

STM32CubeMX学习笔记 (40) ——LVGL嵌入式图形库使用

原创


Leung_ManWah

已于 2022-01-28 11:16:49 修改

4578

已收藏 77

分类专栏: STM32CubeMX 文章标签: stm32 STM32CubeMX LVGL 8.1 LVGL GUI

 STM32CubeMX 专栏收录该内容


一、LVGL简介

LVGL (Light and Versatile Graphics Library) 轻量级通用型图形库，是一个免费的开源 **图形库**，提供了创建嵌入式 GUI 低内存占用等特点。支持触摸屏操作，移植简单方便，开发者一直在不断完善更新。

特点：

- 丰富且强大的模块化**图形组件**：按钮 (buttons)、图表 (charts)、列表 (lists)、滑动条 (sliders)、图片 (images) 等
- 高级的图形引擎：动画、抗锯齿、透明度、平滑滚动、图层混合等效果
- 支持多种**输入设备**：触摸屏、键盘、编码器、按键等
- 支持**多显示设备**
- 不依赖特定的硬件平台，可以在任何显示屏上运行
- 配置可裁剪（最低资源占用：64 kB Flash，16 kB RAM）
- 基于UTF-8的多语种支持，例如中文、日文、韩文、阿拉伯文等
- 可以通过**类CSS**的方式来设计、布局图形界面（例如：**Flexbox**、**Grid**）
- 支持操作系统、外置内存、以及硬件加速（LVGL已内建支持STM32 DMA2D、NXP PXP和VGLite）
- 即便仅有**单缓冲区(frame buffer)**的情况下，也可保证渲染如丝般顺滑
- 全部由C编写完成，并支持C++调用
- 支持Micropython编程，参见：**LVGL API in Micropython**
- 支持**模拟器**仿真，可以无硬件依托进行开发
- 丰富详实的**例程**
- 详尽的**文档**以及API参考手册，可线上查阅或可下载为PDF格式

- LVGL官网：<https://lvgl.io>
- 官方文档：<https://docs.lvgl.io/master/intro/index.html>
- Github仓库：<https://github.com/lvgl/lvgl>
- 国内码云仓库：<https://gitee.com/mirrors/lvgl>

 Leung_ManWah

关注

18

77

3

专栏目录

[zshpin/4free/littleVGL.html](#)

二、FSMC配置LCD屏显示和触摸

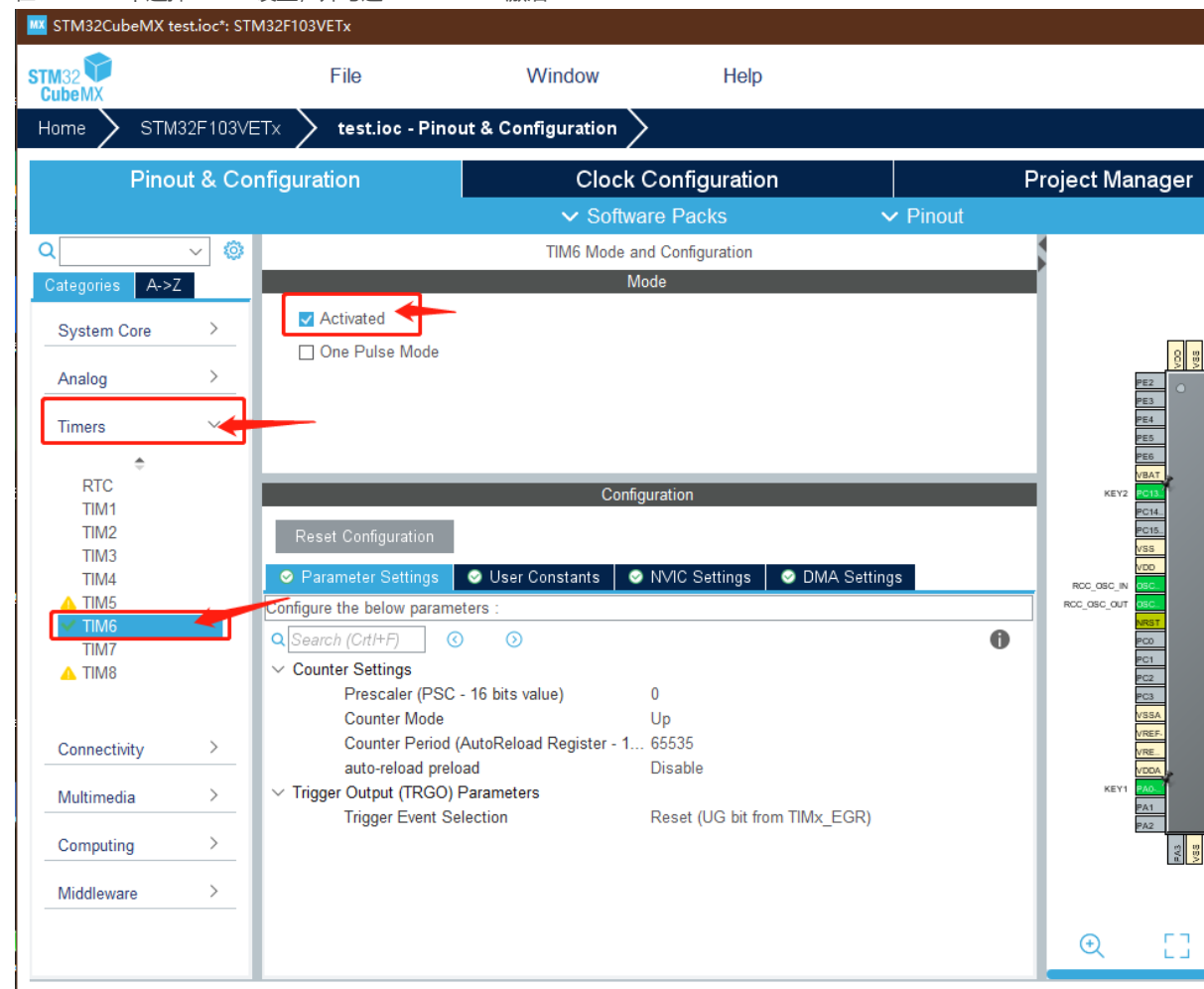
查看 STM32CubeMX学习笔记 (38) ——FSMC接口使用(TFT-LCD屏显示)

查看 STM32CubeMX学习笔记 (39) ——FSMC接口使用(TFT-LCD屏触摸)

三、TIM6基本定时器（可跳过，看LVGL心跳的配置方式选择）

3.1 参数配置

在 Timers 中选择 TIM6 设置，并勾选 Activated 激活



Leung_ManWah

关注

18



77

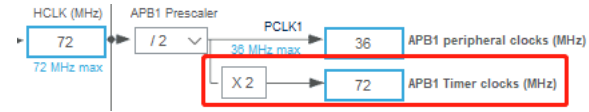


3



专栏目录

Tclk 即内部时钟CK_INT，经过APB1预分频器后分频提供，如果APB1预分频系数等于1，则频率不变，否则频率乘以2，库函数中时钟Tclk=36*2=72M。



