User Guide

正点原子 littleVGL 开发指南

lv_ta 文本域

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_ta 文本域

1. 介绍

lv_ta 文本域控件其实就是我们通常所看到的文本输入框,配合我们后面将要讲到的 lv_kb 键盘控件可以实现输入某些内容,lv_ta 文本域控件是由一个 lv_page 页面控件,一个 lv_label 标签控件,以及还有一个光标构成的,利用 lv_page 页面控件的特性,lv_ta 文本域控件可以实现垂直和水平滚动,利用 lv_label 标签控件的特性,lv_ta 文本域控件可以实现显示文本的功能,而光标的作用就是用于标记当前所在的文本输入位置.

lv_ta 文本域控件具有 Placeholder 提示文字的功能,可以通过 lv_ta_set_placeholder_text(ta, "Placeholder text")接口来设置想要的提示文字,当 lv_ta 文本域控件中没有输入任何文本内容时,lv_ta 文本域控件就会把 Placeholder 提示文字给显示出来,可以起到提醒用户输入,或者用来说明此文本域用途等功能,当文本域中一旦含有文本内容时,不管是用户输入的,还是自己通过 API 接口添加的,此 Placeholder 提示文字都会立即消失.

文本域中是可以显示文本内容的,那我们怎么来给其添加文本内容呢?一是通过 API 接口来直接添加,二是通过 lv_kb 键盘来间接输入,当然了 lv_kb 键盘输入的内部实现原理也是利用到了 API 接口,在文本域控件中,一共有如下三个 API 接口可以实现添加文本:

1) lv_ta_set_text(ta,"xiong jia yu") 这是直接设置文本内容
2) lv_ta_add_char(ta, 'c') 这是添加单个字符
3) lv_ta_add_text(ta, "hello world") 这是添加字符串

添加的文本内容会放到或插入到当前光标所在的位置,而光标的位置是可以通过用户点击和 API 接口来改变的,如果是想通过用户点击来改变光标的位置,那么是有一个前提的,那就是必须得调用 lv_ta_set_cursor_click_pos(ta, true)接口来先使能此功能,不过文本域控件默认是使能了此功能的,如果是想通过 API 接口来改变光标的位置,总共有如下 5 种方式:

1) lv_ta_set_cursor_pos(ta, pos) 设置任意的光标位置,pos 从 0 开始
2) lv_ta_cursor_right(ta) 设置光标向右移动一步
3) lv_ta_cursor_left(ta) 设置光标向左移动一步
4) lv_ta_cursor_up(ta) 设置光标向上移动一步,文本域中有多行文本时才起作用
5) lv_ta_cursor_down(ta) 设置光标向下移动一步,文本域中有多行文本时才起作用

文本域控件有添加文本内容的方式,当然也会有删除文本内容的方式,删除的 API 接口有如下俩个:

1)lv_ta_del_char(ta) 删除当前光标左侧的一个字符
2)lv_ta_del_char_forward(ta) 删除当前光标右侧的一个字符

在文本域控件中,光标的类型总共有 5 种,分别为无, 垂直线, 填充矩形块, 矩形边框, 下划线,它是通过 lv_ta_set_cursor_type(ta, LV_CURSOR_...)接口来设置的.

文本域控件默认情况下是多行模式的,如果你想使能文本域控件的单行模式,可以通过 lv_ta_set_one_line(ta, true)接口来设置,变成单行模式之后,\n 换行符和 word wrap 自动换行功能将会被忽略,和 lv_label 标签一样,文本域控件也具有文本对齐的功能,可以通过 lv_ta_set_text_align(ta, LV_LABEL_ALIGN_LET/CENTER/RIGHT)接口来设置,如果文本域控件是处于单行模式,而且文本是左对齐的,那么当文本内容过长超过文本域的宽度时,文本内容是可以被水平滚动条滑动的.

