Creating a Button

Detailed Tutorial

In this section, we will learn how to use buttons. In this section, we will create a button, and when the button is pressed by the user, we will output Hello World once in the serial terminal.

Let's create a new code first, then copy the basic structure given in the first section and paste it into the new code.

Above the def main() function, declare a function create_button() to create a button, and call this function after the main function initializes lvgl

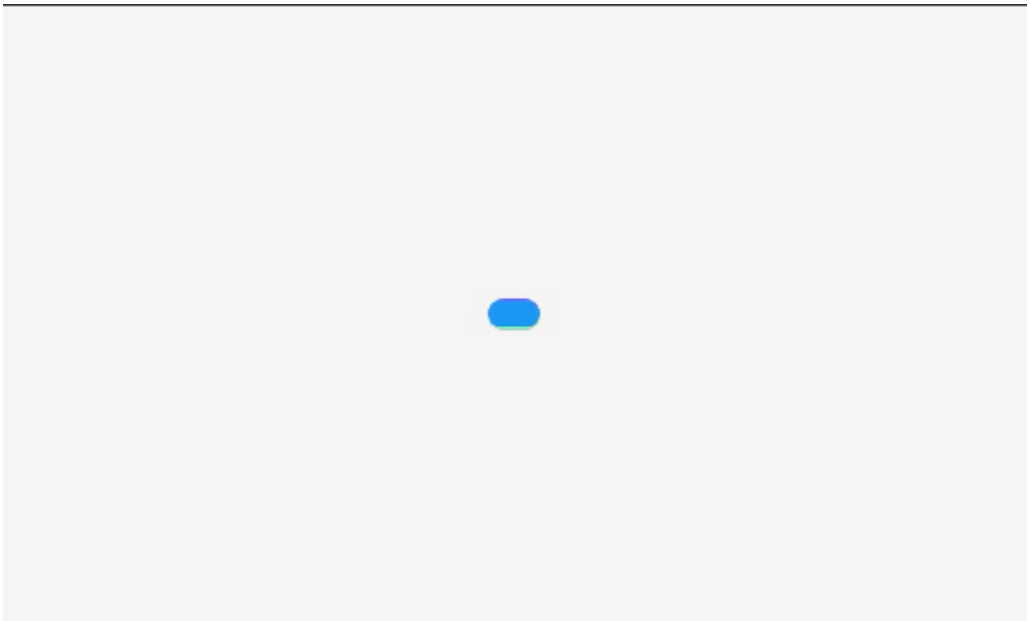
```
def create_button():
    # 创建按钮组件 / create a button

def main():
    """
    主函数
    Main function
    """
    try:
        # 初始化显示设备和LVGL / Initialize display device and LVGL
        display_init()
        lvgl_init()
        create_button() # 调用创建按钮组件的函数 / create button
        print("LVGL initialization completed")
```

Back to create_button(), we create a button in the middle of the screen and center it

```
def create_button():
    # 创建按钮组件 / create a button
    btn = lv.btn(lv.scr_act())
    btn.align(lv.ALIGN.CENTER,0,0)
```

