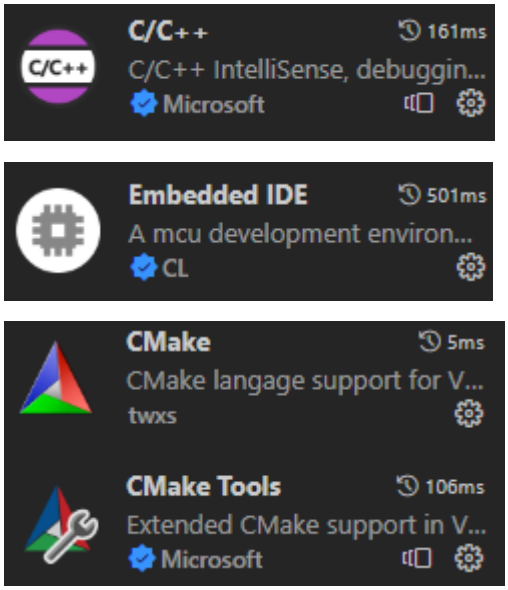
VSCode + LVGL 模拟&开发环境搭建

> 本笔记主要记录了环境搭建中一些关键步骤的说明，但内容可能还有遗漏，配合视频一起学习效果更佳

VSCode插件

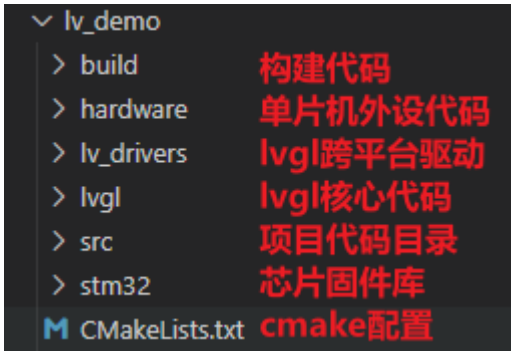


所需资源获取链接

> cmake 获取链接(注意选取windows_x64.zip版本):
> https://cmake.org/download/
>
> lvgl 获取链接(注意选取版本tag):
> https://github.com/lvgl/lvgl
>
> lv_drivers 获取链接(注意选取版本tag):
> https://github.com/lvgl/lv_drivers
>
> sdl2 获取链接(注意选取mingw.zip版本):
> https://github.com/libsdl-org/SDL/releases/
>
> mingw获取链接(可以选择win32-seh-msvcrt-rt_v10-rev2版本):
> https://github.com/niXman/mingw-builds-binaries/releases