littleVGL 中的 lv_ta 文本域控件是非常强大的,在小细节处理上也表现得非常出色,比如

有时候我们可能会有这样的一些需求场景,我们希望文本域只能显示数字或者某些英文字母,同时还想限定其最大长度不超过指定值,希望密码输入框有保护显示功能等等,而 lv_ta 文本域控件可以统统满足你,通过 lv_ta_set_accepted_chars(ta, list)接口,你可以设置文本域所能接受的字符列表,比如 const char * list = “0123456789”;那么此时文本域就只能显示数字了,不在此列表中的字符将统统不能被显示出来,通过 lv_ta_set_pwd_mode(ta, true)接口,你可以使能文本域的密码输入保护功能,使能之后,它将会用*号来代替每个字符进行显示,通过 lv_ta_set_max_length(ta, max_char_num)接口,你可以指定文本域所能显示的最大字符数,然后这里有一点需要注意的是,当文本域显示的字符数超过 20k 时,文本域控件的渲染速度可能会变得很慢,那么你可以通过将 lv_conf.h 中的 LV_LABEL_LONG_TXT_HINT 宏设为 1 来加速渲染过程,这个操作会额外增加 12 字节的内存消耗.

最后我们来说一下文本域控件的事件,除了按下,松手等常用事件外,还有 LV_EVENT_INSERT 和 LV_EVENT_VALUE_CHANGED 俩个特殊事件与之相关,当有新的文本内容放入到或者说是插入到文本域中时,LV_EVENT_INSERT 事件将会被触发,这个新的文本内容将会被作为事件自定义参数给传递过去,同时请确保此新文本内容在事件回调函数被调用之前不能被释放,当文本域中的文本内容发生改变时,LV_EVENT_VALUE_CHANGED 事件将会被触发.

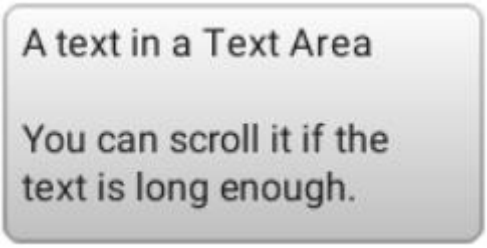

图 1.1 lv_ta 文本域控件的默认外观

2. lv_ta 的 API 接口

2.1 主要数据类型

2.1.1 光标样式数据类型

```
enum {
    LV_CURSOR_NONE,
    LV_CURSOR_LINE,
    LV_CURSOR_BLOCK,
    LV_CURSOR_OUTLINE,
    LV_CURSOR_UNDERLINE,
    LV_CURSOR_HIDDEN = 0x08,
};
typedef uint8_t lv_cursor_type_t;
```

LV_CURSOR_NONE：无光标显示

LV_CURSOR_LINE:垂直线光标,如 

所示

LV_CURSOR_BLOCK：填充矩形块光标,如 

所示

LV_CURSOR_OUTLINE：矩形边框光标,如 

所示

LV_CURSOR_UNDERLINE:下划线光标,如 

所示

LV_CURSOR_HIDDEN：隐藏光标,和 LV_CURSOR_NONE 的作用差不多,不过

LV_CURSOR_HIDDEN：它可以和上面的其他值进行位或操作

2.1.2 文本域样式数据类型

```
enum {
    LV_TA_STYLE_BG,
    LV_TA_STYLE_SB,
    LV_TA_STYLE_CURSOR,
    LV_TA_STYLE_EDGE_FLASH,
    LV_TA_STYLE_PLACEHOLDER,
};
typedef uint8_t lv_ta_style_t;
```

LV_TA_STYLE_BG: 用此样式中的body字段来修饰文本域的背景,用此样式中的text字段来修饰其文本内容,默认值为 lv_style_pretty.

LV_TA_STYLE_SB: 用来修饰水平和垂直滚动条的,使用样式中的 body 字段,默认值为 lv_style_pretty_color,其中 body.padding.right 是用来控制垂直滚动条与页面右边缘的距离,当此值为正数时,是在页面内部,当为负值时,是在页面外部,而 body.padding.bottom 是用来控制水平滚动条与页面底边缘的距离,当此值为正数时,是在页面内部,当为负值时,是在页面外部,而 body.padding.inner 是用来设置滚动条的宽度.

LV_TA_STYLE_CURSOR: 用来修饰光标的,如果不设置的话,则系统会根据文本的字体和颜色来自动设置光标的样式,我们一般不设置.

