

## 正点原子 littleVGL 开发指南

lv\_label 标签控件

开发指南

正点原子  
广州市星翼电子科技有限公司

## 修订历史

版本	日期	原因
V1.00	2020/05/01	第一次发布

# lv\_label 标签控件

## 1. 介绍

lv\_label 标签控件可以说是 littleVGL 中使用最频繁的控件了,他的主要作用就是用来显示文本信息的,你可以在运行时的任何时候,使用 **lv\_label\_set\_text(label, "New text")**接口来动态修改文本内容,littleVGL 内部会重新为这个标签重新分配堆空间,当然了你也可以通过 **lv\_label\_set\_static\_text(label, char\_array)**这样的接口来引用一个外部的 char\_array 文本指针,这样的好处就是 littleVGL 内部不会为这个文本内容分配堆空间,从而可以减少内存的使用,此标签控件支持换行,图标字体,部分文本重绘色等显示功能,同时针对长文本显示,它支持 6 种显示模式.

## 2. lv\_label 的 API 接口

### 2.1 主要数据类型

#### 2.1.1 长文本模式数据类型

```
enum {  
    LV_LABEL_LONG_EXPAND, //自动扩展对象的大小来包裹文本内容  
  
    //保持对象的宽度不变,当文本内容的宽度超过对象的宽度时会  
    //自动换行,然后同时自动扩展对象的高度来包裹文本内容的高度  
    LV_LABEL_LONG_BREAK,  
  
    //保持对象的大小不变,当文本内容太长显示不下时,  
    //会在文本末尾显示...三个点的省略号  
    LV_LABEL_LONG_DOT,  
  
    //保持对象的大小不变,当文本内容太长显示不下时,会自动循环向前向后滚动文本  
    LV_LABEL_LONG_SCROLL,  
  
    //保持对象的大小不变,当文本内容太长显示不下时,会自动循环环形滚动文本  
    LV_LABEL_LONG_SCROLL_CIRC,  
  
    LV_LABEL_LONG_CROP, //保持对象大小不变,超过的文本内容将会被剪切掉  
};  
typedef uint8_t lv_label_long_mode_t;
```

#### 2.1.2 文本内容对齐数据类型

```
enum {  
    LV_LABEL_ALIGN_LEFT, //文本左对齐  
    LV_LABEL_ALIGN_CENTER, //文本居中对齐  
    LV_LABEL_ALIGN_RIGHT, //文本右对齐  
};  
typedef uint8_t lv_label_align_t;
```

#### 2.1.3 标签样式数据类型

```
enum {  
    LV_LABEL_STYLE_MAIN,  
};  
typedef uint8_t lv_label_style_t;
```

## 2.2 API 接口

### 2.2.1 创建标签

```
lv_obj_t * lv_label_create(lv_obj_t * par, const lv_obj_t * copy);
```

参数:

parent: 指向父对象

copy: 此参数可选,表示创建新对象时,把 copy 对象上的属性值复制过来

返回值:

返回新创建出来的标签对象,如果为 NULL 的话,说明堆空间不足了

littleVGL 中的控件都有自己专属的创建 API 接口,虽然名字不同,但是他们的接口命名规范和含义是相同的

### 2.2.2 设置动态文本(字符串形式)

```
void lv_label_set_text(lv_obj_t * label, const char * text);
```

参数:

label: 标签对象

text: 新的文本内容,文本内容要是'\0'空字符结尾,如果传 NULL 的话,那么代表刷新当前文本内容

这个 API 接口是用来设置文本内容的,但是我特意加了”动态”俩个字来修饰,这是因为当用此 API 接口来设置文本时,它会把之前文本所占用的内存空间先给释放掉,然后为这个新文本内容重新分配一个相应大小的内存空间,所以即使外面的 text 指针被释放了,也不会影响此标签控件的显示

### 2.2.3 设置动态文本(数组形式)

```
void lv_label_set_array_text(lv_obj_t * label, const char * array, uint16_t size);
```

参数:

label: 标签对象

array: 新的文本内容,不需要是'\0'空字符结尾,如果传 NULL 的话,那么代表刷新当前文本内容

size: 传入的 array 数组的大小,单位为字节

这个 API 接口和 2.2.2 中的 lv\_label\_set\_text 接口功能是一样的,只不过是传入文本的形式不一样,一个是以字符串形式,一个是以数组形式