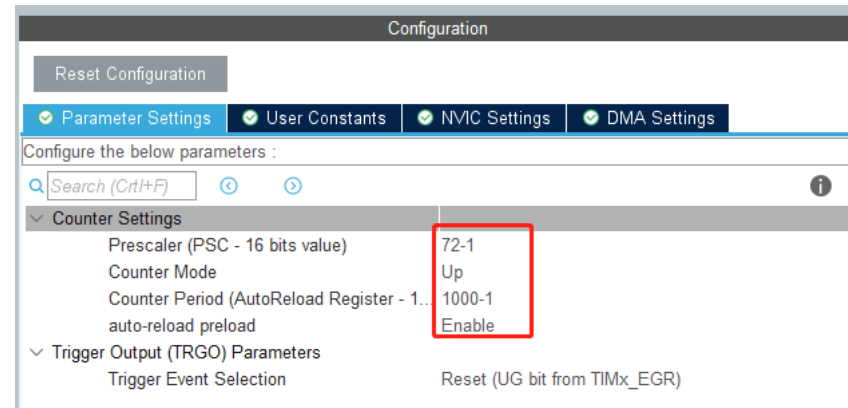
定时器溢出时间：

$$T_{out} = 1 / (T_{clk} / (psc + 1)) * (arr + 1)$$

- 定时器时钟Tclk: 72MHz
- 预分频器psc: 71
- 自动重载寄存器arr: 999

即 $T_{out} = 1 / (72\text{MHz} / (71+1)) * (999+1) = 1\text{ms}$

- **Prescaler (时钟预分频数) : 72-1** 则驱动计数器的时钟 $CK_CNT = CK_INT(即72\text{MHz}) / (71+1) = 1\text{MHz}$
- **Counter Mode (计数模式) : Up (向上计数模式)** 基本定时器只能是向上计数
- **Counter Period (自动重载值) : 1000-1** 则定时时间 $1/CK_CLK * (999+1) = 1\text{ms}$
- **auto-reload-preload (自动重载) : Enable (使能)**
- **TRGO Parameters (触发输出) : 不使能** 在定时器的定时时间到达的时候输出一个信号 (如: 定时器更新产生TRGO信号来触发ADC的同



3.2 配置NVIC



Leung_ManWah

关注

18



77

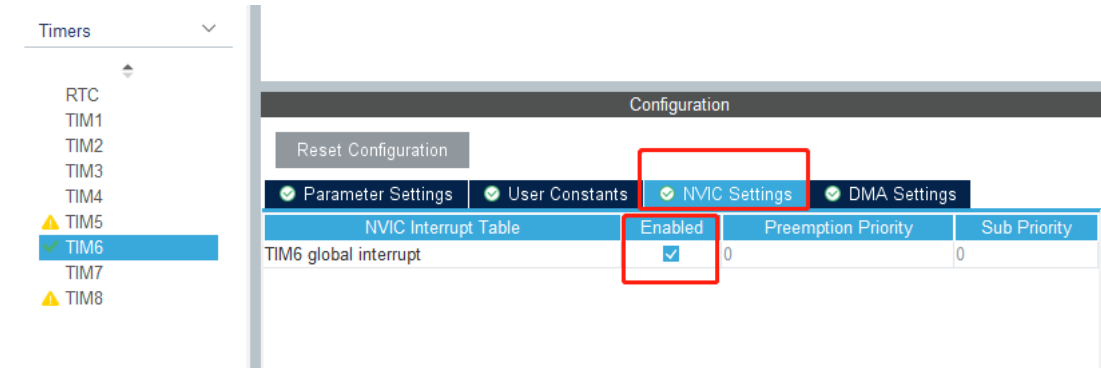


3



专栏目录

使能定时器中断



四、工程管理

4.1 增大栈空间

将最小栈空间改到 0x1000

注意：由于LVGL运行的硬件要求，故需要加大项目的栈空间到 2KB 和使能 C99 编译器功能，修改 Stack_Size 的值大于 2KB(0x00000800)。

Name	Minimal	Recommended
Architecture	16, 32 or 64 bit microcontroller or processor	
Clock	> 16 MHz	> 48 MHz
Flash/ROM	> 64 kB	> 180 kB
Static RAM	> 2 kB	> 4 kB
Stack	> 2 kB	> 8 kB
Heap	> 2 kB	> 8 kB
Display buffer	> 1 × hor. res. pixels	> 10 × hor. res. pixels
Compiler	C99 or newer	



Leung_ManWah

关注

18



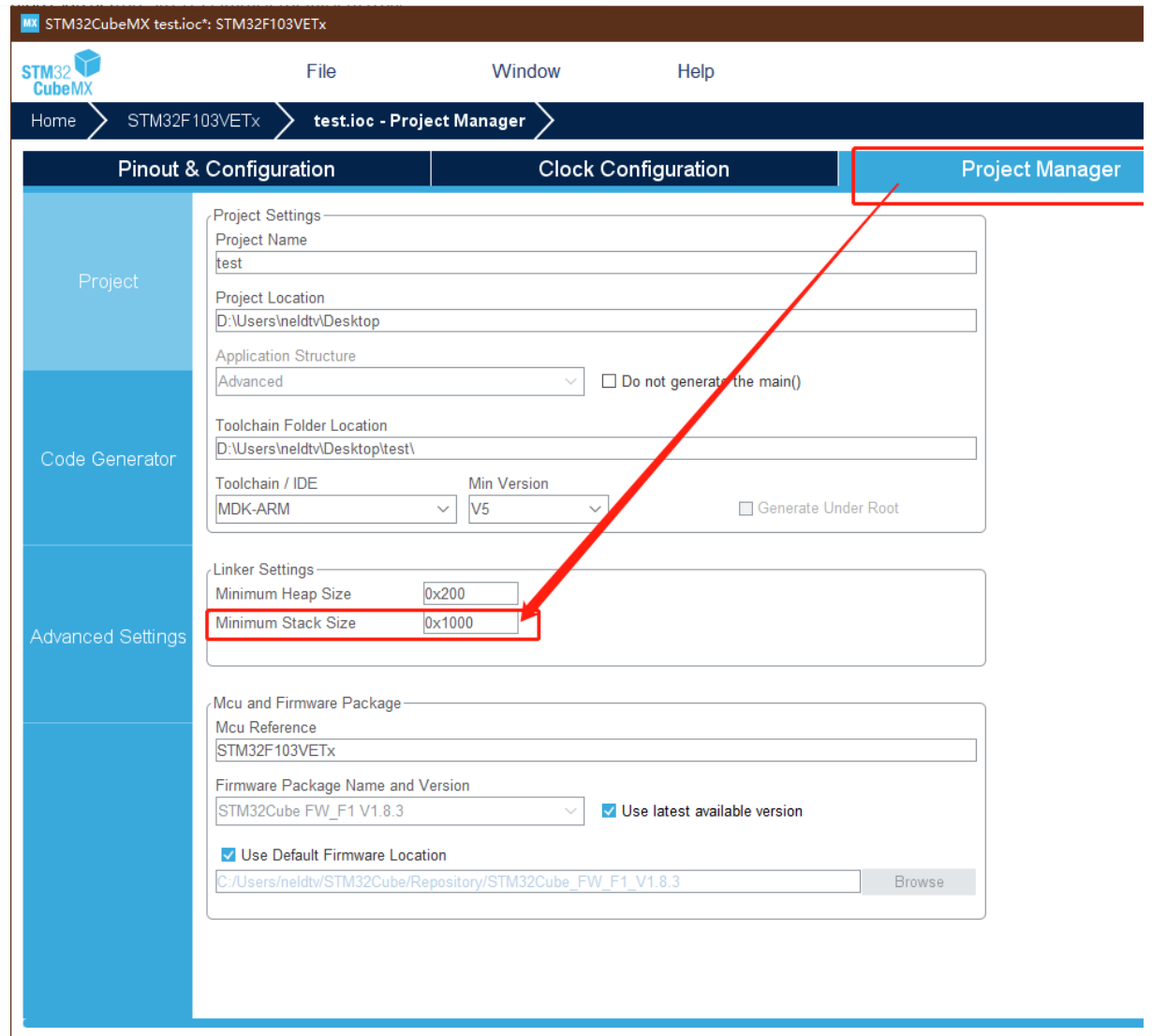
77



3



专栏目录



4.2 生成代码



Leung_ManWah

关注

18



77



3



专栏目录

输入项目名和项目路径

STM32CubeMX test.ioc: STM32F103VETx

File Window Help

Home > STM32F103VETx > test.ioc - Project Manager

Pinout & Configuration Clock Configuration Project Manager

Project

Project Settings

Project Name

test

Project Location

D:\Users\neldtv\Desktop

Application Structure

Advanced

Do not generate the main()