LV_TA_STYLE_EDGE_FLASH: 用来修饰边缘半圆弧动画效果的,一般只用到里面的body.main_color, body.grad_color, body.opa 等样式字段,对于上边缘只用 grad_color 来设置半圆弧的颜色,对于下边缘只用 main_color 来设置半圆弧的颜色,而对于左和右边缘,main_color 和 grad_color 都会用到.

LV_TA_STYLE_PLACEHOLDER: 用来修饰 Placeholder 提示文字的,使用此样式中的 text 字段.

2.2 API 接口

2.2.1 创建对象

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

参数:
par: 父对象
copy: 拷贝的对象,如果无拷贝的话,传 NULL 值

返回值:
返回创建出来的对象,如果返回 NULL 的话,说明堆空间不够了

2.2.2 添加字符内容

```
void lv_ta_add_char(lv_obj_t * ta, uint32_t c);
```

参数:
ta: 文本域对象
c: 字符

在当前光标所在的位置处添加一个字符内容

2.2.3 添加字符串内容

```
void lv_ta_add_text(lv_obj_t * ta, const char * txt);
```

参数:
ta: 文本域对象
txt: 字符串文本内容

在当前光标所在的位置处添加字符串文本内容

2.2.4 删除光标左侧的一个字符

```
void lv_ta_del_char(lv_obj_t * ta);
```

参数:
ta: 文本域对象

2.2.5 删除光标右侧的一个字符

```
void lv_ta_del_char_forward(lv_obj_t * ta);
```

参数:
ta: 文本域对象

2.2.6 设置文本内容

```
void lv_ta_set_text(lv_obj_t * ta, const char * txt);
```

参数:
ta: 文本域对象
txt: 文本内容

这是直接设置文本域控件的文本内容

2.2.7 设置 placeholder 提示文字

```
void lv_ta_set_placeholder_text(lv_obj_t * ta, const char * txt);
```

参数:
ta: 文本域对象
txt: 提示文字

当 lv_ta 文本域控件中没有输入任何文本内容时,lv_ta 文本域控件就会把 placeholder 提示文字给显示出来,可以起到提醒用户输入,或者用来说明此文本域用途等功能,当文本域中一旦含有文本内容时,不管是用户输入的,还是自己通过 API 接口添加的,此 placeholder 提示文字都会立即消失

2.2.8 设置光标的任意位置

```
void lv_ta_set_cursor_pos(lv_obj_t * ta, int16_t pos);
```

参数:
ta: 文本域对象
pos: 光标的位置,从 0 开始,如果为负数的话,则代表以文本末尾为起点开始算,当为 LV_TA_CURSOR_LAST 值时,则是直接将光标定位到末尾

2.2.9 设置光标的类型

```
void lv_ta_set_cursor_type(lv_obj_t * ta, lv_cursor_type_t cur_type);
```

参数:
ta: 文本域对象
cur_type: 光标的类型,有如下 6 个可选值
LV_CURSOR_NONE: 不显示光标
LV_CURSOR_LINE: 垂直线光标
LV_CURSOR_BLOCK: 填充矩形块光标
LV_CURSOR_OUTLINE: 矩形边框光标
LV_CURSOR_UNDERLINE: 下划线光标
LV_CURSOR_HIDDEN: 隐藏光标,和 LV_CURSOR_NONE 的作用差不多

2.2.10 是否使能用户点击改变光标位置

```
void lv_ta_set_cursor_click_pos(lv_obj_t * ta, bool en);
```

参数:
ta: 文本域对象
en: 是否使能

默认是使能的,当使能之后,用户可以点击文本域控件的某处,光标也会相应地跳到手点击的地方

2.2.11 是否使能密码保护模式

```
void lv_ta_set_pwd_mode(lv_obj_t * ta, bool en);
```

参数:
ta: 文本域对象
en: 是否使能

当使能密码保护之后,用户输入的每一个字符在短暂时间显示之后,会被转换成*号来进行最终显示,如下图所示:

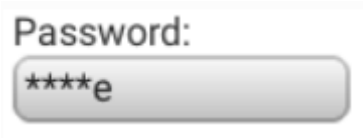

图 2.2.11.1 密码保护效果

注:上面说到的短暂时间是可以通过 lv_ta_set_pwd_show_time 接口来设置的

2.2.12 是否使能单行模式

```
void lv_ta_set_one_line(lv_obj_t * ta, bool en);
```

参数:
ta: 文本域对象
en: 是否使能

使能单行模式之后,\n 换行符和自动换行(word wrap)功能将会被忽略

2.2.13 设置文本对齐方式

```
void lv_ta_set_text_align(lv_obj_t * ta, lv_label_align_t align);
```

参数:
ta: 文本域对象
align: 文本对齐方式

2.2.14 设置文本域可接受的字符列表

```
void lv_ta_set_accepted_chars(lv_obj_t * ta, const char * list);
```

参数:
ta: 文本域对象
list: 字符列表,比如 const char * list = “0123456789”;那么此时文本域就只能显示数字了,不在此列表中的字符将统统不能被显示出来

2.2.15 设置文本域能显示的最大字符数

```
void lv_ta_set_max_length(lv_obj_t * ta, uint16_t num);
```

参数:
ta: 文本域对象
num: 能显示的最大字符数

这里有一点需要注意的是,当文本域实际显示的字符数超过 20k 时,文本域控件的渲染速

度可能会变得很慢,那么你可以通过将 lv_conf.h 中的 LV_LABEL_LONG_TXT_HINT 宏设为 1 来加速渲染过程,这个操作会额外增加 12 字节的内存消耗

2.2.16 用新文本内容替换掉正打算插入的文本内容

```
void lv_ta_set_insert_replace(lv_obj_t * ta, const char * txt);
```

参数:
ta: 文本域对象
txt: 新文本内容

此接口必须得在 LV_EVENT_INSERT 事件回调中调用才能看到效果,主要是用来对用户输入的信息进行过滤或者格式控制的,可能这样理解起来比较费劲,请看如下例子(只给出示意代码):

```
lv_obj_t * ta1;
//事件回调函数
void event_handler(lv_obj_t *obj,lv_event_t event)
{
if(event==LV_EVENT_INSERT) //当有文本插入时,就会触发此事件
{
const char * inserted_txt = lv_event_get_data();//获取到被插入的文本

//如果用户插入的文本是 date 字符串的话,那么我们对其进行过滤或者格式控制
//我这里就把它替换成一个固定的日期时间,当然了,你也可以做其他的操作
if(strcmp(inserted_txt,"date")==0)
      {
  //用"2019/12/26 23:45"字符串替换掉正准备插入的”date”字符串
        lv_ta_set_insert_replace(ta1,"2019/12/26 23:45");
}
}
}

ta1 = lv_ta_create(lv_scr_act(), NULL); //创建文本域
lv_ta_set_text(ta1, "now:");//直接设置文本内容

//模拟用户插入 date 字符串的操作,会触发 LV_EVENT_INSERT 事件
lv_ta_add_text(ta1,"date");
```

最后我们可以看到如下图所示的效果:

图 2.2.16.1 效果图

从上图的效果中我们可以看到,插入的 date 字符串并不会被显示出来,这是因为它已经被 lv_ta_set_insert_replace 所指定的新文本内容给替换掉了

2.2.17 设置滚动条的模式

```
lv_ta_set_sb_mode(lv_obj_t * ta, lv_sb_mode_t mode);
```

参数:
ta: 文本域对象
mode: 滚动条的模式

此 API 接口的使用方法和 lv_page 页面控件中的 lv_page_set_sb_mode 接口的使用方法是一模一样的,详情请看”lv_page 页面”章节

2.2.18 是否使能边缘半圆弧动画效果

```
static inline void lv_ta_set_edge_flash(lv_obj_t * ta, bool en);
```

参数:
ta: 文本域对象
en: 是否使能

此 API 接口的使用方法和 lv_page 页面控件中的 lv_page_set_edge_flash 接口的使用方法是一模一样的,详情请看”lv_page 页面”章节

2.2.19 设置样式

```
void lv_ta_set_style(lv_obj_t * ta, lv_ta_style_t type, const lv_style_t * style);
```