### 2.2.4 设置静态文本(字符串形式)

```
void lv_label_set_static_text(lv_obj_t * label, const char * text);
```

#### 参数:

label: 标签对象

text: 新的文本内容,文本内容要是'\0'空字符结尾,如果传 NULL 的话,那么代表刷新当前文本内容

这是以“静态”的方式设置文本内容,所谓的静态就是标签对象内部不会为这个文本内容分配内存空间来保存它,而只是引用了一下这个文本指针,好处就是在某些场合下,可以节省内存,坏处就是外部的 text 内容空间不能随意的被释放,否则会引起标签对象的显示出错

### 2.2.5 设置静态文本(数组形式)

```
void lv_label_set_array_text(lv_obj_t * label, const char * array, uint16_t size);
```

#### 参数:

label: 标签对象

array: 新的文本内容,不需要是'\0'空字符结尾,如果传 NULL 的话,那么代表刷新当前文本内容

size: 传入的 array 数组的大小,单位为字节

这个 API 接口和 2.2.4 中的 lv\_label\_set\_static\_text 接口功能是一样的,只不过是传入文本的形式不一样,一个是以字符串形式,一个是以数组形式

### 2.2.6 设置长文本模式

```
void lv_label_set_long_mode(lv_obj_t * label, lv_label_long_mode_t long_mode);
```

#### 参数:

label: 标签对象

long\_mode: 长文本模式

此 API 接口很重要,而且也很奇妙,它跟调用时的位置有关(请看后面的例子 3),它的取值会影响到标签对象的大小,如果不设置的话,那么标签对象的长文本模式默认为 LV\_LABEL\_LONG\_EXPAND,下面让我们来看一下每一种长文本模式对标签对象大小的影响:

#### 1) LV\_LABEL\_LONG\_EXPAND

当设置为此模式时,用 lv\_obj\_set\_size 接口来设置大小是无效的,标签对象的宽和高只会随着文本的内容进行横向和纵向的扩展,默认是不会自动换行的,但是文本内容中可以插入'\n'字符来进行手动换行,请看下面 2 个例子(只给出关键示意代码)

例子 1:

```
lv_label_set_text(label1,"I am xiong jia yu");
```



图 2.2.6.1 例子 1 效果图

例子 2:

```
lv_label_set_text(label1,"I am xiong jia yu,Who are you?\nCan you tell me?");//使用 ' \n' 换行
```

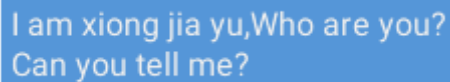


图 2.2.6.2 例子 2 效果图

## 2) LV\_LABEL\_LONG\_BREAK

当设置为此模式时,用 lv\_obj\_set\_size 接口来设置大小时,只有宽度是有效的,高度会随着文本的内容进行扩展,当文本的内容超过对象的指定宽度时,会进行自动换行的,请看下面的一个例子(只给出关键示意代码)

例子 3(正确的方式):

```
lv_label_set_long_mode(label1,LV_LABEL_LONG_BREAK);
lv_obj_set_size(label1,100,0);//宽度为 100 像素,高度是无效的,随意设置吧
lv_label_set_text(label1,"I amxiong jia yu,Who are you?Can you tell me?");
```

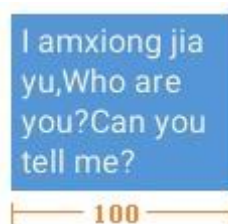


图 2.2.6.3 例子 3 效果图

前面我说过 lv\_label\_set\_long\_mode 接口的调用跟位置是有关系的,在这里 lv\_label\_set\_long\_mode 的调用必须放在 lv\_obj\_set\_size 调用的前面,否则设置的宽和高是无效的,即下面这种方式是错误的:

例子 3(错误的方式):

```
lv_obj_set_size(label1,100,0);//宽度为 100 像素,高度是无效的,随意设置吧
lv_label_set_long_mode(label1,LV_LABEL_LONG_BREAK);
lv_label_set_text(label1,"I amxiong jia yu,Who are you?Can you tell me?");
```

### 3) LV\_LABEL\_LONG\_DOT

当设置为此模式时,用 lv\_obj\_set\_size 接口来设置大小时,宽和高都是有效的,当文本的内容显示不下时,会在文本的末尾显示一个...省略号,同时具有自动换行的功能,请看下面的一个例子(只给出关键示意代码)

例子 4:

```
lv_label_set_long_mode(label1, LV_LABEL_LONG_DOT);
lv_obj_set_size(label1, 100, 50);
lv_label_set_text(label1, "I am xiong jia yu, Who are you? Can you tell me?");
```