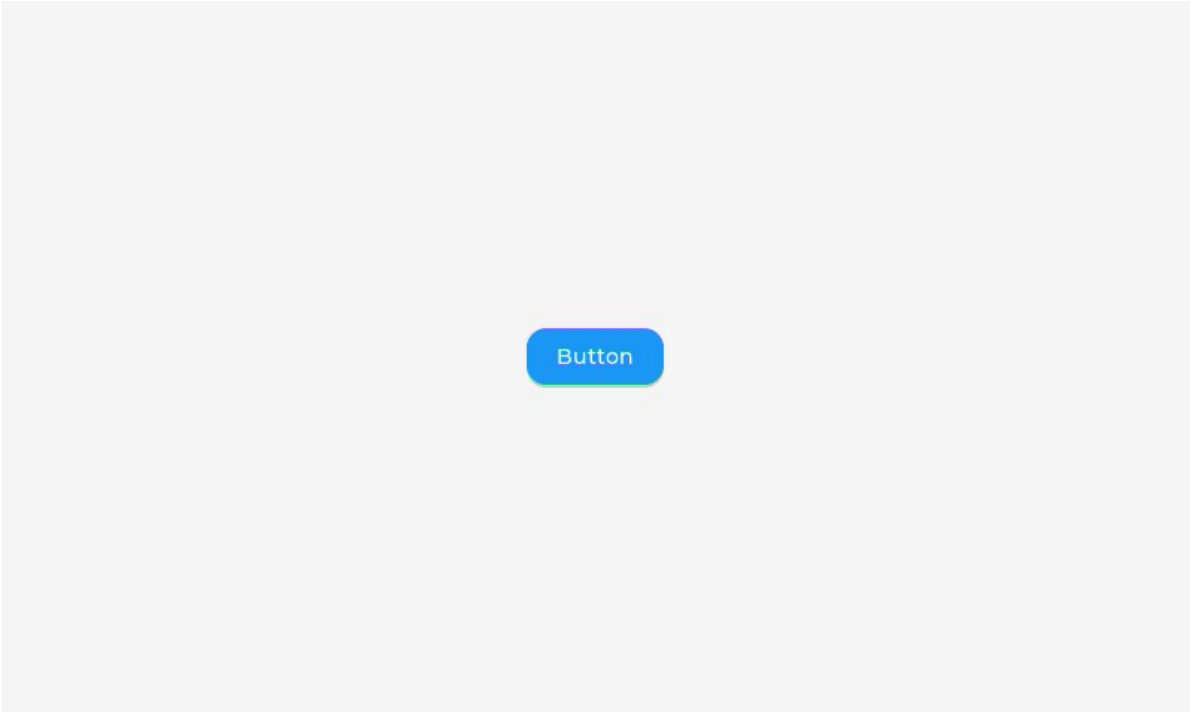
> The basic syntax for creating a component: new_obj = lv.widget(father_obj)
>
> Among them: new_obj is the name of the new component, widget is the component type, here it is btn (button), father_obj is the parent component
>
> lv.scr_act() returns the current screen component

The effect is as shown below:



Next, we add a text component (or label component) label to the button.

```
def create_button():
    # 创建按钮组件 / create a button
    btn = lv.btn(lv.scr_act())
    btn.align(lv.ALIGN.CENTER,0,0)
    label=lv.label(btn)
    label.set_text("Button")
```



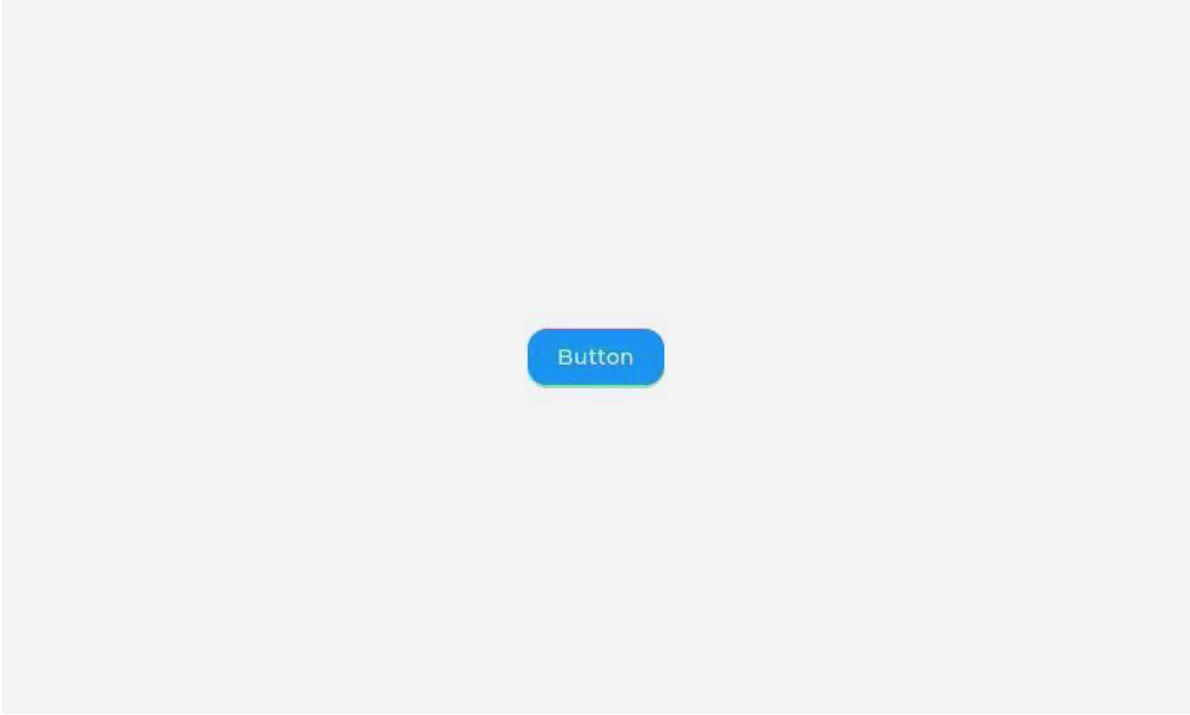
Now we have a button with a label, but if we try to click on it, we will find that there is no click effect. So we need to introduce the [TOUCH] touch module

```
# 导入部分 Import section
from machine import TOUCH
```

```
# 声明一个touch_screen类 Declare a touch_screen class
class touch_screen():
    def __init__(self):
        self.state = lv.INDEV_STATE.RELEASED

        self.indev_drv = lv.indev_create()
        self.indev_drv.set_type(lv.INDEV_TYPE.POINTER)
        self.indev_drv.set_read_cb(self.callback)
        self.touch = TOUCH(0)

    def callback(self, driver, data):
        x, y, state = 0, 0, lv.INDEV_STATE.RELEASED
        tp = self.touch.read(1)
        if len(tp):
            x, y, event = tp[0].x, tp[0].y, tp[0].event
            if event == 2 or event == 3:
                state = lv.INDEV_STATE.PRESSED
        data.point = lv.point_t({'x': x, 'y': y})
        data.state = state

def lvgl_init():
    # ......................
    # lvgl_init方法前面的代码不变，在此省略。在lvgl_init()方法的最后加上一行tp =
touch_screen()
    # The code before the lvgl_init method remains unchanged and is omitted here.
Add a line tp = touch_screen() at the end of the lvgl_init() method
    tp = touch_screen()
```