参数:
ta: 文本域对象
type: 设置那一部分的样式,目前有如下 5 个可选值
LV_TA_STYLE_BG: 修饰文本域的背景和文本内容
LV_TA_STYLE_SB: 修饰水平和垂直滚动条的
LV_TA_STYLE_CURSOR: 修饰光标的,我们一般不设置,系统会自动设置的
LV_TA_STYLE_EDGE_FLASH: 修饰边缘半圆弧动画效果的
LV_TA_STYLE_PLACEHOLDER: 修饰 Placeholder 提示文字的
style: 样式

2.2.20 是否使能文本选中功能

```
void lv_ta_set_text_sel(lv_obj_t * ta, bool en);
```

参数:
ta: 文本域对象
en: 是否使能

使用此 API 接口的前提是先将 lv_conf.h 头文件中的 LV_LABEL_TEXT_SEL 宏给置 1,然后如果将 en 设为 true 使能之后,我们就可以通过手指触摸选中某个区域的文本了,效果如下

图所示:

图 2.2.20.1 文本选中效果

扩展:可以采用如下方法来获取被选中的文本内容

```
lv_obj_t * ta_label = lv_obj_get_ext_attr(ta1)->label;//获取 ta1 文本域中的标签对象
uint16_t   start_pos = lv_label_get_text_sel_start(ta_label);//获取被选中文本的起始位置
uint16_t   end_pos = lv_label_get_text_sel_end(ta_label);//获取被选中文本的终止位置
拿到了 start_pos 和 end_pos,也就间接地拿到了被选中的文本内容
```

2.2.21 判断文本是否有被选中

```
bool lv_ta_text_is_selected(const lv_obj_t * ta);
```

参数:
ta: 文本域对象

返回值:
如果有文本被选中了,返回 false,没有被选中的话,返回 true

使用此 API 接口的前提是先将 lv_conf.h 头文件中的 LV_LABEL_TEXT_SEL 宏给置 1

2.2.22 清除文本选中状态

```
void lv_ta_clear_selection(lv_obj_t * ta);
```

参数:
ta: 文本域对象

使用此 API 接口的前提是先将 lv_conf.h 头文件中的 LV_LABEL_TEXT_SEL 宏给置 1

2.2.23 将光标移动一步

```
void lv_ta_cursor_right(lv_obj_t * ta); //光标向右移动一步
void lv_ta_cursor_left(lv_obj_t * ta);//光标向左移动一步
void lv_ta_cursor_down(lv_obj_t * ta);//光标向下移动一步,文本域中有多行文本时才起作用
void lv_ta_cursor_up(lv_obj_t * ta);//光标向上移动一步,文本域中有多行文本时才起作用
```

参数:
ta: 文本域对象

2.2.24 设置密码保护时的短暂显示时间

```
void lv_ta_set_pwd_show_time(lv_obj_t * ta, uint16_t time);
```

参数:
ta: 文本域对象
time: 单位 ms

在密码保护显示模式下,用户输入的每个字符经过 time 短暂时间显示之后,会被系统转化为*号进行最终的显示,如果 time 为 0 的话,则代表立即被转化成*号

2.2.25 设置光标的闪烁间隔

```
void lv_ta_set_cursor_blink_time(lv_obj_t * ta, uint16_t time);
```

参数:
ta: 文本域对象
time：闪烁间隔,单位为 ms

2.2.26 备注

还有几个 get 获取类型的 API 接口我这里就不列举出来了,比较简单的

3.例程设计

3.1 功能简介

创建一个文本域对象,设置其大小,与屏幕居中对齐,设置其光标类型,文本内容,注册事件回调函数等等,当按下 KEY0 按键时,往光标所在的位置处添加[add txt]文本,当按下 KEY1 按键时,删除光标左侧的一个字符,当按下 WKUP 按键时,往光标所在的位置处添加 d 字符,这个主要是配合 LV_EVENT_INSERT 事件,用来演示插入替换功能的

3.2 硬件设计

本例程所用到的硬件有:

1) 液晶屏
2) KEY0, KEY1, WKUP 按键
3) 串口

3.3 软件设计

在 GUI_APP 目录下创建 lv_ta_test.c 和 lv_ta_test.h 俩个文件,其中 lv_ta_test.c 文件的内容如下:

```
#include "lv_ta_test.h"
#include "lvgl.h"
#include "key.h"
#include <stdio.h>
#include <string.h>

lv_obj_t * ta1;

//事件回调函数
void event_handler(lv_obj_t *obj,lv_event_t event)
{
    if(event==LV_EVENT_VALUE_CHANGED)//文本内容改变时,就会触发此事件
    {
        //把当前的文本打印出来
        printf("LV_EVENT_VALUE_CHANGED:%s\r\n",lv_ta_get_text(ta1));
    }else if(event==LV_EVENT_INSERT)//有文本插入时,就会触发此事件
    {
        const char * inserted_txt = lv_event_get_data();//获取被插入的文本
```

```
        printf("LV_EVENT_INSERT:%s\r\n",inserted_txt);//把插入的文本打印出来
        //如果插入的是 d 字符串的话,就进行过滤或者格式控制
        if(strcmp(inserted_txt,"d")==0)
        {
          //用"2019/12/26"替换掉正准备插入的"d"
          lv_ta_set_insert_replace(ta1,"2019/12/26");
        }
    }
}

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

    //1.创建文本域对象
    ta1 = lv_ta_create(scr,NULL);
    lv_obj_set_size(ta1,200,100);//设置文本域的大小
    lv_obj_align(ta1,NULL,LV_ALIGN_CENTER,0,0);//与屏幕居中对齐
    lv_ta_set_cursor_type(ta1,LV_CURSOR_LINE);//设置光标的类型为竖直线

    //设置提示文字,当文本内容为空时,才会显示出来
    lv_ta_set_placeholder_text(ta1,"Please input");
    lv_ta_set_text(ta1,"This is a text area!");//设置文本内容
    lv_ta_set_pwd_mode(ta1,false);//不使能密码保护模式,默认就是不使能的
    lv_ta_set_one_line(ta1,false);//不使能单行模式,默认就是不使能的
    lv_ta_set_text_sel(ta1,true);//使能文本选中功能
    lv_ta_set_cursor_click_pos(ta1,true);//使能用户点击改变光标位置,默认就是使能的

    //设置文本居左对齐,默认就是居左对齐
    lv_ta_set_text_align(ta1,LV_LABEL_ALIGN_LEFT);
    lv_obj_set_event_cb(ta1,event_handler);//设置事件回调函数

}


//按键处理
void key_handler()
{
    u8 key = KEY_Scan(0);

    if(key==KEY0_PRES)
    {
        //往光标所在的位置处添加文本
```

```
            lv_ta_add_text(ta1,"[add txt]");
        }else if(key==KEY1_PRES)
        {
            //往光标所在的位置处添加字符,用来演示插入替换功能
            lv_ta_add_char(ta1,'d');
        }else if(key==WKUP_PRES)
        {
            //删除光标左侧的一个字符
            lv_ta_del_char(ta1);
        }
    }
```

3.4 下载验证

把代码下载进去之后,可以看到如下所示的初始界面效果:

图 3.4.1 初始界面效果

然后我们可以用手触摸来选中文本,如下图所示:

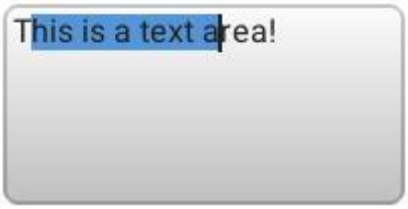

图 3.4.2 文本选中效果

接着我们可以按一下 KEY0 按键,往光标所在的位置处添加[add txt]文本,如下图所示:

图 3.4.3 添加或者插入文本效果

再接着我们可以多按几下 KEY1 按键,删除掉光标左侧的几个字符,如下图所示:

图 3.4.4 删除字符效果

最后我们可以按一下 WKUP 按键,来演示插入替换功能,插入的 d 字符会被替换成 2019/12/26 这个字符串的

图 3.4.5 插入替换功能效果

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

原子哥在线教学平台：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

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