Toolchain Folder Location

D:\Users\neldtv\Desktop\test\

Toolchain / IDE

MDK-ARM

Min Version

V5

Generate Under Root

Linker Settings

Minimum Heap Size

0x200

Minimum Stack Size

0x400

Mcu and Firmware Package

Mcu Reference

STM32F103VETx

Firmware Package Name and Version

STM32Cube FW_F1 V1.8.2

Use latest available version

Use Default Firmware Location

C:/Users/neldtv/STM32Cube/Repository/STM32Cube_FW_F1_V1.8.2

Browse

选择应用的 IDE 开发环境 MDK-ARM V5



Leung_ManWah

关注

18



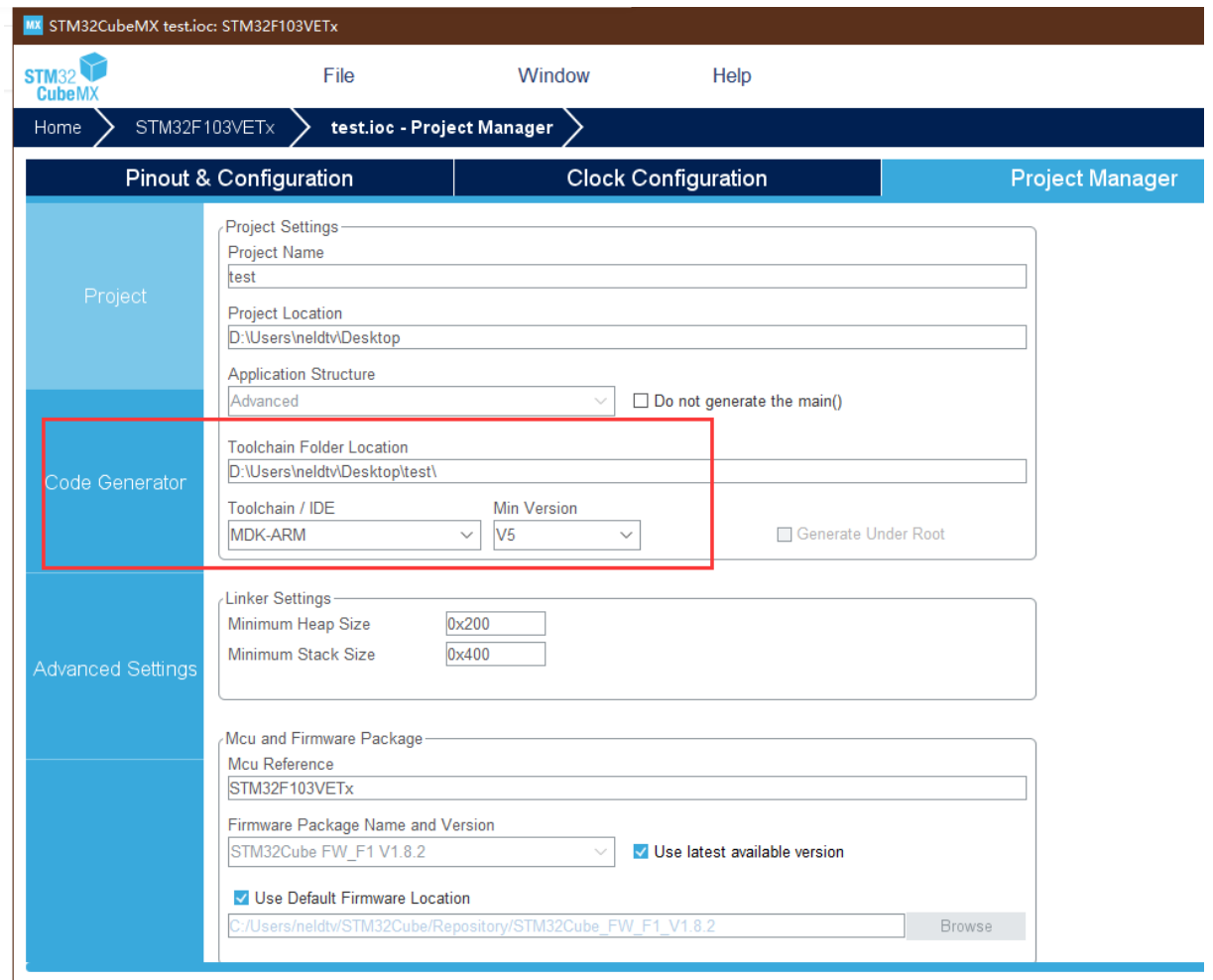
77



3



专栏目录



每个外设生成独立的 '.c/.h' 文件

不勾: 所有初始化代码都生成在 main.c

勾选: 初始化代码生成在对应的外设文件。如 GPIO 初始化代码生成在 gpio.c 中。



Leung_ManWah

关注

18



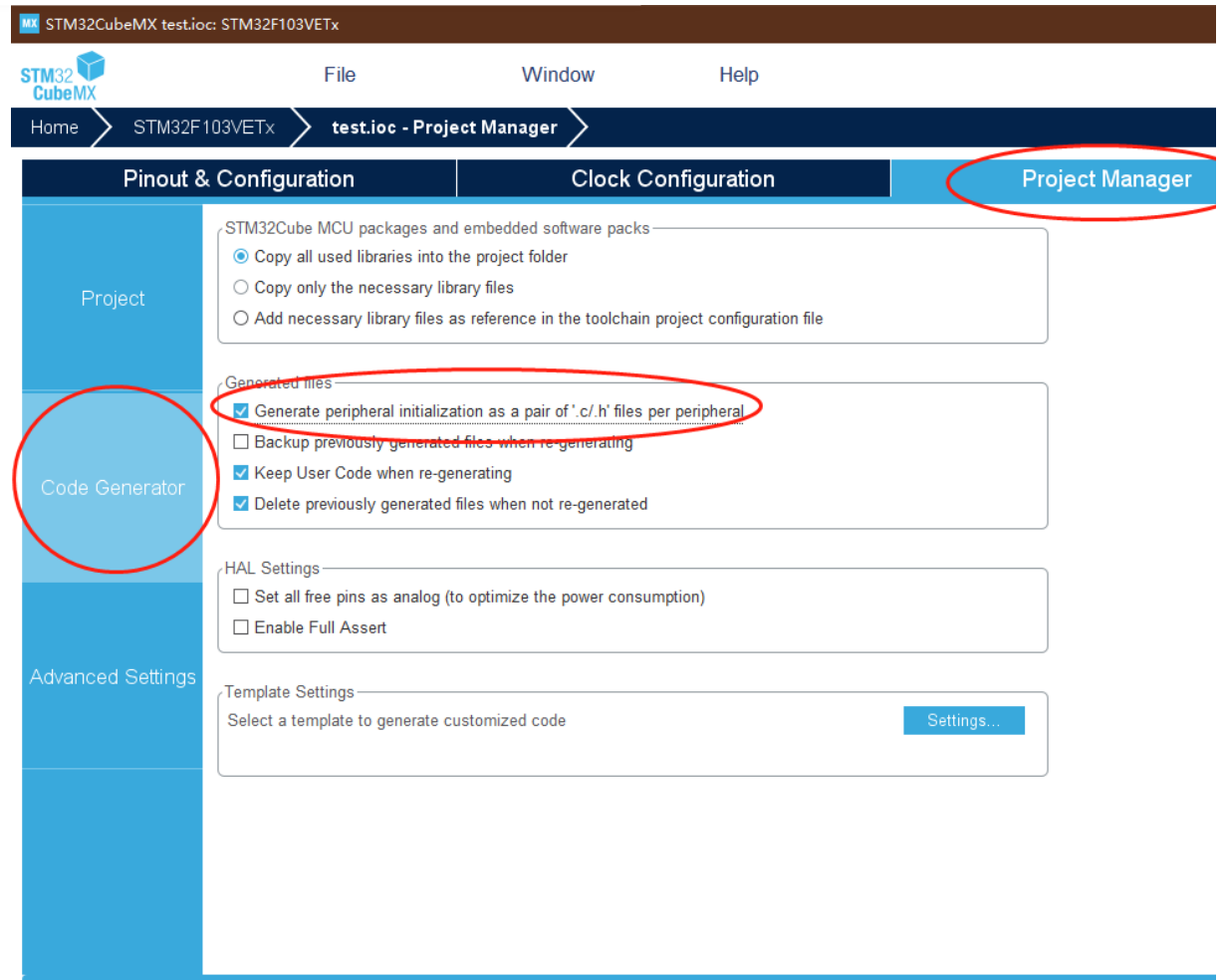
77



3



专栏目录



点击 GENERATE CODE 生成代码



Leung_ManWah

关注

18



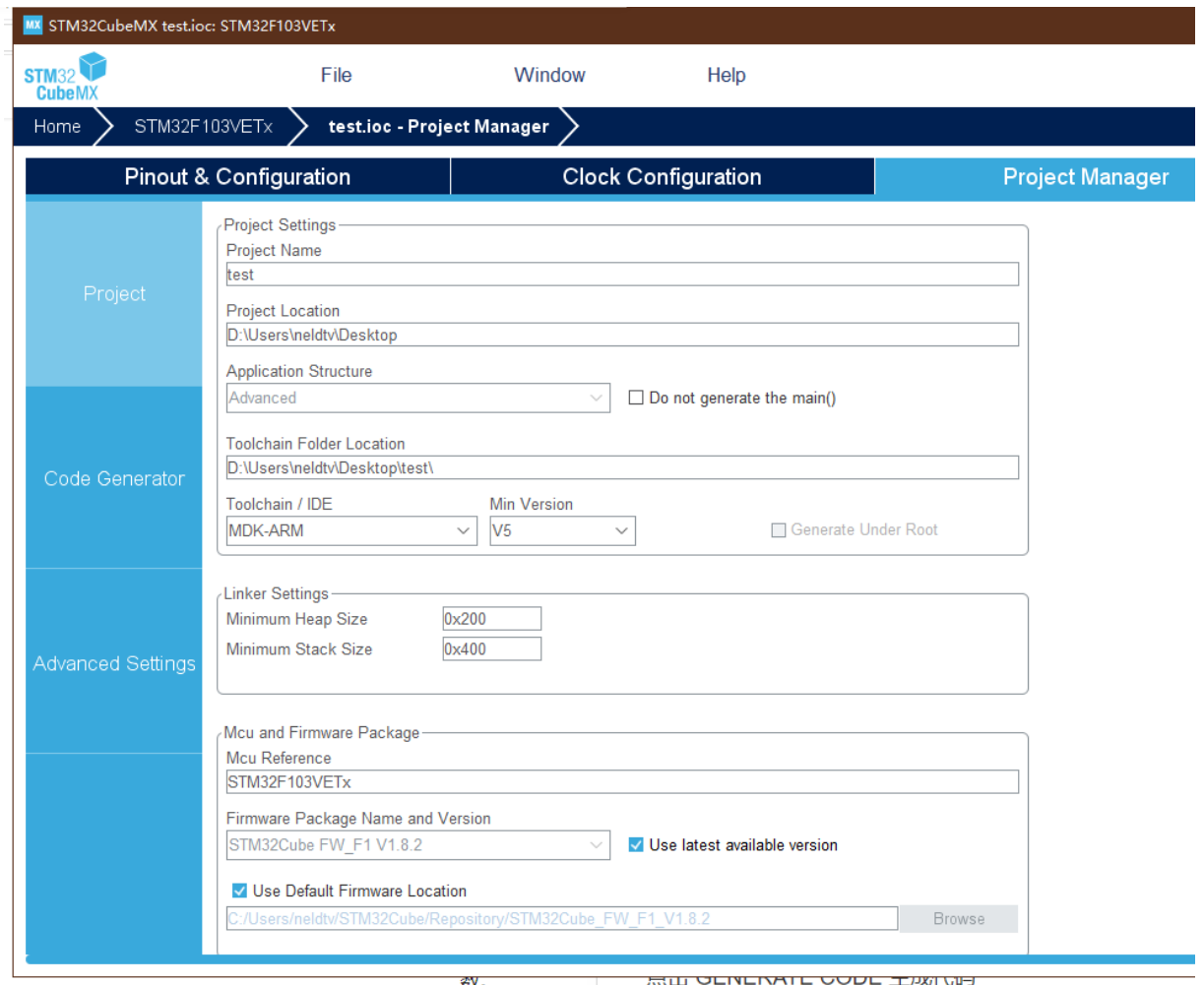
77



3



专栏目录



五、移植LVGL

5.1 下载源码

- Github仓库: <https://github.com/lvgl/lvgl>
- 国内码云仓库: <https://gitee.com/mirrors/lvgl>



Leung_ManWah

关注

18



77



3



专栏目录

我用的是 LVGL 8.1 版本

release/v8.1

分支 89 标签 49

文件

