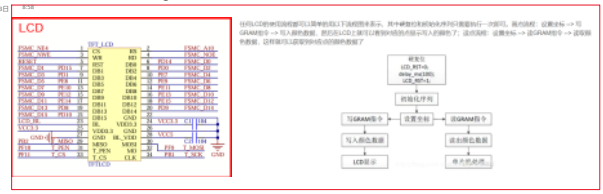
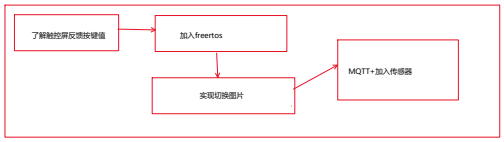
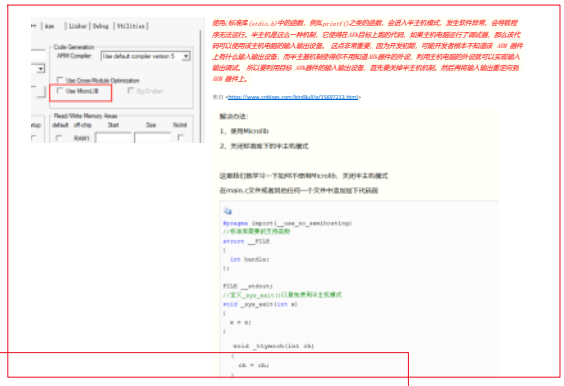
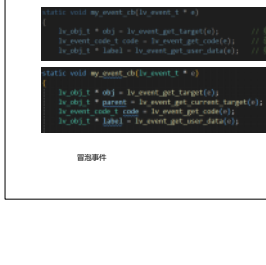
LVGL+MQTT+Smarthome Thinking

