8   *state;         // 定义状态栏指针 (若为空时则不显示状态栏)
} WINDOWS;
```

表 5 图标菜单操作函数(MENU.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_Button49x14(uint32 x, uint32 y, uint8 *dat) | x，y 显示按钮起始坐标<br>*dat 按钮的点阵数据 | 显示按钮(按钮大小为 49*14) |
| void GUI_Button_OK(uint32 x, uint32 y) | x，y 显示按钮起始坐标 | 显示 OK 按钮 |
| void GUI_Button_OK1(uint32 x, uint32 y) | x，y 显示按钮起始坐标 | 显示选中状态的 OK 按钮 |
| void GUI_Button_Cancle(uint32 x, uint32 y) | x，y 显示按钮起始坐标 | 显示 CANCLE 按钮 |
| void GUI_Button_Cancle1(uint32 x, uint32 y) | x，y 显示按钮起始坐标 | 显示选中状态的 CANCLE 按钮 |
| uint8 GUI_MenuIcoDraw(MENUICO *ico) | *ico 图标菜单句柄 | 显示图标菜单 |

图标菜单数据结构 MENUICO 如程序清单 3 所示。

**程序清单 3 图标菜单数据结构**

```
/* 定义图标菜单数据结构 */
typedef  struct
{  uint32  x;                    // 图标菜单位置(左上角的 x 坐标)
   uint32  y;                    // 图标菜单位置(左上角的 y 坐标)
   uint8   *icodat;              // 32*32 的 ICO 数据地址
   uint8   *title;               // 相关标题提示 (42*13)
   uint8   state;                // 图标菜单状态，为 0 时表示未选中，为 1 时表示已选中
   void    (*Function)(void);    // 对应的服务程序
} MENUICO;
```

**表 6 下拉菜单操作函数**(MENU.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint8 GUI_MMenuDraw(MMENU *men) | *men 主菜单句柄 | 显示主菜单 |
| void GUI_MMenuSelect(MMENU *men,<br>uint8 no) | *men 主菜单句柄<br>no 选中的主菜单项 | 选择主菜单项(no 为 0～n) |
| void GUI_MMenuNSelect(MMENU *men,<br>uint8 no) | *men 主菜单句柄<br>no 取消选中的主菜单项 | 取消主菜单项的选择(no 为 0～n) |
| uint8 GUI_SMenuDraw(SMENU *men) | *men 子菜单句柄 | 显示子菜单 |
| void GUI_SMenuSelect(SMENU *men,<br>uint8 old_no,<br>uint8 new_no) | *men 子菜单句柄<br>old_no 原选中的子菜单项<br>new_no 新选中的子菜单项 | 选择子菜单项(选取消 old_no 项的选中状态，然后选择 new_no 项) |
| uint8 GUI_SMenuHide(SMENU *men) | *men 子菜单句柄 | 消隐子菜单 |

**下拉式菜单数据结构** MMENU、SMENU **以及菜单规格如程序清单** 4 **所示**。

**程序清单 4 下拉式菜单数据结构**

```
/* 定义主菜单宽度，及最大菜单个数 */
#define  MMENU_WIDTH          34
#define  MMENU_NO             6

/* 定义菜单的宽度(下拉菜单)，及最大子菜单个数 */
#define  SMENU_WIDTH          66
#define  SMENU_NO             8

/* 定义一子菜单项的数据结构 */
typedef  struct
{  WINDOWS  *win;                    // 所属窗口
   uint8    mmenu_no;                // 对应的主菜单项号(0-n)

   uint8    no;                      // 子菜单项个数
   char     *str[SMENU_NO];          // 子菜单字符串
```

```
    uint8       state;                          // 所选择的子菜单

    void        (*Function[SMENU_NO])(void);    // 子菜单对应的服务程序
} SMENU;