Web IDE

克隆/下载

GK Gabor Kiss-Vamosi chore(docs): fix lv_list_add_text ac31c7c 17天前 7483 次提交

.github	ci(micropython) add rp2 port	3个月前
docs	chore(docs): fix lv_list_add_text	17天前
examples	feat(example) add text with gradient example	3个月前
rt-thread	feat(rt-thread): support LVGL projects with GCC/Keil(AC5)/Keil(AC6)/IAR	3个月前
scripts	docs(changelog) improve changelog template	3个月前
src	fix(bidi): add weak characters to the previous strong character's run (#2777)	3个月前
tests	test fix LV_USE_LOG_LEVEL -> LV_LOG_LEVEL typo	3个月前
zephyr	Add support for Zephyr intergartion (#1979)	1年前
.codecov.yml	ci(codecov) hide statuses on commits for now	5个月前
.editorconfig	merge master	1年前
.gitignore	test convert Makefile to CMake (#2495)	5个月前
CMakeLists.txt	fix(config): remove the nonexistent Kconfig (#2654)	4个月前
Kconfig	fix(conf): Make LV_COLOR_MIX_ROUND_OFS configurable (#2766)	3个月前
LICENCE.txt	docs(license) update company name and year	7个月前
README.md	Update README.md (#2516)	5个月前
SConscript	feat add support for rt-thread RTOS (#2660)	4个月前

