LVGL 移植高通字库

字库芯片: GT5SLAD3BFA MCU: STM32F429 LVGL 版本：V8.4

一、实现 gt_read_data()

gt_read_data()函数的作用：与字库 flash 进行通信，函数的定义里调用 spi 发送数据和接收数据的接口。用户只需要实现该函数，就可以调用 GT_Font_Init()进行字库初始化。

请参考下面视频 如何在 32 位 MCU 上使用高通点阵字库：

https://www.bilibili.com/video/BV1aG41117uH/?spm_id_from=333.999.0.0

高通字库使用教程(1)硬件链接与注意事项部分：

https://www.bilibili.com/video/BV1PxcyeZEWW/?vd_source=c084c395e4fcfac58df3ea21ada16b68

高通字库使用教程(2)SPI 底层函数使用：

https://www.bilibili.com/video/BV1CxcyeoEpJ/?vd_source=c084c395e4fcfac58df3ea21ada16b68

高通字库使用教程(3)SPI 底层函数验证：

https://www.bilibili.com/video/BV1CxcyeoEXM/?vd_source=c084c395e4fcfac58df3ea21ada16b68

高通字库使用教程(4)关于库函数的讲解：

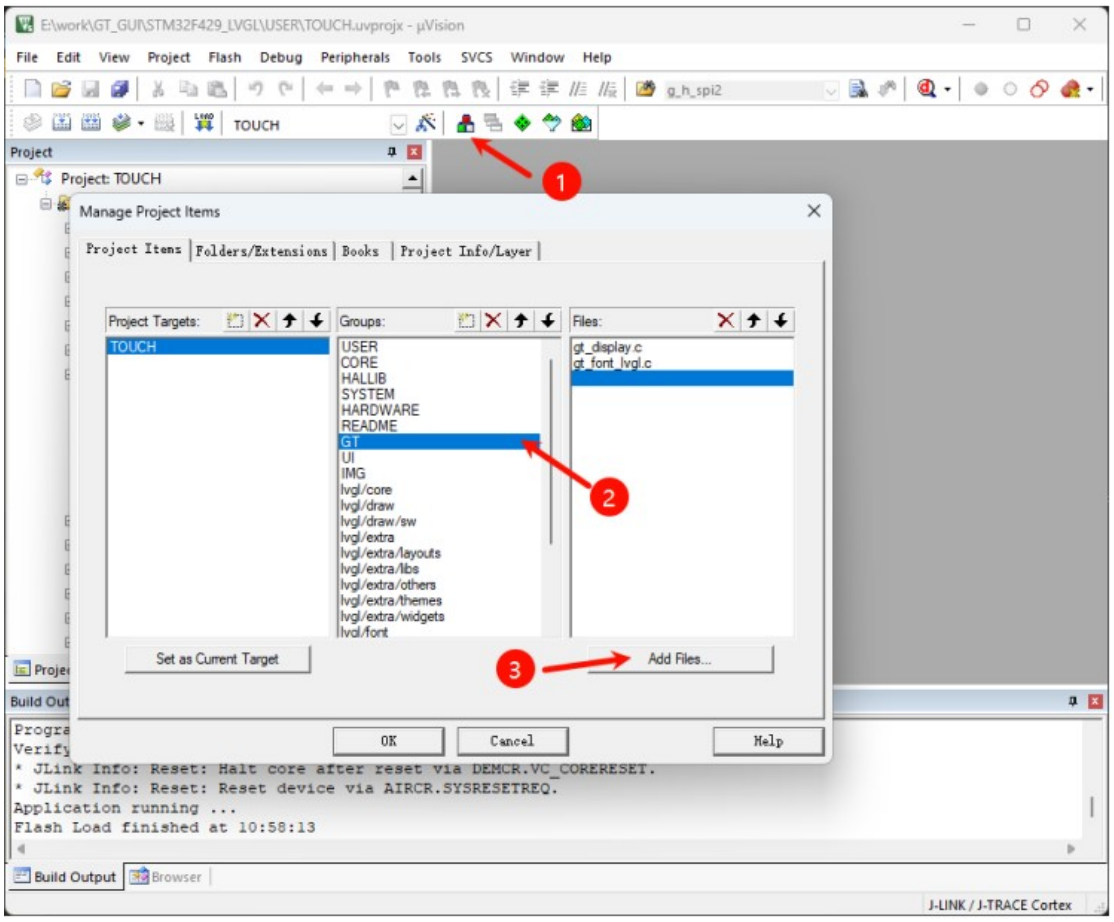
https://www.bilibili.com/video/BV1CxcyeoEnR/?vd_source=c084c395e4fcfac58df3ea21ada16b68