项目架构搭建

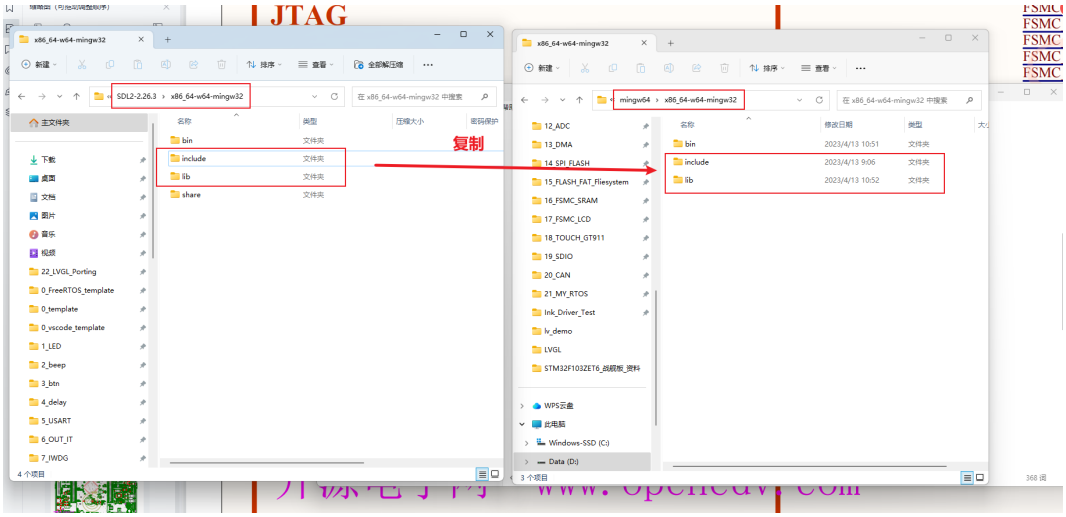
- **项目大致结构图示**



- **安装SDL2库**

  1. 将 {SDL2源码目录}\x86_64-w64-mingw32下的 include 和 lib 目录

     复制到 {mingw目录}\x86_64-w64-mingw32下的 include 和 lib 目录



  2. 将 {SDL2源码目录}\x86_64-w64-mingw32\bin下的 SDL2.dll文件拿出来

     等项目创建后，放到项目根目录下，回头window模拟需要使用到

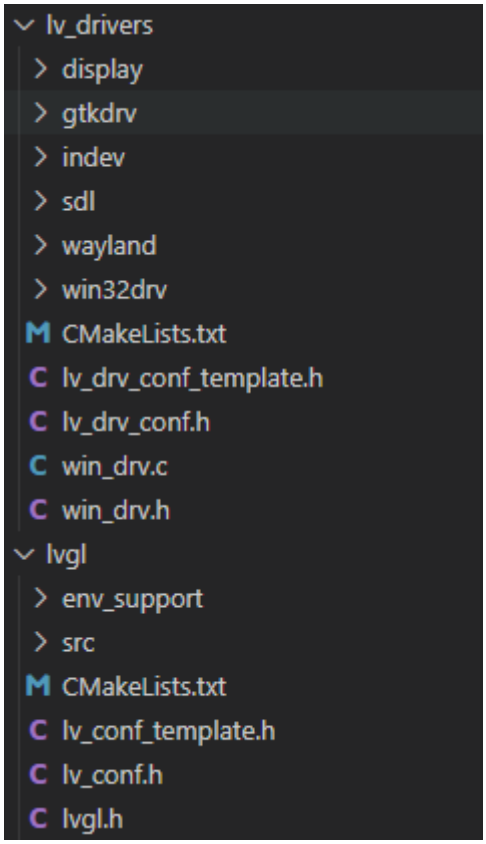
- **用EIDE创建单片机项目，移植MCU固件库，配置编译资源和Arm编译工具**

  EIDE是用来做单片机项目开发、编译、烧入、调试的插件，具体使用方法可见教程：

  https://www.bilibili.com/video/BV1Rm4y1B73J/

- **移植 lvgl 源码 和 lv_drivers**

  1. 将下载好的 lvgl、lv_drivers 复制到项目
  2. 为项目复制添加 lv_conf.h、lv_drv_conf 两个配置文件，如何添加建官方文档或视频教程
  3. 可以移除lvgl中大部分无用的文件和目录如 .gitignore、.md、.json、LICENSED、doc 等，结果如下:



注意： 如果移除了 lvgl 源码下的 demo 和 example 示例代码，

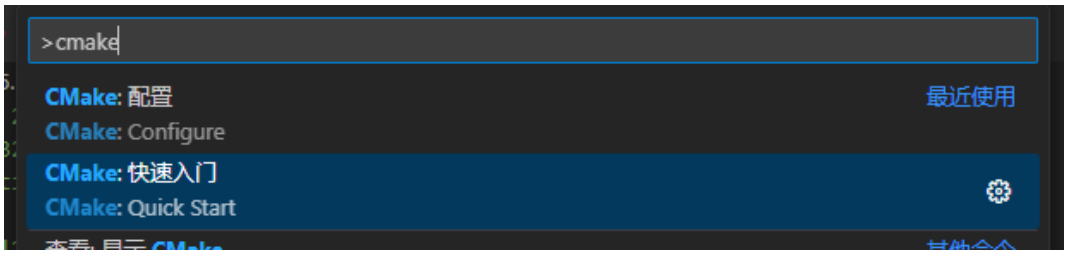
需要在 /lvgl/env_support/cmake/custom.cmake 文件内注释掉对应的编译、包含配置

```
file(GLOB RECURSE SOURCES ${LVGL_ROOT_DIR}/src/*.c)
# file(GLOB_RECURSE EXAMPLE_SOURCES ${LVGL_ROOT_DIR}/examples/*.c)
# file(GLOB_RECURSE DEMO_SOURCES ${LVGL_ROOT_DIR}/demos/*.c)

if (BUILD_SHARED_LIBS)
  add_library(lvgl SHARED ${SOURCES})
else()
  add_library(lvgl STATIC ${SOURCES})
endif()

add_library(lvgl::lvgl ALIAS lvgl)
# add_library(lvgl_examples STATIC ${EXAMPLE_SOURCES})
# add_library(lvgl::examples ALIAS lvgl_examples)
# add_library(lvgl_demos STATIC ${DEMO_SOURCES})
# add_library(lvgl::demos ALIAS lvgl_demos)

target_compile_definitions(
  lvgl PUBLIC $<$<BOOL:${LV_LVGL_H_INCLUDE_SIMPLE}>:LV_LVGL_H_INCLUDE_SIMPLE>
              $<$<BOOL:${LV_CONF_INCLUDE_SIMPLE}>:LV_CONF_INCLUDE_SIMPLE>)

# Include root and optional parent path of LV_CONF_PATH
target_include_directories(lvgl SYSTEM PUBLIC ${LVGL_ROOT_DIR} ${LV_CONF_DIR})

# Include /examples folder
# target_include_directories(lvgl_examples SYSTEM
#                            PUBLIC ${LVGL_ROOT_DIR}/examples)
# target_include_directories(lvgl_demos SYSTEM
#                            PUBLIC ${LVGL_ROOT_DIR}/demos)

# target_link_libraries(lvgl_examples PUBLIC lvgl)
# target_link_libraries(lvgl_demos PUBLIC lvgl)