5.2 新建文件夹

- 首先在工程根目录下新建两个文件夹，命名 GUI 和 GUI_APP

GUI 目录是用来存放跟LVGL库相关的所有文件的。
GUI_APP 是用来放我们自己的 GUI 应用代码的，因为现在才刚开始移植，还来不及自己写GUI应用，所以GUI_APP目录里

Core	2020/10/16 14:07	文件夹	
Drivers	2022/1/20 14:31	文件夹	
GUI	2022/1/26 13:41	文件夹	
GUI_APP	2022/1/25 16:37	文件夹	
MDK-ARM	2022/1/26 17:35	文件夹	
Middlewares	2021/3/31 15:09	文件夹	
.mxproject	2022/1/22 13:50	MXPROJECT 文件	7 KB
test	2022/1/22 13:50	STM32CubeMX	10 KB

- 然后在 GUI 文件夹下新建三个空文件夹，lvgl、lvgl_port

lvgl 官方各类控件的源程序。

lvgl_port 用于存放LVGL显示屏驱动、输入设备驱动及文件系统驱动。

脑 > 桌面 > test > GUI	
名称	修改日期
lvgl	2022/1/26 13:41
lvgl_port	2022/1/26 16:17

5.3 拷贝LVGL src文件和根目录lvgl.h到lvgl文件夹

src 所有源码都在项目根目录的src文件夹里。

lvgl.h 包含了LVGL库中的所有头文件。



Leung_ManWah

关注

18



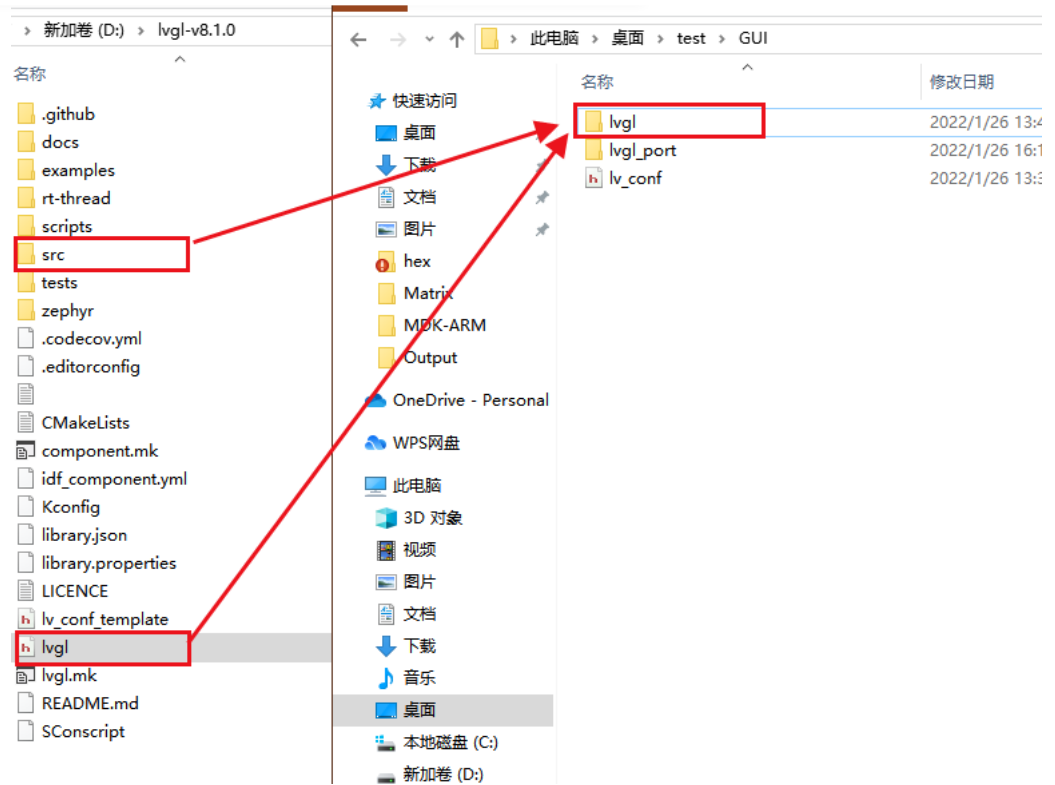
77



3



专栏目录



5.4 拷贝LVGL examples/porting文件到lvgl_port文件夹

复制"lvgl-8.1.0\examples\porting"中的文件到工程目录下的"GUI\lvgl_port"目录下。并将他们改名 去掉template 。

lv_port_disp 为LVGL显示驱动。
lv_port_fs 为LVGL文件系统驱动。
lv_port_indev 为LVGL输入设备驱动。

5.5 拷贝配置文件lv_conf_template.h到GUI文件夹

复制LVGL库根目录下的"lv_conf_template.h"文件到工程文件夹"GUI"下，并将"lv_conf_template.h"重命名为"lv_conf.h"。

lv_conf.h 是LVGL库的配置文件，里面有各种宏。



Leung_ManWah

关注

18



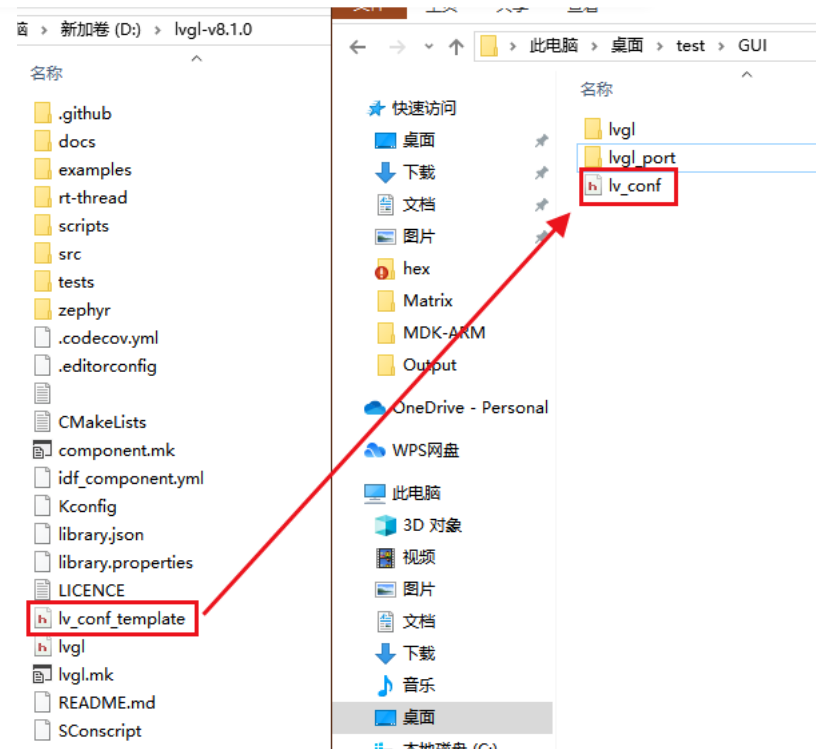
77



3



专栏目录



5.6 添加源码到工程组文件夹

接下来我们在 mdk 里面新建 `GUI/lvgl` 和 `GUI/lvgl_port` 两个组文件夹，其中 `GUI/lvgl` 用于存放 `src` 文件夹的内容，`GUI/l`