This part is also fixed on our K230. You don't need to study the principle in depth. Just copy it when you need to use it. (Students who want to learn the principle can refer to the tutorial of TOUCH touch section in the basic chapter and search for lvgl how to bind touch)

After completing these steps, we click the Run button again and try to click the button on the screen. You can see that the click effect has appeared.



Finally, we add a click event to the button so that we can trigger the code we want to trigger when the button is pressed.

We declare a callback function that will be triggered when a button is pressed and bind it to our button

```
def on_click(event):
    print("I was clicked!")

def create_button():
    # 创建按钮组件 / create a button
    btn = lv.btn(lv.scr_act())
    btn.align(lv.ALIGN.CENTER,0,0)
    label=lv.label(btn)
    label.set_text("Button")
    btn.add_event(on_click, lv.EVENT.CLICKED, None)
```

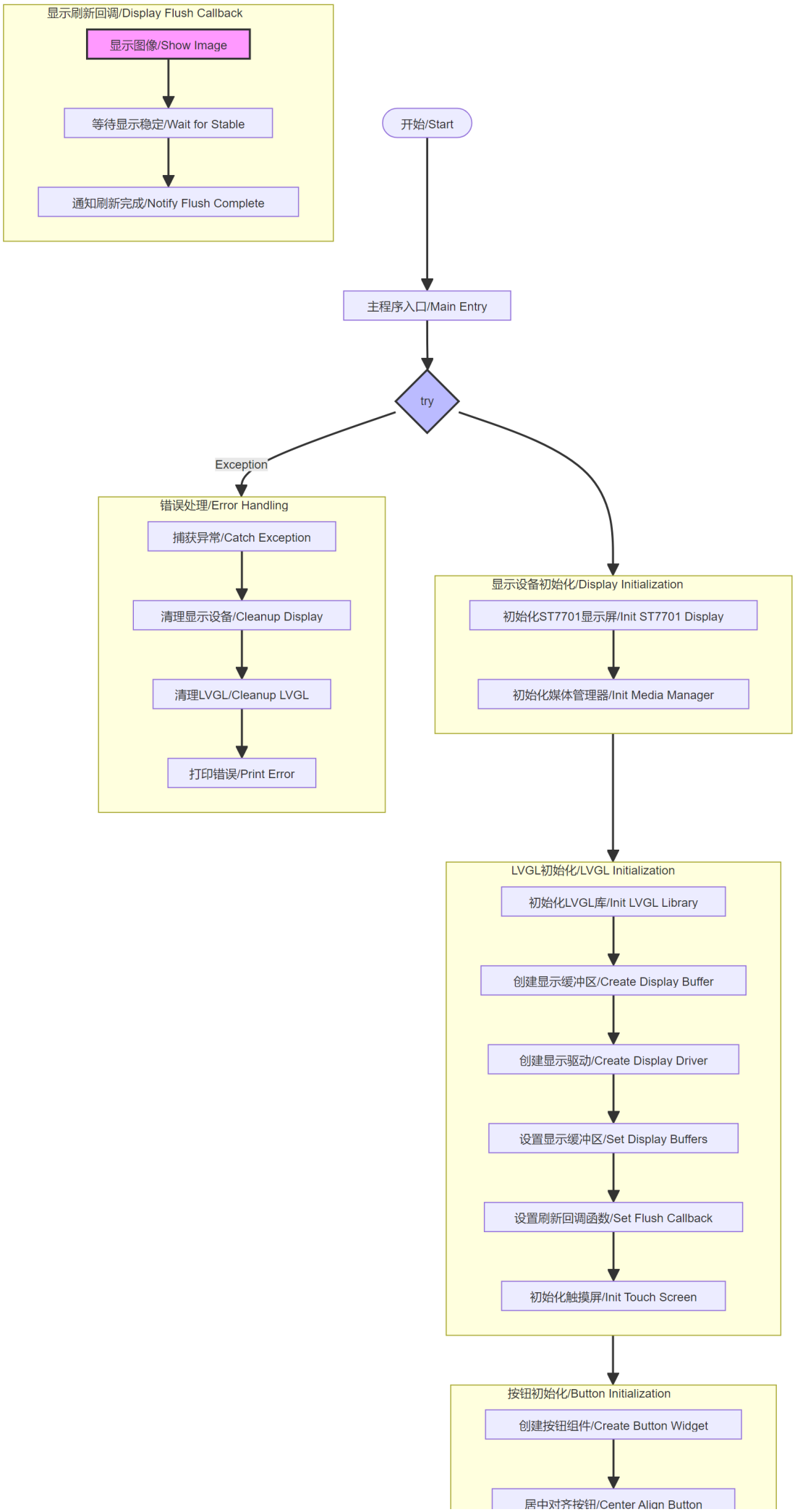
After adding, we rerun the code, then click the button on the screen and observe the serial terminal output:

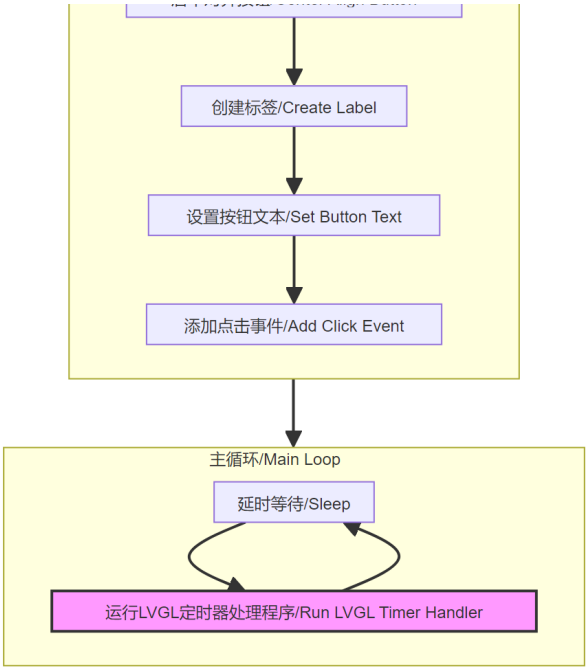
```
buffer pool :  3
LVGL initialization completed
I was clicked!
I was clicked!
I was clicked!
I was clicked!
I was clicked!
```

Through the above steps, we can successfully create a lvgl program with a button

Complete code

For the complete code, please refer to the file [Source Code Summary / 12.Lvgl / 02.lvgl_button.py]


显示刷新回调/Display Flush Callback
显示图像/Show Image
等待显示稳定/Wait for Stable
通知刷新完成/Notify Flush Complete
开始/Start
主程序入口/Main Entry
try
Exception
错误处理/Error Handling
捕获异常/Catch Exception
清理显示设备/Cleanup Display
清理LVGL/Cleanup LVGL
打印错误/Print Error
显示设备初始化/Display Initialization
初始化ST7701显示屏/Init ST7701 Display
初始化媒体管理器/Init Media Manager
LVGL初始化/LVGL Initialization
初始化LVGL库/Init LVGL Library
创建显示缓冲区/Create Display Buffer
创建显示驱动/Create Display Driver
设置显示缓冲区/Set Display Buffers
设置刷新回调函数/Set Flush Callback
初始化触摸屏/Init Touch Screen
按钮初始化/Button Initialization
创建按钮组件/Create Button Widget
居中对齐按钮/Center Align Button


创建标签/Create Label
设置按钮文本/Set Button Text
添加点击事件/Add Click Event
主循环/Main Loop
延时等待/Sleep
运行LVGL定时器处理程序/Run LVGL Timer Handler