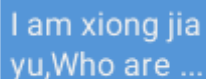


图 2.2.6.4 例子 4 效果图

注:默认情况下是显示...三个点的省略号,不过这个是可以通过 LV\_LABEL\_DOT\_NUM 宏来配置的,此宏在 lv\_label.h 头文件中,如下所示:

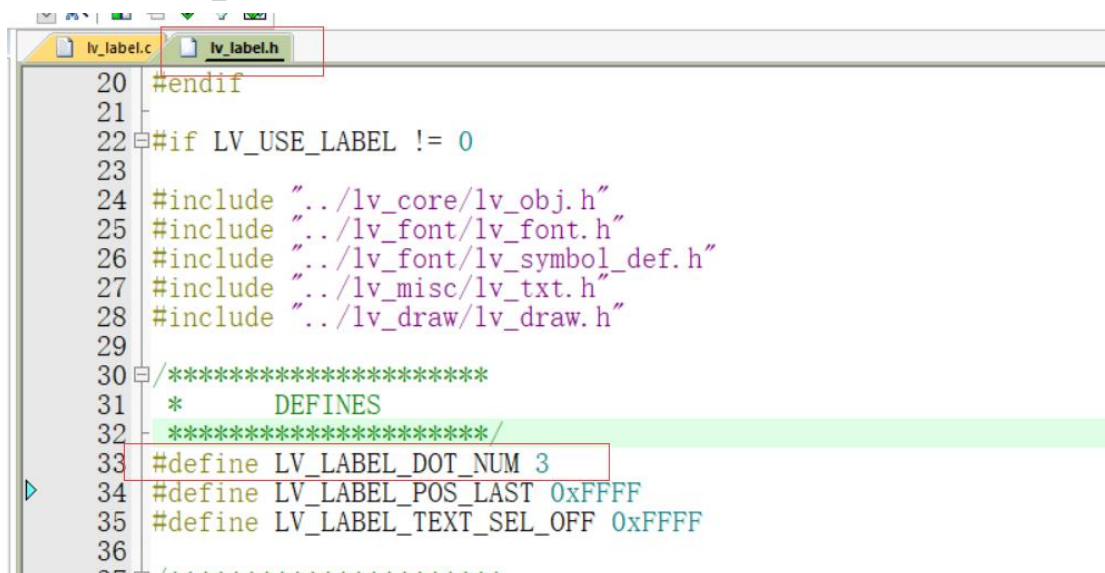


图 2.2.6.5 LV\_LABEL\_DOT\_NUM 宏

### 4) LV\_LABEL\_LONG\_SCROLL

当设置为此模式时,用 lv\_obj\_set\_size 接口来设置大小时,宽和高都是有效的,当文本的内容显示不下时,会进行向前向后的滚动显示,不具有自动换行的功能,请看下面的例子(只给出关键示意代码)

例子 5:

```
lv_label_set_long_mode(label1, LV_LABEL_LONG_SCROLL);
lv_obj_set_size(label1, 100, 50);
lv_label_set_text(label1, "I am xiong jia yu, Who are you? Can you tell me?");
```



图 2.2.6.5 例子 5 效果图

### 5) LV\_LABEL\_LONG\_SROLL\_CIRC

和 LV\_LABEL\_LONG\_SROLL 模式的功能基本是一样的,唯一区别就是 LV\_LABEL\_LONG\_SROLL\_CIRC 不是向前向后滚动,而是环形滚动,请看下面的例子(只给出关键示意代码)

例子 6:

```
lv_label_set_long_mode(label1, LV_LABEL_LONG_SROLL_CIRC);
lv_obj_set_size(label1, 100, 50);
lv_label_set_text(label1, "I am xiong jia yu, Hello");
```



图 2.2.6.6 例子 6 效果图

### 6) LV\_LABEL\_LONG\_CROP

当设置为此模式时,用 lv\_obj\_set\_size 接口来设置大小时,宽和高都是有效的,当文本的内容显示不下时,超出的部分会被直接剪切掉,不具有自动换行的功能,请看下面的例子(只给出关键示意代码)

例子 7:

```
lv_label_set_long_mode(label1, LV_LABEL_LONG_CROP);
lv_obj_set_size(label1, 100, 50);
lv_label_set_text(label1, "I am xiong jia yu, Hello");
```