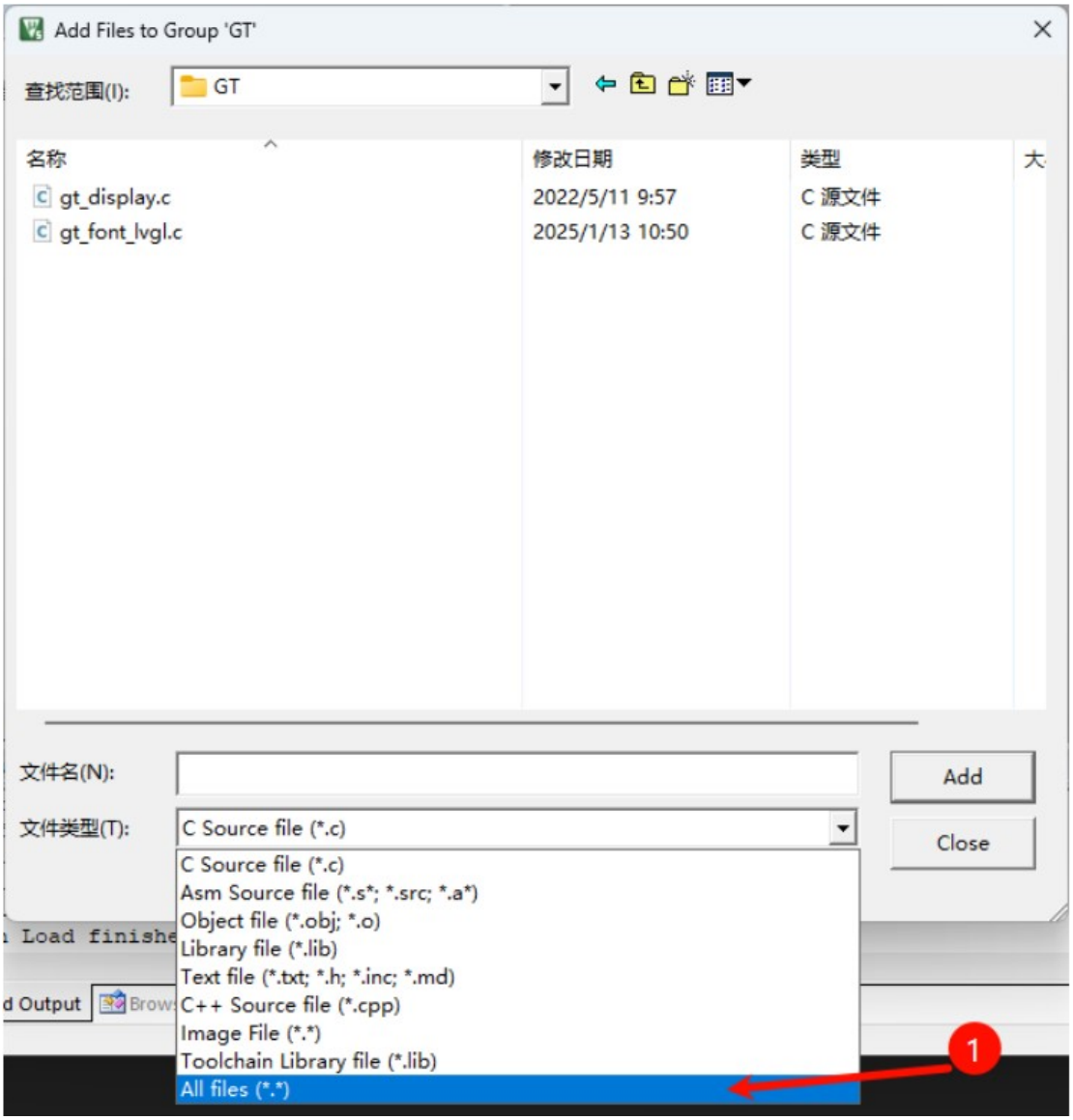
二、移植

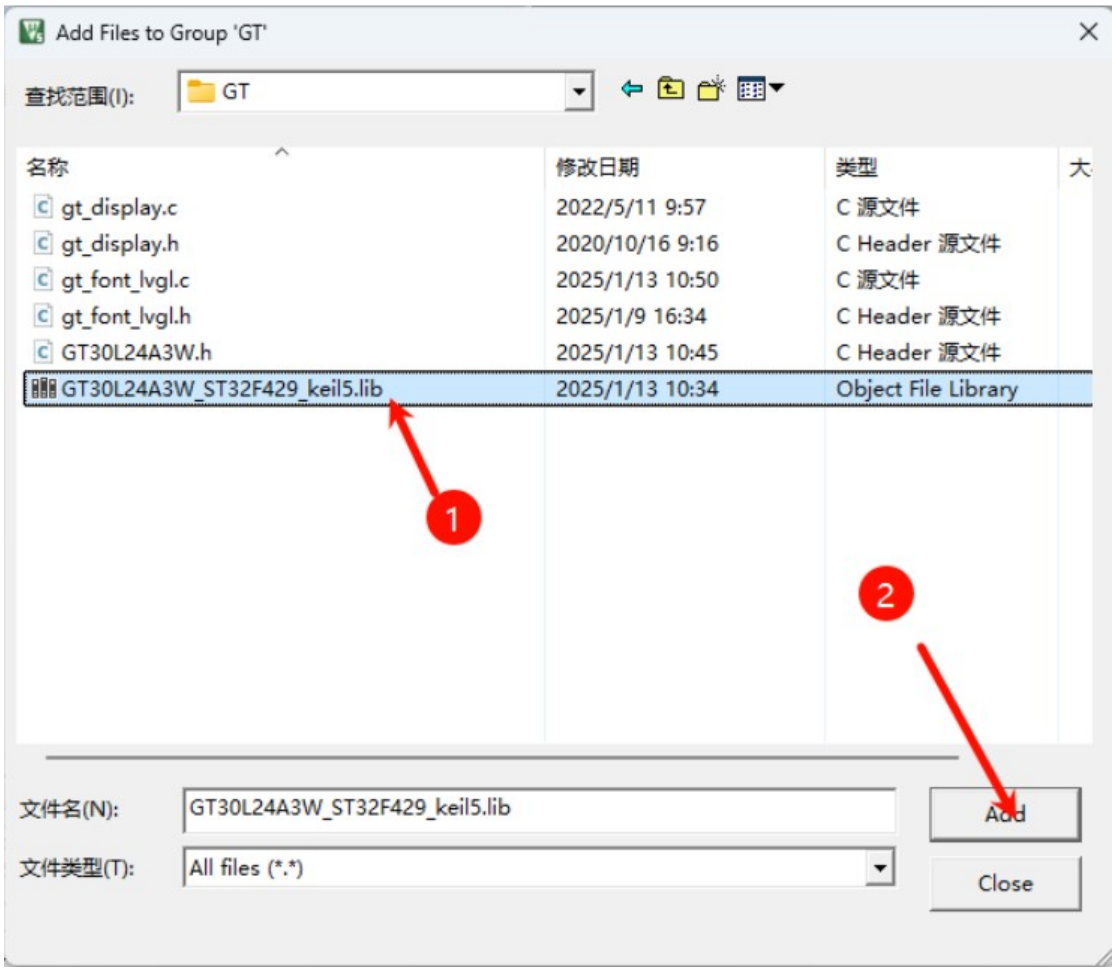
注：如果用户的编译环境是 keil，可以通过下方链接下载 mindcraft，在 mindcraft 中自动生成库功生成相应的库文件。