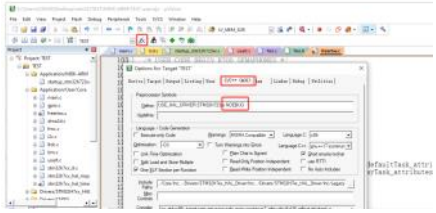
2024年8月3日

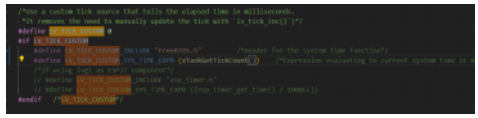


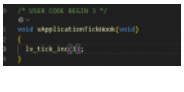


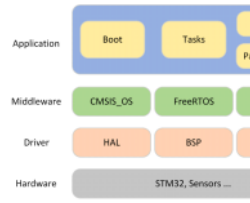


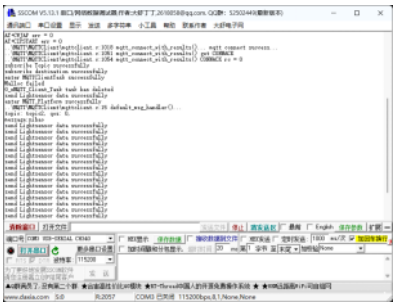


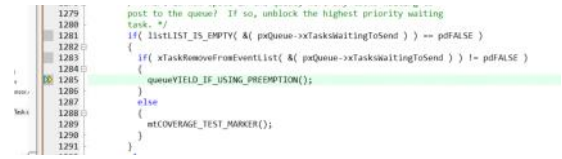


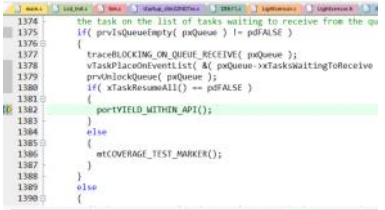


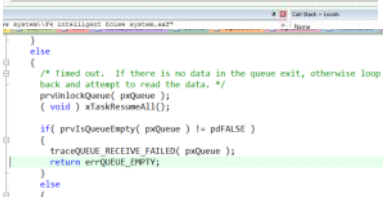


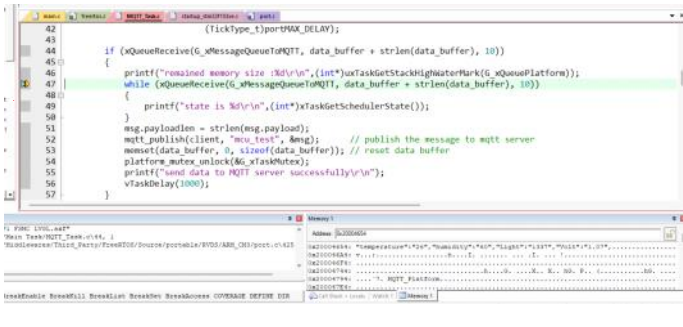


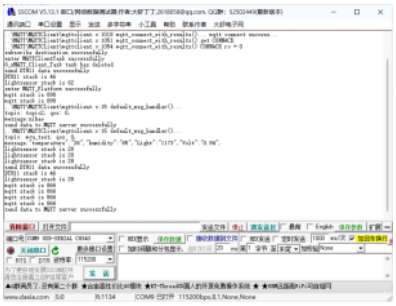


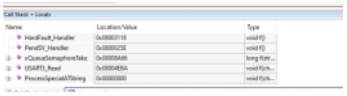


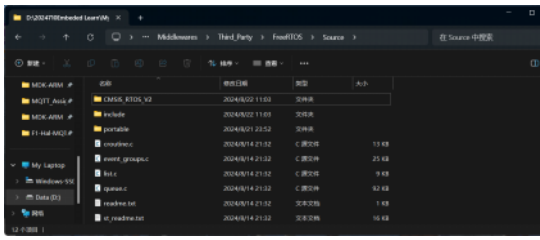


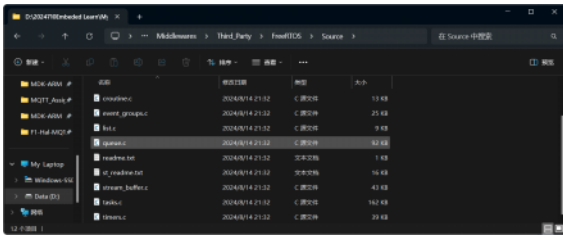


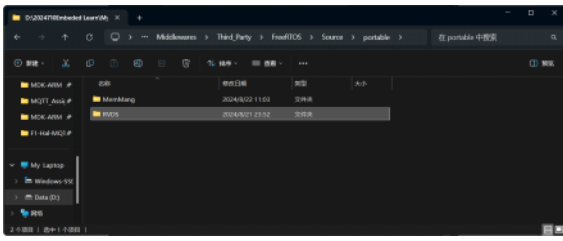












使用这个函数要注意以下问题：

1. 此函数切不可在中断服务程序中调用。

2. 这里再着重说明下这个函数的返回值，通过返回值用户可以检测是哪个事件标志位被置 1 了。

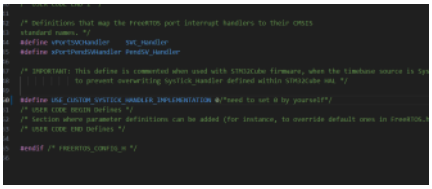
□ 如果由于设置的等待时间超时，函数的返回值可能会有部分事件标志位被置 1。

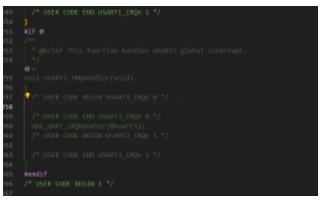
□ 如果由于指定的事件标志位被置1而返回， 并且设置了这个函数的参数xClearOnExit为pdTRUE，那么此函数的返回值是清零前的事件标志组数值。

另外，调用此函数的任务在离开阻塞状态到退出函数 xEventGroupWaitBits 之间这段时间，如果一个

高优先级的任务抢占执行了，并且修改了事件标志位，那么此函数的返回值会跟当前的事件标志组数值不同 。

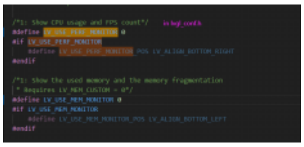
实验分析插：



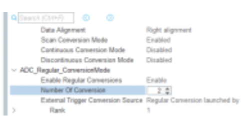


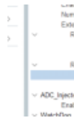
```
    ( void ) xTaskResumeAll();

    if( prvIsQueueEmpty( pxQueue ) != pdFALSE )
    {
      traceQUEUE_RECEIVE_FAILED( pxQueue );
      return errQUEUE_EMPTY;
    }
    else
    {
      mtCOVERAGE_TEST_MARKER();
    }
```


Size: Code=199480 RO-data=60132 RW-data=984 ZI-data=59872
creating hex file...
LVGL\F1 FSMC LVGL.axf" - 0 Error(s), 4 Warning(s).

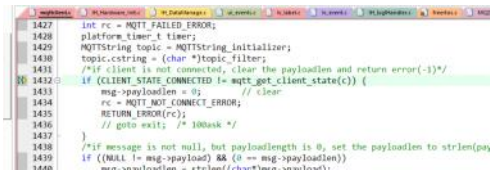

/*1: Show CPU usage and FPS count*/
#define LV_USE_PERF_MONITOR 0
#if LV_USE_PERF_MONITOR
#define LV_USE_PERF_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
#endif
/*1: Show the used memory and the memory fragmentation
* Requires LV_MEM_CUSTOM = 0*/
#define LV_USE_MEM_MONITOR 0
#if LV_USE_MEM_MONITOR
#define LV_USE_MEM_MONITOR_POS LV_ALIGN_BOTTOM_LEFT
#endif