在`GUI/lvgl`组中添加以下文件夹中所有的.c文件：

```
GUI/lvgl/src/lv_core
GUI/lvgl/src/lv_draw
GUI/lvgl/src/lv_extra (除了lib外，除非你用到了相关功能)
GUI/lvgl/src/lv_font
GUI/lvgl/src/lv_hal
GUI/lvgl/src/lv_misc
GUI/lvgl/src/lv_themes
GUI/lvgl/src/lv_widgets
//注意不要添加 GUI/lvgl/src/lv_gpu 中的文件，除非你用到了相关功能
```



Leung_ManWah

关注

18



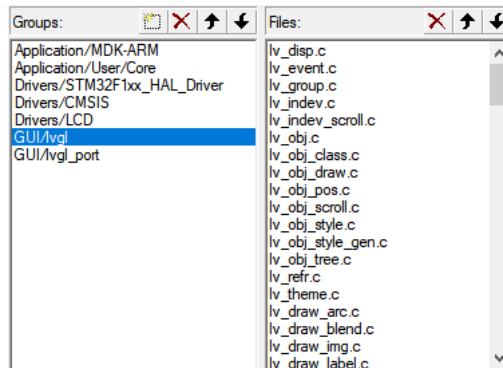
77



3



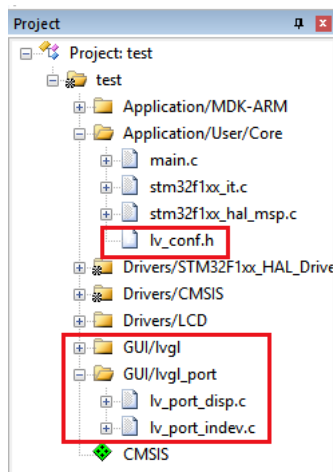
专栏目录



在GUI/lvgl_port组中添加以下.c文件：

```
GUI/lvgl_port/lv_port_disp.c
GUI/lvgl_port/lv_port_indev.c
//注意不要添加 GUI/lvgl_port/lv_port_fs.c 中的文件，除非你用到了相关功能
```

在User组中添加lvgl_conf.h配置文件：



5.7 指定头文件路径

LVGL 的源码已经添加到开发环境的组文件夹下面，编译的时候需要为这些源文件指定头文件的路径，不然编译会报错。只需要将面指定即可。



Leung_ManWah

关注

18



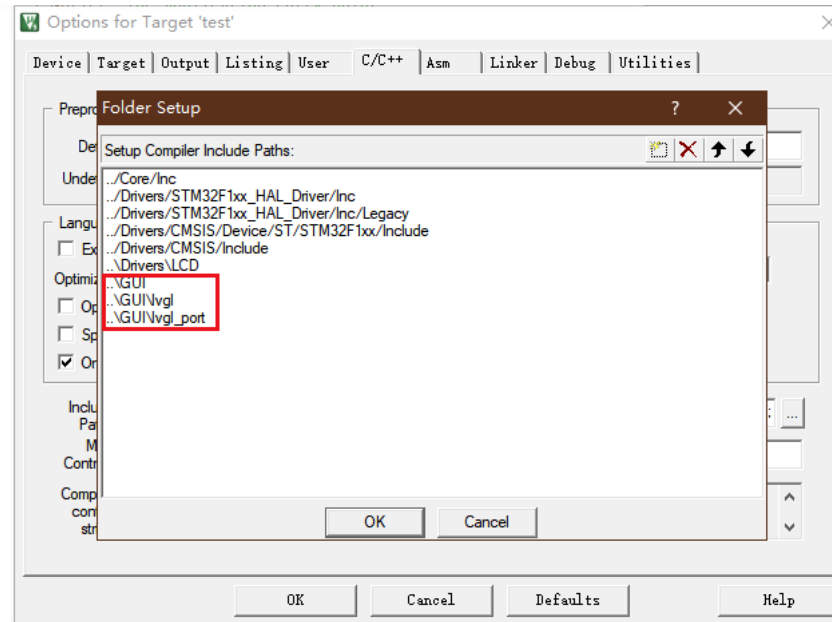
77



3

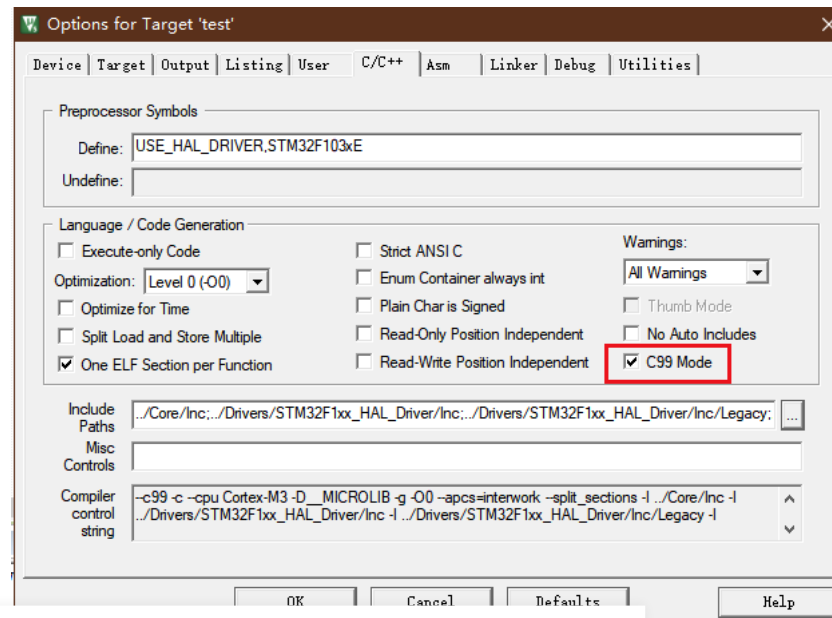


专栏目录



5.8 设置编译参数

C99: LVGL要求 C99 或更新的编译器, 否则编译是会报错的。



Leung_ManWah

关注

18



77



3



专栏目录

5.9 修改FreeRTOSConfig.h

- 使能配置文件

在第15行设置为`#if 1`

```
/**
 * @file lv_conf.h
 * Configuration file for v8.1.0
 */

/*
 * Copy this file as `lv_conf.h`
 * 1. simply next to the `lvgl` folder
 * 2. or any other places and
 *    - define `LV_CONF_INCLUDE_SIMPLE`
 *    - add the path as include path
 */

/* clang-format off */
#if 1 /*Set it to "1" to enable content*/
```

- 颜色设置

定义颜色深度，如果是单色屏的话就改为1

```
/*=====
 * COLOR SETTINGS
 *=====*/

/*Color depth: 1 (1 byte per pixel), 8 (RGB332), 16 (RGB565), 32 (ARGB8888)*/
#define LV_COLOR_DEPTH 16
```

- 内存设置

给LVGL分配动态内存RAM的大小，至少需要2k，资源允许的情况下可以稍微设大些，这个设置过小的话，在跑一些稍微复杂

```
/*=====
 * MEMORY SETTINGS
 *=====*/

/*1: use custom malloc/free, 0: use the built-in `lv_mem_alloc()` and `lv_mem_free()`*/
#define LV_MEM_CUSTOM 0
#if LV_MEM_CUSTOM == 0
/*Size of the memory available for `lv_mem_alloc()` in bytes (>= 2kB)*/
#define LV_MEM_SIZE (1024 * 1024) /*1MB*/
#endif
```