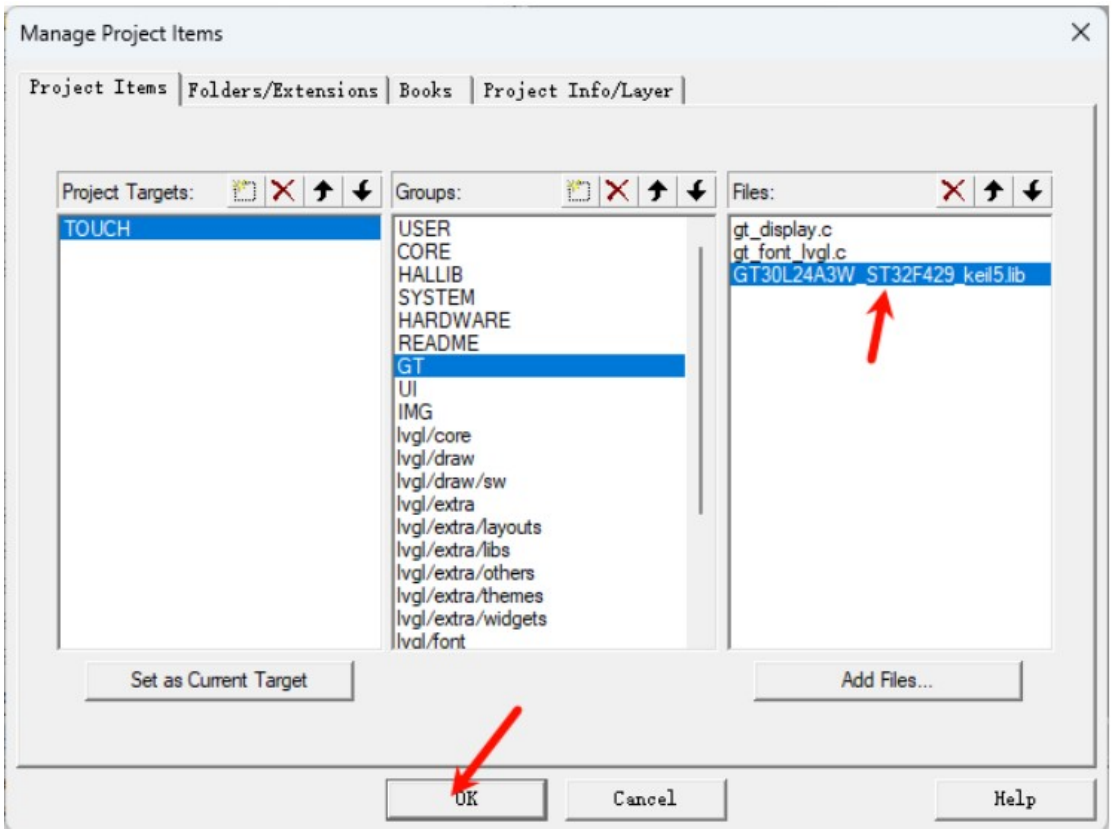
https://www.hmi.gaotongfont.cn/gtzkxpkfzl