/* 主菜单数据结构 */
typedef    struct
{   WINDOWS    *win;                            // 所属窗口

    uint8       no;                             // 主菜单个数
    char        *str[MMENU_NO];                 // 主菜单字符串
} MMENU;
```

**表 7　图形及汉字操作函数**(LOADBIT.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_LoadPic (uint32 x, uint32 y,<br>uint8 *dat,<br>uint32 hno, uint32 lno) | x，y 显示的起始坐标<br>*dat 点阵数据指针<br>hno，lno 图形的行/列数 | 显示单色点阵图形 |
| void GUI_LoadPic1 (uint32 x, uint32 y,<br>uint8 *dat,<br>uint32 hno, uint32 lno) | x，y 显示的起始坐标<br>*dat 点阵数据指针<br>hno，lno 图形的行/列数 | 显示单色点阵图形(反相显示) |
| void GUI_PutHZ(uint32 x, uint32 y,<br>uint8 *dat,<br>uint8 hno, uint8 lno) | x，y 显示的起始坐标<br>*dat 点阵数据指针<br>hno，lno 汉字的行/列数 | 汉字显示 |

**表 8　5×7 ASCII 码字符显示函数**(FONT5_7.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint8 GUI_PutChar(uint32 x, uint32 y,<br>uint8 ch) | x，y 字符显示的坐标<br>ch 字符的十六进制值 | 显示一个字符(返回值为 1 时表示操作成功，为 0 表示失败) |
| void GUI_PutString(uint32 x, uint32 y,<br>char *str) | x，y 显示的起始坐标<br>*str 指向字符串的指针 | 显示一字符串(以'\0'结束，没有自动换行功能) |
| void GUI_PutNoStr(uint32 x, uint32 y,<br>char *str,<br>uint8 no) | x，y 显示的起始坐标<br>*str 指向字符串的指针<br>no 要显示字符的个数 | 显示字符串中指定个数的字符 |

**表 9　8×8 ASCII 码字符显示函数**(FONT8_8.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint8 GUI_PutChar8_8(uint32 x, uint32 y,<br>uint8 ch) | x，y 字符显示的坐标<br>ch 字符的十六进制值 | 显示一个字符(返回值为 1 时表示操作成功，为 0 表示失败) |
| void GUI_PutString8_8(uint32 x, uint32 y,<br>char *str) | x，y 显示的起始坐标<br>*str 指向字符串的指针 | 显示一字符串(以'\0'结束，没有自动换行功能) |

**接上表**

| 函数原型 | 参数 | 功能 |
|---|---|---|
| void GUI_PutNoStr8_8(uint32 x, uint32 y,<br>char *str,<br>uint8 no) | x，y **显示的起始坐标**<br>*str **指向字符串的指针**<br>no **要显示字符的个数** | **显示字符串中指定个数的字符** |

**表** 10 24×32 ASCII **数字字符显示函数**(FONT24_32.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint8 GUI_PutChar24_32(uint32 x, uint32 y,<br>uint8 ch) | x，y **字符显示的坐标**<br>ch **字符的十六进制值** | **显示一个字符(返回值为** 1 **时表示操作成功，为** 0 **表示失败**) |

**表** 11 **颜色转换操作函数**(CONVERTCOLOR.C)

| 函数原型 | 参数 | 功能 |
|---|---|---|
| uint16 GUI_Color2Index_565(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 64K **色索引值** |
| uint32 GUI_Index2Color_565(uint16 index) | index **索引颜色值** | 64K **色索引值** → RGB **颜色值** |
| uint16 GUI_Color2Index_555(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 32K **色索引值** |
| uint32 GUI_Index2Color_555(uint16 index) | index **索引颜色值** | 32K **色索引值** → RGB **颜色值** |
| uint16 GUI_Color2Index_444(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 4K **色索引值** |
| uint32 GUI_Index2Color_444(uint16 index) | index **索引颜色值** | 4K **色索引值** → RGB **颜色值** |
| uint8 GUI_Color2Index_332(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 256 **色索引值** |
| uint32 GUI_Index2Color_233(uint8 index) | index **索引颜色值** | 256 **色索引值** → RGB **颜色值** |
| uint8 GUI_Color2Index_222(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 64 **色索引值** |
| uint32 GUI_Index2Color_222(uint8 index) | index **索引颜色值** | 64 **色索引值** → RGB **颜色值** |
| uint8 GUI_Color2Index_111(uint32 colorRGB) | colorRGB RGB **颜色值** | RGB **颜色值** → 8 **色索引值** |
| uint32 GUI_Index2Color_111(uint8 Index) | index **索引颜色值** | 8 **色索引值** → RGB **颜色值** |