```

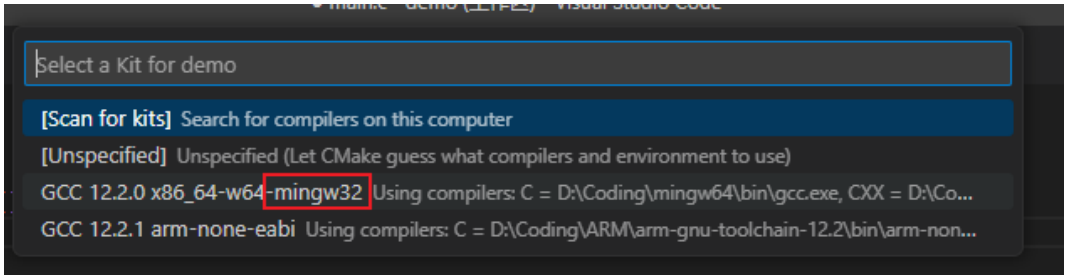
- **使用Cmake插件为创建好的项目配置window编译环境**

通过cmake + mingw + sdl2 实现 widnows 下的 LVGL 模拟

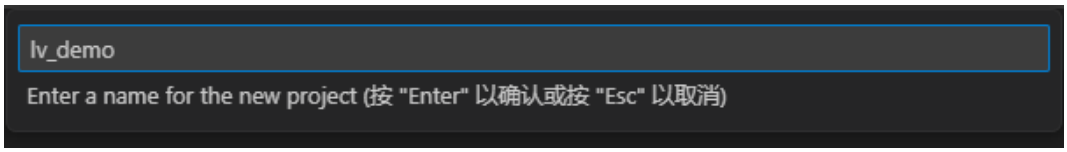
1. Ctrl + Shift + P 输入 cmake, 选择 Quick Start 为项目快速配置 cmake



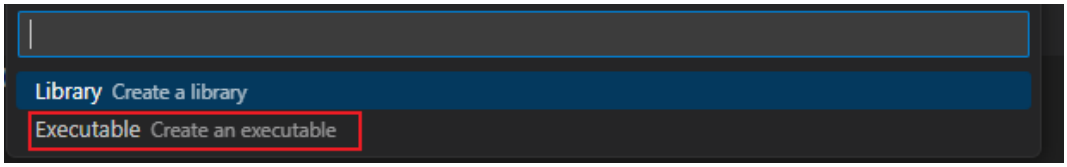
2. 选择 mingw 作为编译工具



3. 为cmake项目取个名字



4. 选择要生成目标 Library库文件，Excuatable可执行文件



5. 为生成好的 CMakeLists.txt 文件添加配置，配置见 《CMakeLists.txt 文件模板》

- **编写 lvgl 移植相关驱动**

> 本教程中提供了一个根据官方代码简略后的window驱动代码，以及一个正点原子STM32开发板驱动代码案例，详见《LVGL移植 - 模拟驱动代码案例》
>
> 各位小伙伴也可以根据自己的芯片、外设开发自己的驱动

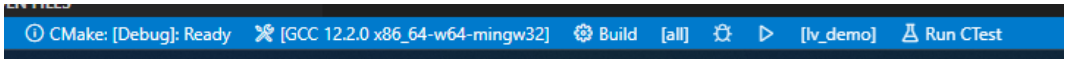
- **配置SDL2模拟窗口分辨率**

> 在lv_driver_conf.h中可以修改SDL串口分辨率

```
96
97  #if USE_SDL || USE_SDL_GPU
98  #  define SDL_HOR_RES     480
99  #  define SDL_VER_RES     272
100
```