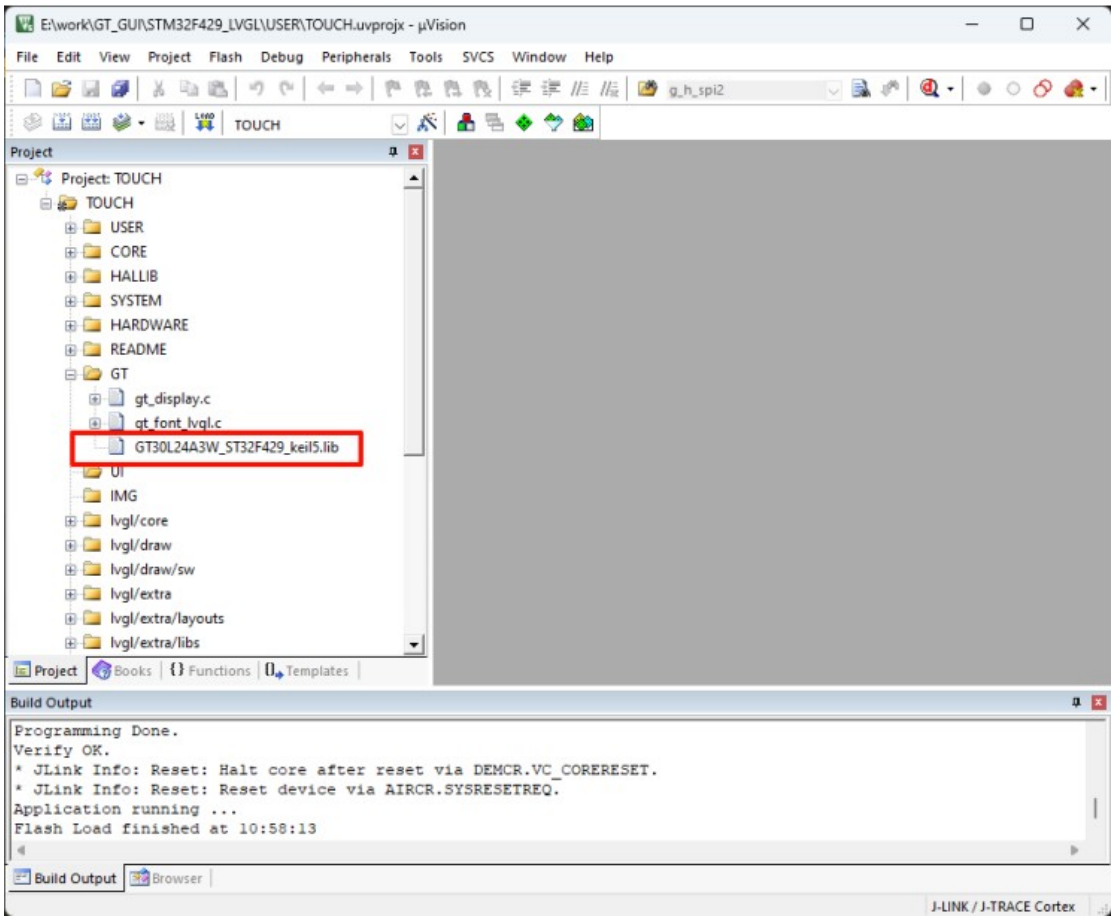
1、添加库文件（.lib 或 .a 文件）到 LVGL 项目工程中




Add Files to Group 'GT'
查找范围(I):
GT
名称
修改日期
类型
gt_display.c
2022/5/11 9:57
C 源文件
gt_font_lvgl.c
2025/1/13 10:50
C 源文件
文件名(N):
Add
文件类型(T):
C Source file (*.c)
Close
C Source file (*.c)
Asm Source file (*.s*; *.src; *.a*)
Object file (*.obj; *.o)
Library file (*.lib)
Text file (*.txt; *.h; *.inc; *.md)
C++ Source file (*.cpp)
Image File (*.*)
Toolchain Library file (*.lib)
All files (*.*)
Load finishe
d Output
1