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

tes]*/

- 底层设置

```
/*Use a custom tick source that tells the elapsed time in milliseconds.
 *It removes the need to manually update the tick with `lv_tick_inc()` */
/* 时钟源提供者, 如果LV_TICK_CUSTOM==1, 那么就不用lv_tick_inc()提供时钟了 */
#define LV_TICK_CUSTOM    1
#if LV_TICK_CUSTOM
/* ↓ 这里可以指定时钟源提供者, 例如STM32的HAL库的HAL_GetTick() */
#define LV_TICK_CUSTOM_INCLUDE "stm32f1xx_hal.h" /*Header for the system time function*/
#define LV_TICK_CUSTOM_SYS_TIME_EXPR (HAL_GetTick())
#endif /*LV_TICK_CUSTOM*/

/*Default Dot Per Inch. Used to initialize default sizes such as widgets sized, style paddings.
 *(Not so important, you can adjust it to modify default sizes and spaces)*/
#define LV_DPI_DEF 130 /*[px/inch]*/
//用来调节界面缩放比例的, 此值越大, 控件分布的就越散, 控件自身的间隔也会变大。可根据实际情况进行更改, 例如128x128分辨率1
```

- 监控设置

显示CPU运行效率和FPS等

```
/*1: Show CPU usage and FPS count in the right bottom corner*/
#define LV_USE_PERF_MONITOR 0
#if LV_USE_PERF_MONITOR
#define LV_USE_PERF_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
#endif

/*1: Show the used memory and the memory fragmentation in the left bottom corner
 * Requires LV_MEM_CUSTOM = 0*/
#define LV_USE_MEM_MONITOR 0
#if LV_USE_PERF_MONITOR
#define LV_USE_MEM_MONITOR_POS LV_ALIGN_BOTTOM_LEFT
#endif
```

六、修改显示驱动接口

- lv_port_disp.c

使能文件及添加头文件

```
/*Copy this file as "lv_port_disp.c" and set this value to "1" to enable content*/
#if 1
```



Leung_ManWah

关注

👍 18



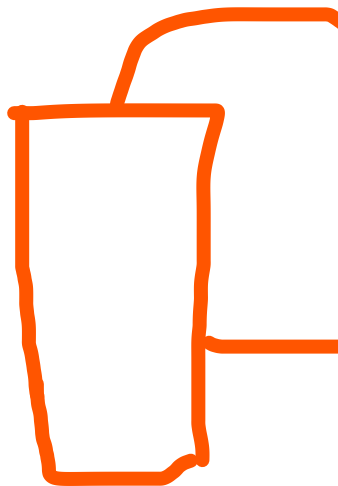
🌟 77



💬 3



专栏目录



```
#include "lv_port_disp.h"
#include "lvgl.h"
#include "bsp_ili9341_lcd.h"
```

添加及修改屏幕像素高度和宽度，根据实际屏幕尺寸

```
/*
 *   DEFINES
 */
#define MY_DISP_HOR_RES    (320)    // 屏幕像素高度
#define MY_DISP_VER_RES    (240)    // 屏幕像素宽度
...
/*Set the resolution of the display*/
disp_drv.hor_res = MY_DISP_HOR_RES;
disp_drv.ver_res = MY_DISP_VER_RES;
```

找到 `disp_init()` 函数，将显示屏初始化驱动 `ILI9341_Init()` 放到这里

```
/*Initialize your display and the required peripherals.*/
static void disp_init(void)
{
    ILI9341_Init();
}
```

修改 `lv_port_disp_init()` 函数

```
void lv_port_disp_init(void)
{
    /*-----
    * Initialize your display
    * -----*/
    disp_init(); // 显示屏驱动初始化

    /*-----
    * Create a buffer for drawing
    * -----*/
```

修改 `disp_flush()` 函数，将自己显示屏对应的填充颜色块函数放到这里，这个函数是用来刷新显示区域的，速度越快越好



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

```

/*Flush the content of the internal buffer the specific area on the display
*You can use DMA or any hardware acceleration to do this operation in the background but
*'lv_disp_flush_ready()' has to be called when finished.*/
static void disp_flush(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_p)
{
    /*The most simple case (but also the slowest) to put all pixels to the screen one-by-one*/

    int32_t x;
    int32_t y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {

```

```

static void disp_flush(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t
{
    /*The most simple case (but also the slowest) to put all pixels to the screen or

    int32_t x;
    int32_t y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            ILI9341_DrawPixel(x, y, color_p->full);
            color_p++;
        }
    }

    /* IMPORTANT!!!
    * Inform the graphics library that you are ready with the flushing*/
    lv_disp_flush_ready(disp_drv);
}

```

红色标注部分的函数也就是以单个像素点填充屏幕的函数，这个函数野火写的不足调用要求，稍微将原来的驱动代码进行了更改

```

void ILI9341_DrawPixel ( uint16_t usX, uint16_t usY, uint16_t uColor )
{
    if ( ( usX < LCD_X_LENGTH ) && ( usY < LCD_Y_LENGTH ) )
    {
        ILI9341_SetCursor ( usX, usY );
        ILI9341_FillColor ( 1, uColor );
    }
}

```



Leung_ManWah

关注

18



77



3



专栏目录

- **lv_port_disp.h**

使能文件及声明函数

```
/*Copy this file as "lv_port_disp.h" and set this value to "1" to enable content*/
#if 1
...
...
/*****
 * GLOBAL PROTOTYPES
 *****/
void lv_port_disp_init(void);
```

七、修改输入设备驱动接口

- **lv_port_indev.c**

使能文件及添加头文件

```
/*Copy this file as "lv_port_indev.c" and set this value to "1" to enable content*/
#if 1

/*****
 * INCLUDES
 *****/
#include "lv_port_indev.h"
#include "lvgl.h"
#include "bsp_xpt2046_lcd.h"
```

找到 `touchpad_init()` 函数，将触摸屏初始化驱动 XPT2046_Init_Init() 放到这里

```
/*-----
 * Touchpad
 * -----*/

/*Initialize your touchpad*/
static void touchpad_init(void)
{
    XPT2046_Init();
}
```

修改 `lv_port_indev_init()` 函数，这里是初始化输入设备驱动和在LVGL中注册一个输入设备。输入设备可以是触摸屏、鼠标、除。



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

```
static lv_indev_drv_t indev_drv;

/*-----
 * Touchpad
 * -----*/

/*Initialize your touchpad if you have*/
touchpad_init();
```

修改 `touchpad_is_pressed()` 和 `touchpad_get_xy()` 函数

```
/*Return true is the touchpad is pressed*/
static bool touchpad_is_pressed(void) // 判断是否触摸
{
    if(TOUCH_PRESSED == XPT2046_TouchDetect())
    {
        return true;
    }
    return false;
}

/*Get the x and y coordinates if the touchpad is pressed*/
```

• `lv_port_indev.h`

使能文件及声明函数

```
/*Copy this file as "lv_port_indev.h" and set this value to "1" to enable content*/
#if 1
...
...
/*****
 * GLOBAL PROTOTYPES
 *****/
void lv_port_indev_init(void);
```



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

8.1 包含头文件

```
/* USER CODE END Header */
/* Includes -----*/
#include "main.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include "bsp_ili9341_lcd.h"
#include "bsp_xpt2046_lcd.h"
#include "lv_port_disp.h"
#include "lv_port_indev.h"
#include "lvgl.h"
```

8.2 初始化LVGL

```
lv_init();           // lvgl初始化, 如果这个没有初始化, 那么下面的初始化会崩溃
lv_port_disp_init(); // 显示器初始化
lv_port_indev_init(); // 输入设备初始化 (如果没有实现就注释掉)
```