图 2.2.6.7 例子 7 效果图

### 2.2.7 设置文本对齐方式

```
void lv_label_set_align(lv_obj_t * label, lv_label_align_t align);
```

参数:

label: 标签对象

align: 水平方向上的文本对齐方式

记住,要想让标签的文本内容具有对齐的效果,那么必须得先保证标签对象具有指定的宽度,请看下面例子

```
lv_obj_t *src = lv_scr_act();//获取屏幕对象
lv_obj_t *label1 = lv_label_create(src,NULL);//创建对象
lv_label_set_long_mode(label1,LV_LABEL_LONG_CROP);//设置长文本模式,不能设置为
//LV_LABEL_LONG_EXPAND 模式,因为其指定不了宽度
lv_obj_set_pos(label1,20,20);//设置坐标
lv_obj_set_size(label1,100,50);//设置宽和高
lv_label_set_text(label1,"Hello");//设置文本内容
lv_label_set_style(label1,LV_LABEL_STYLE_MAIN,&lv_style_plain_color);//设置具有背景色
lv_label_set_body_draw(label1,true);//设置绘制背景
lv_label_set_align(label1,LV_LABEL_ALIGN_CENTER);//设置文本居中对齐
```



图 2.2.7.1 居中对齐效果

### 2.2.8 是否使能文本重绘色功能

```
void lv_label_set_recolor(lv_obj_t * label, bool en);
```

参数:

label: 标签对象

en: 是否使能

使能之后,可以让标签的部分文本显示出不同的颜色,即一个标签里可以含有多种不同颜色的文本,这在其他的 GUI 库中,一般是办不到的,使用格式为: **#十六进制颜色值 文本#** ,注意了颜色值和文本之间至少得有一个空格,请看下面例子(只给出关键代码)

```
lv_label_set_recolor(label1,true);//先得使能文本重绘色功能
lv_label_set_text(label1,"#ff0000 red#,#00ff00 green#,#0000ff blue#");//使用了 3 次重绘色
```



图 2.2.8 文本重绘色效果

### 2.2.9 是否使能背景绘制功能

```
void lv_label_set_body_draw(lv_obj_t * label, bool en);
```

**参数:**

label: 标签对象

en: 是否使能

默认情况下,lv\_label 标签对象是没有背景的,即透明的,但是我们可以通过这个接口和样式使标签对象具有背景,请看如下例子

```
lv_label_set_style(label1,LV_LABEL_STYLE_MAIN,&lv_style_plain_color);//设置主背景的样
式为
//lv_style_plain_color
lv_label_set_body_draw(label1,true);//使能背景绘制
lv_label_set_text(label1,"Hello world");
```



图 2.2.9.1 带有背景的效果

### 2.2.10 设置动画时的速度

```
void lv_label_set_anim_speed(lv_obj_t * label, uint16_t anim_speed);
```

**参数:**

label: 标签对象

anim\_speed: 动画速度,单位为 px/sec(一秒多少个像素)

这个接口主要用来设置文本滚动(LV\_LABEL\_LONG\_SROLL/SCROLL\_CIRC 模式下)时的速度

### 2.2.11 设置样式

```
static inline void lv_label_set_style(lv_obj_t * label, lv_label_style_t type, const lv_style_t * style)
```

#### 参数:

label: 标签对象

type: 哪一个部件的样式,目前就 LV\_LABEL\_STYLE\_MAIN 主背景部件这一个可选值

style: 样式

### 2.2.12 插入文本

```
void lv_label_ins_text(lv_obj_t * label, uint32_t pos, const char * txt);
```

#### 参数:

label: 标签对象

pos: 插入的位置,0 代表从最前面插入, LV\_LABEL\_POS\_LAST 代表从最后面插入

txt: 要插入的文本

这个接口有一点需要注意,那就是必须得保证之前的文本是动态设置,而不能是静态文本,因为我们知道插入动作是修改内存的操作,而如果之前的文本是通过静态方式设置的话,那么此内存有可能是常量区(用 const 关键字修饰),常量区内存是不能修改的