Add Files to Group 'GT'
查找范围(I):
GT
名称
修改日期
类型
大
gt_display.c
2022/5/11 9:57
C 源文件
gt_display.h
2020/10/16 9:16
C Header 源文件
gt_font_lvgl.c
2025/1/13 10:50
C 源文件
gt_font_lvgl.h
2025/1/9 16:34
C Header 源文件
GT30L24A3W.h
2025/1/13 10:45
C Header 源文件
GT30L24A3W_ST32F429_keil5.lib
2025/1/13 10:34
Object File Library
1
2
文件名(N):
GT30L24A3W_ST32F429_keil5.lib
文件类型(T):
All files (*.*)
Add
Close


Manage Project Items
Project Items
Folders/Extensions
Books
Project Info/Layer
Project Targets:
TOUCH
Groups:
USER
CORE
HALLIB
SYSTEM
HARDWARE
README
GT
UI
IMG
lvgl/core
lvgl/draw
lvgl/draw/sw
lvgl/extra
lvgl/extra/layouts
lvgl/extra/libs
lvgl/extra/others
lvgl/extra/themes
lvgl/extra/widgets
lvgl/font
Files:
gt_display.c
gt_font_lvgl.c
GT30L24A3W_ST32F429_keil5.lib
Set as Current Target
Add Files...
OK
Cancel
Help

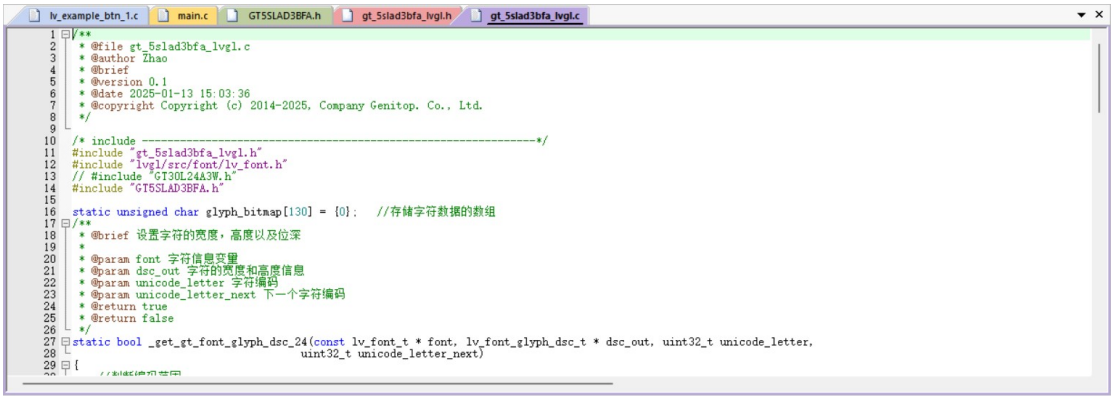


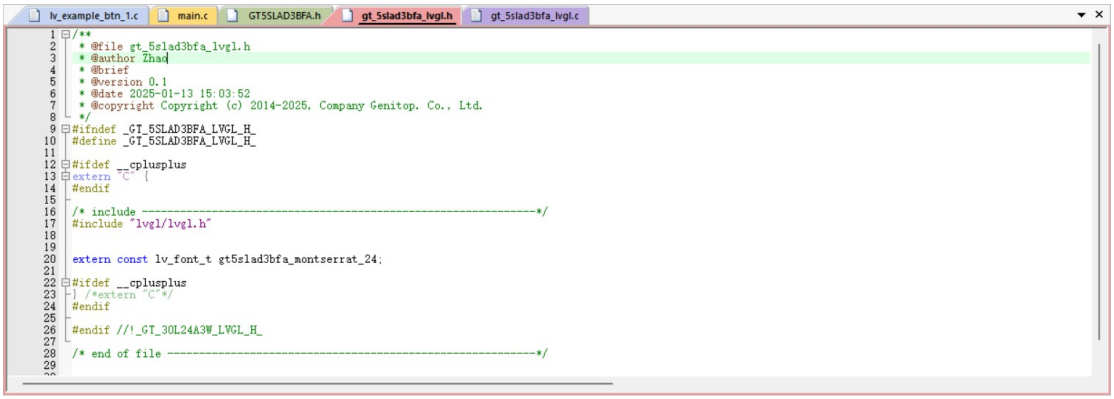
2、初始化字库

```
//字库初始化函数返回大于 0 表示初始化成功
  int ret = GT_Font_Init();
  printf("GT_Font_Init:%d\n", ret);
```