8.3 配置LVGL心跳

8.3.1 配置LV_TICK_CUSTOM方式 (推荐, 选择其中一种)

过 x 毫秒调用 `lv_tick_inc(x)` 函数一次 ($1 \leq x \leq 10$) , 这个函数是LittlevGL运行所需的时钟源。

如果定义 `LV_TICK_CUSTOM` 为 1 的话, 就无须在应用程序中主动调用 `lv_tick_inc(x)` 函数, 而是需要定义一个获取当前系统已运行时间戳。
`LV_TICK_CUSTOM_SYS_TIME_EXPR` 表示该函数, 这个函数会在调用 `lv_task_handler()` 函数的时候自动调用并获取当前时间戳。

```
/*Use a custom tick source that tells the elapsed time in milliseconds.
 *It removes the need to manually update the tick with `lv_tick_inc()`*/
#define LV_TICK_CUSTOM 1
#if LV_TICK_CUSTOM
#define LV_TICK_CUSTOM_INCLUDE "stm32f1xx_hal.h" /*Header for the system time function*/
#define LV_TICK_CUSTOM_SYS_TIME_EXPR (HAL_GetTick()) /*Expression evaluating to current system time in ms*/
#endif /*LV_TICK_CUSTOM*/
```

```
/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    ...
    ...
    /* Infinite loop */
```



Leung_ManWah

关注

18



77



3



专栏目录

8.3.1 基本定时器方式（选择其中一种）

修改中断回调函数

打开 `stm32f1xx_it.c` 中断服务函数文件，找到 TIM6 中断的服务函数 `TIM6_IRQHandler()`
中断服务函数里面就调用了定时器中断处理函数 `HAL_TIM_IRQHandler()`

```
stm32f1xx_it.c
229  * @brief This function handles TIM6 global interrupt.
230  */
231 void TIM6_IRQHandler(void)
232 {
233     /* USER CODE BEGIN TIM6_IRQn 0 */
234
235     /* USER CODE END TIM6_IRQn 0 */
236     HAL_TIM_IRQHandler(&htim6);
237     /* USER CODE BEGIN TIM6_IRQn 1 */
238
239     /* USER CODE END TIM6_IRQn 1 */
240 }
241
```

打开 `stm32f1xx_hal_tim.c` 文件，找到定时器中断处理函数原型 `HAL_TIM_IRQHandler()`，其主要作用就是判断是哪个定时器
数 `HAL_TIM_PeriodElapsedCallback()`。

```
stm32f1xx_hal_tim.c
5487 * @param htim TIM handle
5488 * @retval None
5489 */
5490 __weak void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
5491 {
5492     /* Prevent unused argument(s) compilation warning */
5493     UNUSED(htim);
5494
5495     /* NOTE : This function should not be modified, when the callback is needed,
5496              the HAL_TIM_PeriodElapsedCallback could be implemented in the user file
5497     */
5498 }
5499
```

```
/* NOTE: This function Should not be modified, when the callback is needed,
the HAL_GPIO_EXTI_Callback could be implemented in the user file
*/
```

这个函数不应该被改变，如果需要使用回调函数，请重新在用户文件中实现该函数。

`HAL_TIM_PeriodElapsedCallback()` 按照官方提示我们应该再次定义该函数，`__weak` 是一个弱化标识，带有这个的函数就是一模一样的函数，编译器就会忽略这一个函数，而去执行你写的那个函数；而 `UNUSED(htim)`，这就是一个防报错的定义，当传警告。其实我们在开发的时候已经不需要去理会中断服务函数了，只需要找到这个中断回调函数并将其重写即可而这个回调函数过数规整到一起并调用一个回调函数，也就是无论几个中断，我



Leung_ManWah

关注

18



77



3



专栏目录

接下来我们就在 `stm32f1xx_it.c` 这个文件的最下面添加 `HAL_TIM_PeriodElapsedCallback()`

```
/* USER CODE BEGIN 1 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static uint32_t time = 0;
    if(htim->Instance == TIM6) // 定时器6基地址
    {
        lv_tick_inc(1);    lv_tick_inc(1); //lvgl 的 1ms 心跳
    }
}
/* USER CODE END 1 */
```

添加定时器启动函数

现在进入 `main` 函数并在 `while` 循环前加入开启定时器函数 `HAL_TIM_Base_Start_IT()`，这里所传入的 `htim6` 就是刚刚定时器初

```
/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    ...
    ...
    /* USER CODE BEGIN 2 */
    HAL_TIM_Base_Start_IT(&htim6);
    /* USER CODE END 2 */
```



8.4 执行demo

实现 `btn_event_cb()` 按键事件回调，实现 `lvgl_first_demo_start()` 并在 `main()` 中调用。

```
/* Private user code -----*/
/* USER CODE BEGIN 0 */
static void btn_event_cb(lv_event_t * event)
{
    lv_obj_t *btn = lv_event_get_target(event); // 获得事件最初瞄准的对象。即使事件是冒泡的，也是
    if(event->code == LV_EVENT_CLICKED)
    {
        static uint8_t cnt = 0;
```



Leung_ManWah

关注

👍 18



🌟 77



💬 3

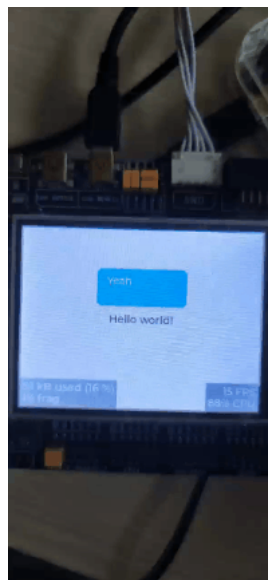


专栏目录

is the label and change its text*/



查看效果：



九、工程代码

链接：<https://pan.baidu.com/s/1mmcFnxAvYHJT-WxbNh3spQ?pwd=0htb> 提取码：0htb

十、注意事项



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

用户代码要加在 **USER CODE BEGIN N** 和 **USER CODE END N** 之间，否则下次使用 STM32CubeMX 重新生成代码后，会被删除。

```
/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */
    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */
    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    /* USER CODE BEGIN 2 */
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
        /* USER CODE END 3 */
    }
}
```

• 由 Leung 写于 2022 年 1 月 27 日

- 参考：[【LVGL学习之旅 01】移植LVGL到STM32](#)
[STM32移植LittleVgl\(LVGL\)嵌入式开源图形库](#)
[LittleVGL\(LVGL\) V8版本 干货入门教程一之移植到STM32并运行](#)
[野火指南者开发板移植 lvgl 库](#)

新知实验室——TRTC实时音视频产品体验官计划

训练营带你掌握音视频热门成熟解决方案，报名并参加新知实验室活动，挑战五万元奖金池，更有机会获得精品周边礼品.....



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录

“相关推荐”对你有帮助么？

😡 非常没帮助 😞 没帮助 😐 一般 😊 有帮助 😄 非常有帮助

[关于我们](#) [招贤纳士](#) [商务合作](#) [寻求报道](#) ☎ 400-660-0108 ✉ kefu@csdn.net 🗣 [在线客服](#) 工
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 [经营性网站备案信息](#) [北京互联网通](#)
[家长监护](#) [网络110报警服务](#) [中国互联网举报中心](#) [Chrome商店下载](#) [账号管理规范](#) [版权与免责声明](#) [版权申诉](#) 出
©1999-2022北京创新乐知网络技术有限公司



Leung_ManWah

关注

👍 18



🌟 77



💬 3



专栏目录