- **使用cmake tools一键编译、模拟LVGL项目**

  在完成以上设置后可以在vscode左下角工具栏中进行window环境的编译、运行、debug操作



- **配置 自定义宏 MCU_ENV 切换 Window 和 MCU 编译环境**

  参考《LVGL移植 - 模拟驱动代码案例》

  在代码中有一行 **#ifdef MCU_ENV** 代码，根据是否有 **MCU_ENV** 宏定义而切换要编译的代码

  我们可以在EIDE插件中单独申明 **MCU_ENV** 宏，如下：

  这样使用 cmake 编译的程序会调用window驱动，反之用 EIDE 编译的代码会调用自己实现的外设驱动

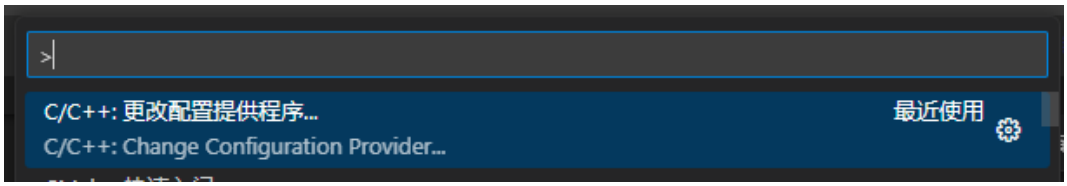
- **更改配置提供程序！！**

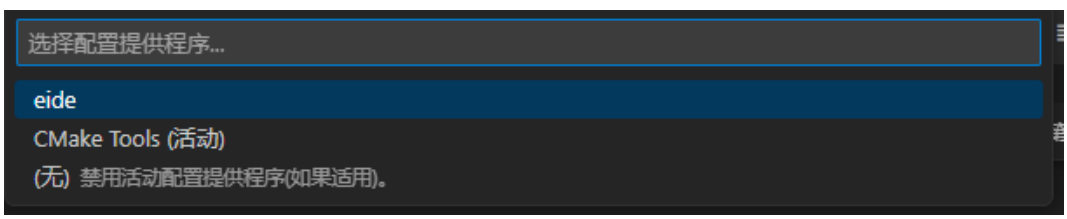
  由于 EIDE 和 Cmake插件配置中所引入的 源代码、头文件、宏，都有所差异。

  而 C/C++插件 会根据这些配置为我们提供对应的代码提示、补全、跳转等功能。

  所以我们需要主动告知VSCode我们目前需要使用哪个插件的配置：

  使用 Ctrl + Shift + P 快捷键， 输入 ConfigurationProvier 调出更改配置选项框，选择对应配置





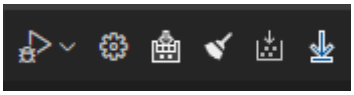
- **使用 EIDE插件 为MCU进行 编译、烧入、DUBUG**

  与Window模拟环境不同，MCU代码不需要引入lv_drivers, SDL2库 和 window驱动移植等代码

  我们可以使用EIDE的配置菜单，单独为MCU引入需要编译的模块(固件库、外设驱动)、头文件、宏定义等

并可以用，EIDE自带的按钮实现MCU编译、烧入、调试功能。

从而实现一套代码，两套编译、调试环境



常见错误

- mcu编译没找到 **lv_conf.h** 配置文件， 添加 **lv_conf.h** 并 设置 **LV_CONF_INCLUDE_SIMPLE** 宏

```
>> [  6%] CC 'lvgl/src/core/lv_obj_tree.c'
".\lvgl\src\core\../lv_conf_internal.h", line 41: Error:  #5: cannot open source input file "../../lv_conf.h": No such file or directory
         #include "../../lv_conf.h"              /*Else assume lv_conf.h is next to the lvgl folder*/
```

- MCU 链接时没找到 **__aeabi_assert**， 添加 **NDEBUG** 宏

```
Error: L6218E: Undefined symbol __aeabi_assert (referred from qrcodegen.o).
```