3、创建一个新的 c 文件和一个 h 文件

存放 lvgl 获取矢量字库的代码





具体 c 文件代码如下

4、定义 lv_font_t 变量

lv_font_t 类型的变量主要用来设置获取字库数据的方法以及获取字符数据信息的方法。函数指针 get_glyph_dsc 指向的是设置字符宽度和高度的函数；函数指针 get_glyph_bitmap 指向的是获取字符数据的函数，并返回存储数据的首地址。用户可以只修改这两处，其他参数默认即可。

```
//获取字符的信息和数据
#if LV_VERSION_CHECK(8, 0, 0)   //lvgl 版本号
const lv_font_t gt5slad3bfa_montserrat_24 = {
#else
lv_font_t gt5slad3bfa_montserrat_24 = {
#endif
    .get_glyph_dsc = _get_gt_font_glyph_dsc_24,             //指向获取字符信息的函数
    .get_glyph_bitmap = _get_gt_font_bitmap_fmt_txt_24,     //指向获取字符数据的函数
    .line_height = 16,              //字体需要的最大行数
    .base_line = 3,                 //从线底部测量的基线
#if !(LVGL_VERSION_MAJOR == 6 && LVGL_VERSION_MINOR == 0)   //lvgl 版本号范围
    .subpx = LV_FONT_SUBPX_NONE,    //lv_font_subpx_t 中的一个元素
```

```
#endif
#if LV_VERSION_CHECK(7, 4, 0) || LVGL_VERSION_MAJOR >= 8    //lvgl 版本号范围
    .underline_position = -1,       //下划线顶部与基线之间的距离（<0 表示基线以下）
    .underline_thickness = 1,       //下划线的厚度
#endif
    .dsc = &font_dsc           //存储的字符数据
};
```

5、定义 lv_font_fmt_txt_dsc_t 变量

lv_font_fmt_txt_dsc_t 变量主要用来设置存储字符数据的内存。用户可以提前定义好一个数组，将这个数组的首地址赋值给 glyph_bitmap 函数指针。

```
#if LV_VERSION_CHECK(8, 0, 0)   //lvgl 版本号
static  lv_font_fmt_txt_glyph_cache_t cache;    //字体的信息：字符的编码和字形的 ID
//存储字体的所有自定义数据
static const lv_font_fmt_txt_dsc_t font_dsc = {
#else
static lv_font_fmt_txt_dsc_t font_dsc = {
#endif
    .glyph_bitmap = glyph_bitmap,   // 数据存储数组，最小为一个文字大小 static unsigned char
glyph_bitmap[130] = {0};
    .glyph_dsc = NULL,              //描述字形
    .bpp = 1,                       //每像素比特数：1、2、3、4、8
    .cmaps = NULL,                  //“lv_font_cmap_fmt_txt_t”变量数组
    .kern_dsc = NULL,               //*存储字距值。可以是“lv_font_fmt_txt_ern_pair_t*”或
“lv_fent_ern_classe_fmt_xt_t” 取决于 kern_classes
    .kern_scale = 0,                //以 12.4 格式缩放 kern 值
    .cmap_num = 0,                  //cmap 表的数量
    .kern_classes = 0,              //kern_dsc 类型
    .bitmap_format = 0,             //lv_font_fmt_txt_bitmap_format_t 位图的存储格式
#if LV_VERSION_CHECK(8, 0, 0)   //lvgl 版本号
    .cache = &cache                 //缓存最后一个字母和字形 id
#endif
};
```