### 2.2.13 剪切文本

```
void lv_label_cut_text(lv_obj_t * label, uint32_t pos, uint32_t cnt);
```

#### 参数:

label: 标签对象

pos: 剪切的起始位置,从 0 开始

cnt: 要剪切的字符数量

剪切文本也可以理解成删除指定区域的文本,这个接口也同样有一点需要注意,那就是必须得保证之前的文本是动态设置,而不能是静态文本

### 2.2.14 备注

还有一些比较简单的 API 接口,我这里就不列出来了,通过看函数名就能知道大概啥意思了,这些接口基本都是 get 获取函数,用来获取对象的属性的

## 3. 例程设计

### 3.1 功能简介

创建 label1 和 label2 俩个标签,label1 标签主要是用来做标题的,显示 label2 标签的当前长文本模式,而 label2 标签主要是用来演示 6 大长文本模式的,为 label2 标签注册了事件回调函数,通过点击 label2 标签,来循环切换 label2 标签的长文本模式,如果按下 KEY0 键的话,则是加大 label2 标签的动画速度,可以在 LV\_LABEL\_LONG\_SCROLL 和 LV\_LABEL\_LONG\_SCROLL\_CIRC 模式下观察到明显的动画快慢效果,如果按下 KEY1 键,则是在 label1 文本的最前面插入 OK 字符串,如果按下 KEY2 键,则是删除 label1 文本最前面的 2 个字符

### 3.2 硬件设计

本例程所用到的硬件有:

- 1) 串口 1
- 2) KEY0,KEY1,KEY2 按键
- 3) 液晶屏

### 3.3 软件设计

在 GUI\_APP 目录下新建 lv\_label\_test.c 和 lv\_label\_test.h 俩个文件, lv\_label\_test.c 文件的内容如下:

```
#include "lv_label_test.h"
#include "lvgl.h"
#include "delay.h"
#include "usart.h"
#include "key.h"

lv_obj_t* label1;
lv_obj_t* label2;
//模式标题
const char * const MODE_STR[] = {"EXPAND","BREAK","DOT","SCROLL","SCROLL_CIRC","CR
OP"};

//事件处理
void event_handler(lv_obj_t * obj, lv_event_t event)
{
```

```

static lv_label_long_mode_t mode = LV_LABEL_LONG_EXPAND;

if(obj==label2)
{
    if(event==LV_EVENT_RELEASED)//触摸释放事件
    {
        lv_label_set_long_mode(label2,mode);//设置新的长文本模式
        if(mode==LV_LABEL_LONG_EXPAND)//自动扩展模式
        {
            lv_label_set_text(label2,"EXPAND:0123456789ABCDEFGHIJKLMN\n
I am xiong jia yu\nWho are you?");
        }else if(mode==LV_LABEL_LONG_BREAK)//自动换行模式
        {
            lv_obj_set_width(label2,100);
            lv_label_set_text(label2,"BREAK:Auto to break line");
        }else if(mode==LV_LABEL_LONG_DOT)//自动显示省略号模式
        {
            lv_obj_set_size(label2,100,40);
            lv_label_set_text(label2,"DOT:too long,0123456789ABCDEFGHIJKL
MN");
        }else if(mode==LV_LABEL_LONG_SROLL)//自动前后滚动模式
        {
            lv_obj_set_size(label2,100,40);
            lv_label_set_text(label2,"SROLL:KEY0 to add speed");
        }else if(mode==LV_LABEL_LONG_SROLL_CIRC)//自动环形滚动模式
        {
            lv_obj_set_size(label2,100,40);
            lv_label_set_text(label2,"SROLL_CIRC:KEY0 to add speed");
        }else if(mode==LV_LABEL_LONG_CROP)//剪切模式
        {
            lv_obj_set_size(label2,100,40);
            lv_label_set_text(label2,"CROP:0123456789ABCDEF");
        }
        lv_obj_realign(label2);//因为 label2 的大小发生了改变,为了让 label2
                                //继续与屏幕保持居中对齐,可以调用重对齐接口
        lv_label_set_text(label1,MODE_STR[mode]);
        printf("current long mode:%d\r\n",mode);
        //切换到下一个模式
        mode++;
        if(mode>LV_LABEL_LONG_CROP)
            mode = LV_LABEL_LONG_EXPAND;
    }
}
}