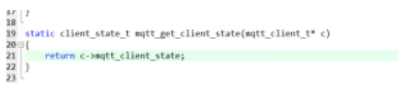
adc


Search (Ctrl+F)
Data Alignment
Right alignment
Scan Conversion Mode
Enabled
Continuous Conversion Mode
Disabled
Discontinuous Conversion Mode
Disabled
ADC_Regular_ConversionMode
Enable Regular Conversions
Enable
Number Of Conversion
2
External Trigger Conversion Source
Regular Conversion launched by
Rank
1




static int mqtt_yield(mqtt_client_t* c, int timeout_ms)


int rc = MQTT_FAILED_ERROR;
platform_timer_t timer;
MQTTString topic = MQTTString_initializer;
topic.cstring = (char *)topic_filter;
/*if client is not connected, clear the payloadlen and return error(-1)*/
if (CLIENT_STATE_CONNECTED != mqtt_get_client_state(c)) {
msg->payloadlen = 0; // clear
rc = MQTT_NOT_CONNECT_ERROR;
RETURN_ERROR(rc);
// goto exit; /* 100ask */
}
/*if message is not null, but payloadlength is 0, set the payloadlen to strlen(pay
if ((NULL != msg->payload) && (0 == msg->payloadlen))


static client_state_t mqtt_get_client_state(mqtt_client_t* c)
{
return c->mqtt_client_state;
}


ber Of Conversion 2
rnal Trigger Conversion Source Regular Conversion launched by software
ank 1
Channel Channel 1
Sampling Time 1.5 Cycles
ank 2
Channel
Sampling Time 1.5 Cycles
ed_ConversionMode
le Injected Conversions Disable

```
  MX_USART3_UART_Init();
  MX_I2C1_Init();
  MX_TIM3_Init();
  MX_USART3_UART_Init();
  MX_ADC1_Init();
  MX_RTC_Init();
  /* USER CODE BEGIN 2 */
  //HardwareInitTask(NULL);
  /* USER CODE END 2 */

  /* Init scheduler */
  osKernelInitialize();

  /* Call init function for freertos objects (in cmsis
  MX_FREERTOS_Init();

  /* Start scheduler */
```

因为模块上电不稳定，会乱发一些数据过来，然后我提前开启了中断，这时候我在自己写uart3handler里提前释放了信号量，导致rtos卡死在队列获取中断函数里

```
/* add threads, ... */
HardwareInitTaskHandle = osThreadNew(HardwareInitTask, NULL, &HardwareInitTask_attributes);
```

```
void HardwareInitTask(void *argument)
{
  while (1)
  {
    /*********************UART INITIATION*********************/
    Uart3_Lock_Init();
    AT_Init();
    // Delay_Init();
    vTaskSuspendAll();
    /*********************DHT11 INITIATION*********************/
    DHT11_Start();
    DHT11_Wait_Ack();
    /*********************LightSensor INITIATION*********************/
    LightSensor_Init();
    /*********************LVGL INITIATION*********************/
    lv_init();
    lv_port_disp_init();
    lv_port_indev_init();
    xTaskResumeAll();
    /*********************                INITIATION*********************/
    printf("Hello World!\r\n");
    /*********************TIM INITIATION*********************/
    // HAL_TIM_Base_Start_IT(&htim1);
    vTaskDelete(NULL);
    vTaskDelay(500);
  }
}
```

```
21  if(((isrflags&USART_SR_RXNE)!=RESET)&&((cr1its&USART_CR1_RXNEIE)!=RESET))
22  {
23    // c = (char)HAL_UART_Receive_IT(&huart3, (uint8_t *)&c, 1);
24    c=huart3.Instance->DR;
25    ring_buffer_write(c, &uart3_buffer);
26    platform_mutex_unlock_fromisr(&uart_recv_mutex);
27    return;
28  }
29 }
30
31 void USART3_Write(char *buf,int len)
32 {
33   int i=0;
34   while(i<len)
35   {
36     while((huart3.Instance->SR & USART_SR_TXE)==0);
37     huart3.Instance->DR = buf[i];
```

关于FreeRTOS函数xSemaphoreGiveFromISR卡死的问题_xsemaphoregivefromisr导致卡死-CSDN博客

在一些嵌入式系统中，特别是在启动过程中，各种外设和系统模块可能需要一些时间来初始化和稳定。这个稳定的时间可能与硬件设计、外设配置、时钟设置等因素有关。延迟一段时间可以确保系统已经完全初始化并进入稳定状态。这样在启用 UART 空闲中断之前，就可以避免因为系统处于不稳定状态而触发意外的中断。刚刚那个flag的例子就很好的说明了，有很多种情况在不稳定的时候进了中断，所以需要延迟！