6、实现 _get_gt_font_glyph_dsc_24() 函数

该函数用来设置 lvgl 最后显示字符的时候需要的参数信息，例如字符的宽度和高度。

```
#include "gt_30124a3w_lvgl.h"
```

```c
#include "lvgl/src/font/lv_font.h"
#include "GT5SLAD3BFA.h"

tatic unsigned char glyph_bitmap[130] = {0};   //存储字符数据的数组
/**
 * @brief 设置字符的宽度，高度以及位深
 *
 * @param font 字符信息变量
 * @param dsc_out 字符的宽度和高度信息
 * @param unicode_letter 字符编码
 * @param unicode_letter_next 下一个字符编码
 * @return true
 * @return false
 */
static bool _get_gt_font_glyph_dsc_24(const lv_font_t * font, lv_font_glyph_dsc_t * dsc_out,
uint32_t unicode_letter,
                                      uint32_t unicode_letter_next)
{
    //判断编码范围
    if(unicode_letter >= 0x20 && unicode_letter < 0x80){
        dsc_out->adv_w = 24;     // 文字的实际宽度
        dsc_out->box_w = 24;     // 文字的显示宽度
        dsc_out->ofs_y = -2;    // 文字的显示位置偏移
    }
    else{
        dsc_out->adv_w = 24;    // 文字的实际宽度
        dsc_out->box_w = 24;    // 文字的显示宽度
        dsc_out->ofs_y = 0;     // 文字的显示位置偏移
    }

    dsc_out->box_h = 24;        // 文字的显示高度
    dsc_out->ofs_x = 0;         // 文字的显示位置偏移
    dsc_out->bpp   = 1;         // 文字的位深
    dsc_out->is_placeholder = false; // 是否是占位符
    return true;
}
```

7、实现 _get_gt_font_bitmap_fmt_txt_24 函数

该函数内通过调用矢量字库的 get_font 函数，获取字库 flahs 里面的字符数据，获取到的字符数据会存放到 lv_font_fmt_txt_dsc_t 变量的成员 glyoh_bitmap 指向的数组。get_font 可以根据用户的需求设置获取字符的宽度，高度以及粗细。

```
/**
 * @brief 获取字符数据
 *
 * @param font 字符信息变量
 * @param unicode_letter 字符编码
 * @return const uint8_t*
 */
static const uint8_t * _get_gt_font_bitmap_fmt_txt_24(const lv_font_t * font, uint32_t
unicode_letter)
{
    lv_font_fmt_txt_dsc_t * fdsc = (lv_font_fmt_txt_dsc_t *)font->dsc;  //赋值存储字符数据的数组
的首地址

    //判断位图的存储格式
    if(fdsc->bitmap_format == LV_FONT_FMT_TXT_PLAIN) {
        //判断编码范围
        if(unicode_letter >= 0x20 && unicode_letter < 0x80){
            // 读取 ASCII 字符
            get_font(&fdsc->glyph_bitmap[0], VEC_FT_ASCII_STY, unicode_letter, 24, 24, 24);
            return &fdsc->glyph_bitmap[0];  //返回存储数组的首地址
        }
        else{
    #if 01
            // Unicode 转换为 GBK 编码
            uint32_t gbk = U2G(unicode_letter);
            if(0 == gbk){
                return NULL;
            }
            // 读取 GBK 字符
            get_font(&fdsc->glyph_bitmap[0], VEC_SONG_STY, unicode_letter, 24, 24, 24);
    #else
            U2G_GetData_16X16(unicode_letter, &fdsc->glyph_bitmap[0]);
    #endif
            return &fdsc->glyph_bitmap[0];  //返回存储数组的首地址
        }
    }
    return NULL;
```

```
}
```

8、使用

```
lv_obj_set_style_text_font(label, &gt5slad3bfa_montserrat_24, 0);//获取字库数据
```