```

```
//例程入口函数
void lv_label_test_start()
{
    lv_obj_t* scr = lv_scr_act();//获取当前活跃的屏幕对象

    //创建 label1 标签,用来显示 label2 标签的长文本模式
    label1 = lv_label_create(scr,NULL);//创建标签
    lv_label_set_long_mode(label1,LV_LABEL_LONG_BREAK);//设置长文本模式
    //设置宽度,一定得放在 lv_label_set_long_mode 的后面,否则不起作用的
    lv_obj_set_width(label1,160);
    lv_label_set_recolor(label1,true);//使能文本重绘色功能
    lv_label_set_text(label1,"#ff0000 Title:#mode");//设置文本,带有颜色重绘
    lv_label_set_align(label1,LV_LABEL_ALIGN_CENTER);//文本居中对齐
    //设置主背景样式
    lv_label_set_style(label1,LV_LABEL_STYLE_MAIN,&lv_style_plain_color);
    lv_label_set_body_draw(label1,true);//使能背景绘制
    //注意:设置与屏幕的对齐方式,这个接口也是跟调用位置有关系的,最好放到后面调
    //用,因为放的太前调用的话,
    //在那时标签对象的大小可能还是未知的,此时 lv_obj_align 接口就没办法算出对齐
    //之后的坐标,从而也就
    //达不到我们所要的对齐效果
    lv_obj_align(label1,NULL,LV_ALIGN_IN_TOP_MID,0,20);

    //创建 label2 标签
    label2 = lv_label_create(scr,NULL);//创建标签
    //设置主背景样式
    lv_label_set_style(label2,LV_LABEL_STYLE_MAIN,&lv_style_plain_color);
    lv_label_set_body_draw(label2,true);//使能背景绘制
    lv_obj_set_click(label2,true);//使能点击功能
    lv_obj_set_event_cb(label2,event_handler);//设置事件回调函数
    lv_label_set_text(label2,"Please click me!");//设置文本
    lv_obj_align(label2,NULL,LV_ALIGN_CENTER,0,0);//设置其与屏幕居中对齐
}

//按键处理
//注意:请按照先按 KEY0 键,再按 KEY1 键,最后按 KEY2 键的顺序来观察实验现象
void key_handler()
{
    u8 key = KEY_Scan(0);

    if(key==KEY0_PRES)
    {
```

```
//调节动画速度
lv_label_set_anim_speed(label2,lv_label_get_anim_speed(label2)+5);
printf("anim speed:%d\r\n",lv_label_get_anim_speed(label2));
}else if(key==KEY1_PRES)
{
    //在 label1 文本的最前面插入 OK 字符串
    lv_label_ins_text(label1,0,"OK");
}else if(key==KEY2_PRES)
{
    //删除 label1 文本最前面的 2 个字符
    lv_label_cut_text(label1,0,2);
}
}
```

### 3.4 下载验证

请根据功能简介,一一验证实验效果,下面给出一张代码刚下载完后的初始界面效果

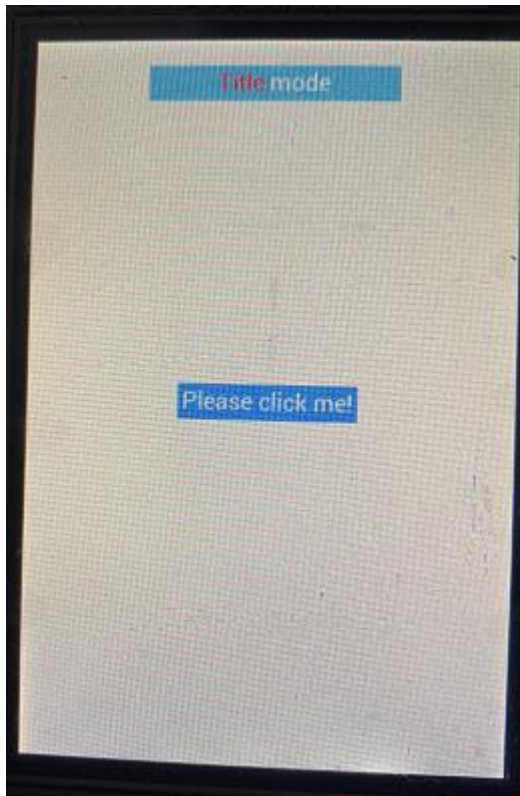


图 3.4.1 初始界面效果

**注:**如果是电阻屏的话,在开机之前可以先按住 KEY0 键不放,接着在开机(或按复位键也行),可以先进入到电阻屏触摸校准程序,校准完成后会自动进入到演示例程,此注意事项对以后的所有例程同样适用



## 4. 资料下载

正点原子公司名称：广州市星翼电子科技有限公司

LittleVGL 资料连接：[www.openedv.com/thread-309664-1-1.html](http://www.openedv.com/thread-309664-1-1.html)

原子哥在线教学平台：[www.yuanzige.com](http://www.yuanzige.com)

正点原子淘宝店铺：<https://openedv.taobao.com>

正点原子官方网站：[www.alientek.com](http://www.alientek.com)

正点原子 B 站视频：<https://space.bilibili.com/394620890>

电话：020-38271790 传真：020-36773971

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

请关注正点原子公众号，资料发布更新我们会通知。



扫码下载“原子哥”APP



扫码关注正点原子公众号