- No space in xxxxxx，MCU **RAM** 或 **FLASH** 不足。是否忘记配置 **RAM**、**ROM** ？

  如果使用AC5编译器，可以再它自带的可视化界面中修改RAM、ROM配置

  如果使用GCC交叉编译器，则需要修改 ld 链接脚本

  如果是真的 **RAM** 或 **FLASH** 不够就只能换芯片或采取其他措施了

```
Error: L6406E: No space in execution regions with .ANY selector matching lv_label.o(i.lv_anim_set_exec_cb).
Error: L6406E: No space in execution regions with .ANY selector matching lv_label.o(i.lv_anim_set_playback_time).
Error: L6406E: No space in execution regions with .ANY selector matching lv_label.o(i.lv_anim_set_time).
Error: L6406E: No space in execution regions with .ANY selector matching lv_label.o(i.lv_anim_set_var).
Error: L6406E: No space in execution regions with .ANY selector matching lv_textarea.o(i.lv_anim_set_exec_cb).
Error: L6406E: No space in execution regions with .ANY selector matching lv_textarea.o(i.lv_anim_set_path_cb).
Error: L6406E: No space in execution regions with .ANY selector matching lv_textarea.o(i.lv_anim_set_time).
Error: L6406E: No space in execution regions with .ANY selector matching lv_textarea.o(i.lv_anim_set_var).
Error: L6406E: No space in execution regions with .ANY selector matching lv_textarea.o(.data).
Error: L6406E: No space in execution regions with .ANY selector matching stm32f10x_gpio.o(i.GPIO_ResetBits).
Error: L6406E: No space in execution regions with .ANY selector matching stm32f10x_gpio.o(i.GPIO_SetBits).
Error: L6406E: No space in execution regions with .ANY selector matching lv_refr.o(i._lv_refr_init).
Error: L6406E: No space in execution regions with .ANY selector matching lv_draw.o(i.lv_draw_init).
Error: L6406E: No space in execution regions with .ANY selector matching lv_img_cache.o(i.lv_img_cache_invalidate_src).
Error: L6406E: No space in execution regions with .ANY selector matching lv_draw_sw.o(i.lv_draw_sw_wait_for_finish).
```

- 有种情况是用该流程搭建环境后，window模拟成功，但MCU下载后不显示图像。

  可能是 栈 或 堆 空间分配的不够，导致内存溢出。还是去改配置就好。

  栈建议最小 2K 即 0x0800

  堆空间自定义小分配一点都没事, 在lv_config中可以配置 lvgl 堆缓存大小，然后用lv_mem_alloc获取

CMakeLists.txt 文件模板

```
cmake_minimum_required(VERSION 3.0.0)
project(lv_demo VERSION 0.1.0) #项目名(可改)

include(CTest)
enable_testing()

# 添加宏定义
add_definitions(-DUSE_SDL)

# 添加头文件目录
include_directories(src)

# 添加子库编译
add_subdirectory("lvgl")
add_subdirectory("lv_drivers")

# 添加源文件(递归包含)
FILE(GLOB_RECURSE src_source ./src/*.c)

# 编译可执行文件
add_executable(${PROJECT_NAME} ${src_source})

# 链接库
target_link_libraries(${PROJECT_NAME} lvgl)
target_link_libraries(${PROJECT_NAME} lv_drivers)
target_link_libraries(${PROJECT_NAME} mingw32)
target_link_libraries(${PROJECT_NAME} SDL2main)
target_link_libraries(${PROJECT_NAME} SDL2)

# 拷贝SDL2.dll 文件到 build目录
file(COPY SDL2.dll DESTINATION ../build)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

LVGL移植 - 模拟驱动代码案例

```c
/** lv_porting.h ***/
#ifndef __LV_PORTING_H__
#define __LV_PORTING_H__

void lv_porting_init(void);

void lv_porting_delay(void);

#endif
```

```c
/** lv_porting.c ***/
#include "lv_porting.h"
#include "lvgl.h"

#define MONITOR_HOR_RES 480
#define MONITOR_VER_RES 272
#define MONITOR_BUF_SIZE MONITOR_HOR_RES * 10


#ifdef MCU_ENV

#include "gt911.h"
#include "lcd.h"
#include "timer.h"
#include "systick.h"

void disp_drv_init(void);
void indev_drv_init(void);

void lv_porting_init(void)
{
    lv_init();
    disp_drv_init();          // 显示设备驱动初始化
    indev_drv_init();         // 输入设备驱动初始化
    timer2_it_init(7199, 49); // 时钟中断初始化
}

void TIM2_IRQHandler(void)
{
    lv_tick_inc(5);
    timer2_clearFlag();
}

void disp_flush(lv_disp_drv_t *disp_drv, const lv_area_t *area, lv_color_t
*color_p)
{
    LCD_draw_area(area->x1, area->y1, area->x2, area->y2, (uint16_t *)color_p);
    lv_disp_flush_ready(disp_drv);
}

// 显示设备驱动初始化
void disp_drv_init(void)
{

    lcd_init();
    LCD_WriteReg(0x3600, 0x60); // 横屏
```

```
    // 初始化图像缓冲区，第二个缓冲区(可选)可以传入NULL
    static lv_disp_draw_buf_t disp_buf;
    static lv_color_t buf_1[MONITOR_BUF_SIZE];
    static lv_color_t buf_2[MONITOR_BUF_SIZE];
    lv_disp_draw_buf_init(&disp_buf, buf_1, buf_2, MONITOR_BUF_SIZE);

    // 初始化显示驱动
    static lv_disp_drv_t disp_drv;
    lv_disp_drv_init(&disp_drv);
    disp_drv.hor_res  = MONITOR_HOR_RES;
    disp_drv.ver_res  = MONITOR_VER_RES;
    disp_drv.draw_buf = &disp_buf;
    disp_drv.flush_cb = disp_flush;
    lv_disp_drv_register(&disp_drv);
}

void indev_read(struct _lv_indev_drv_t *indev_drv, lv_indev_data_t *data)
{

    GT911_touch_point *touchPoints = gt911_scan();
    if (touchPoints != NULL) {
        // 横屏坐标转换
        data->point.x = touchPoints->y;
        data->point.y = MONITOR_HOR_RES - touchPoints->x - 1;
        data->state   = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}

// 输入设备驱动初始化
void indev_drv_init(void)
{
    // 初始化触摸屏
    gt911_init();

    static lv_indev_drv_t indev_drv;
    lv_indev_drv_init(&indev_drv);
    indev_drv.type    = LV_INDEV_TYPE_POINTER;
    indev_drv.read_cb = indev_read;
    lv_indev_drv_register(&indev_drv);
}

void inline lv_porting_delay(void)
{
    delay_ms(5);
}

#else

#define _DEFAULT_SOURCE  /* needed for usleep() */
#define SDL_MAIN_HANDLED /*To fix SDL's "undefined reference to WinMain" issue*/
#include <SDL2/SDL.h>
#include "sdl/sdl.h"

static int tick_thread(void *data)
{
```

```
    LV_UNUSED(data);

    while (1) {
        SDL_Delay(5);
        lv_tick_inc(5); /*Tell LittelvGL that 5 milliseconds were elapsed*/
    }

    return 0;
}

void lv_porting_init(void)
{
    lv_init();
    monitor_init();
    // SDL创建线程
    SDL_CreateThread(tick_thread, "tick", NULL);

    // 初始化图像缓冲区，第二个缓冲区(可选)可以传入NULL
    static lv_disp_draw_buf_t disp_buf;
    static lv_color_t buf_1[MONITOR_BUF_SIZE];
    static lv_color_t buf_2[MONITOR_BUF_SIZE];
    lv_disp_draw_buf_init(&disp_buf, buf_1, buf_2, MONITOR_BUF_SIZE);

    /* 注册显示驱动 */
    static lv_disp_drv_t disp_drv;
    lv_disp_drv_init(&disp_drv);
    disp_drv.draw_buf     = &disp_buf;
    disp_drv.flush_cb     = monitor_flush;
    disp_drv.hor_res      = MONITOR_HOR_RES;
    disp_drv.ver_res      = MONITOR_VER_RES;
    disp_drv.antialiasing = 1;
    lv_disp_t *disp       = lv_disp_drv_register(&disp_drv);

    /* 注册鼠标驱动 */
    static lv_indev_drv_t indev_drv_1;
    lv_indev_drv_init(&indev_drv_1);
    indev_drv_1.type        = LV_INDEV_TYPE_POINTER;
    indev_drv_1.read_cb     = sdl_mouse_read;
    lv_indev_t *mouse_indev = lv_indev_drv_register(&indev_drv_1);
}

void inline lv_porting_delay(void)
{
    SDL_Delay(5);
}

#endif
```

更改配置提供程序,多说一遍！！

> 由于 EIDE 和 Cmake 插件配置中所引入的 源代码、头文件、宏，都有所差异。
>
> 而 C/C++插件 会根据这些配置为我们提供对应的代码提示、补全、跳转等功能。
>
> 所以我们需要主动告知VSCode我们目前需要使用哪个插件的配置：

使用 Ctrl + Shift + P 快捷键， 输入 ConfigurationProvier 调出更改配置选项框，选择对应配置