导航
博客园
首页
新随笔
联系
订阅 XML
管理

公告
渡我不渡她 -
Not available
00:00 / 03:41
1 渡我不渡她
2 小镇姑娘
3 PDD洪荒之力
加入QQ群
昵称： 杨奉武
园龄： 5年10个月
粉丝： 637
关注： 1

搜索
找找看
谷歌搜索

我的标签
8266(88)
MQTT(50)
GPRS(33)
SDK(29)
Air202(28)
云服务器(21)
ESP8266(21)
Lua(18)
小程序(17)
STM32(16)
更多

随笔分类
Air724UG学习开发(2)
Android(22)
Android 开发(8)
C# 开发(4)
CH395Q学习开发(17)
CH579M学习开发(7)
ESP32学习开发(15)
ESP8266 AT指令开发(基于STC89C52单片机)(3)
ESP8266 AT指令开发(基于STM32)(1)
ESP8266 AT指令开发基础入门篇备份(12)
ESP8266 LUA脚本语言开发(13)

ESP8266 SDK开发: 外设篇-串口

ESP8266:SDK开发(源码见资料源码)

开发板购买链接:开发板购买链接

资料源码:https://github.com/yangfengwu45/learn-esp8266-sdk.git

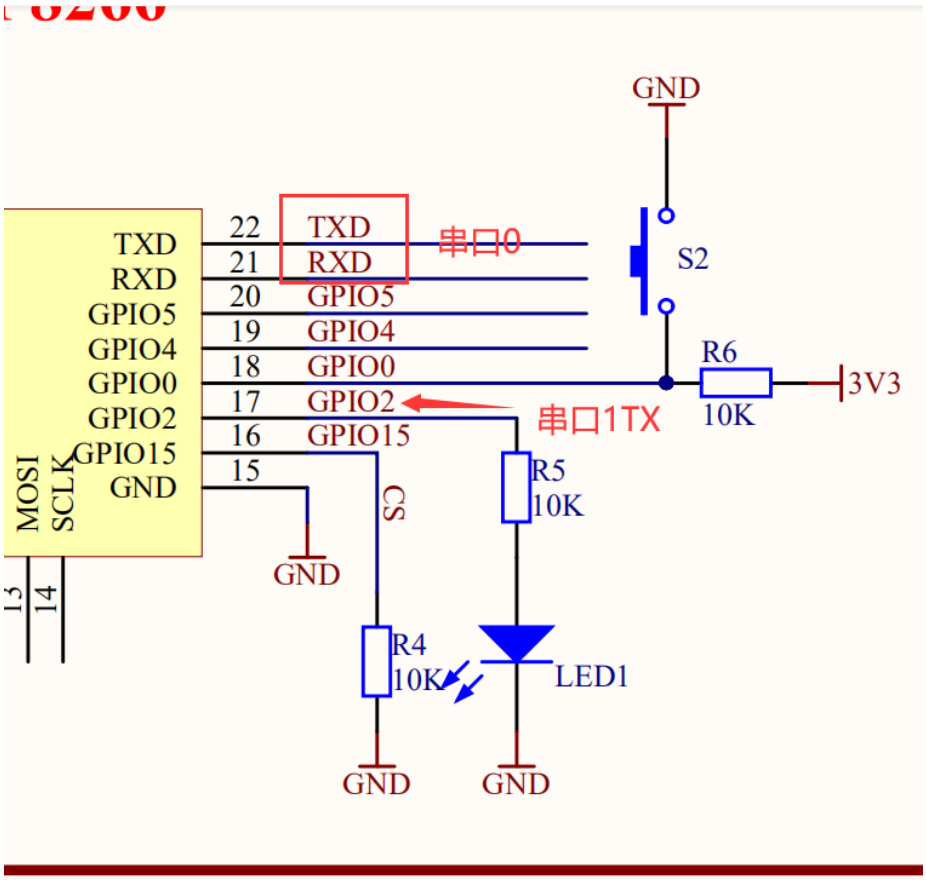
开发软件:https://mnifdv.cn/resource/cnblogs/Learn8266ForSDK/AiThinkerIDE_V0.5

点击加入群聊【ESP8266开发交流群】： 加入QQ群

- 基础开源教程:ESP8266:LUA脚本开发
- 基础开源教程:ESP8266 AT指令开发(基于51单片机)
- 基础开源教程:Android学习开发
- 基础开源教程:C#学习开发
- 基础开源教程:微信小程序开发入门篇
  需要搭配的Android，C#等基础教程如上，各个教程正在整理。

- 1.01-准备工作-硬件说明
- 1.02-整体运行测试-APP使用SmartConfig配网绑定ESP8266,并通过MQTT远程通信控制,采集DHT11温湿度数据
- 2.01 开发环境搭建(RTOS 2.2.0)(建议只参考这篇文章搭建即可,教程以NONOS版本为主!)
- 2.01 开发环境搭建(NONOS 2.2.0)
- 2.02-外设篇-GPIO输出高低电平
- 2.03-外设篇-GPIO输入检测
- 2.04-外设篇-GPIO中断检测
- 2.05-外设篇-定时器,延时
- 2.05-外设篇-系统任务(消息队列,通知)
- 2.06-外设篇-串口
- 2.07-外设篇-PWM,呼吸灯(RTOS 2.2.0)
- 2.08-外设篇-SPI(RTOS 2.2.0)
- 2.09-外设篇-温湿度传感器-DHT11
- 2.11-外设篇-时钟芯片DS1302使用和拓展知识time.h的使用
- 2.12-外设篇-内存分布说明及Flash读写
-
- 3.01-网络篇-8266TCP服务器(LWIP,RAW模式,PCB控制块)(RTOS 2.2.0)
- 3.02-网络篇-8266TCP服务器(espconn实现) (NONOS 2.2.0)
- 3.03-网络篇-8266连接路由器(实现局域网网络通信控制)
- 3.04-网络篇-TCP客户端(espconn) (NONOS 2.2.0)
- 3.10-网络篇-UDP通信 - 微信小程序篇-微信小程序通过UDP实现和ESP8266局域网通信控制
-
- 4.01-自建MQTT服务器篇-安装MQTT服务器,ESP8266连接MQTT服务器实现通信控制
- 4.02-自建MQTT服务器篇-ESP8266配网 SmartConfig
- 4.03-自建MQTT服务器篇-APP使用SmartConfig配网绑定ESP8266,并通过MQTT远程通信控制
- 4.05-自建MQTT服务器篇-编写微信小程序连接MQTT服务器程序
-
-
- 4.10 阿里云物联网平台篇-测试MQTT调试助手和ESP8266连接阿里云物联网平台

串口分布



串口内部自带一个FIFO缓存,数据接收以后先缓存到内部FIFO缓存里面

最新评论

1. Re:(一)Lua脚本语言入门
楼主可以分享一下这本电子书吗？
--戢思

2. Re:学习C语言-学习指针
学到了学到了，很清晰的思路，给博主赞赞赞
--*夏日么么茶

阅读排行榜

6. (一)基于阿里云的MQTT远程控制(Android 连接MQTT服务器,ESP8266连接MQTT服务器实现远程通信控制----简单的连接通信)(35982)
7. 关于TCP和MQTT之间的转换(33340)
8. C#中public与private与static(32610)
9. android 之TCP客户端编程(31968)
10. android客服端+eps8266+单片机+路由器之远程控制系统(31338)

推荐排行榜

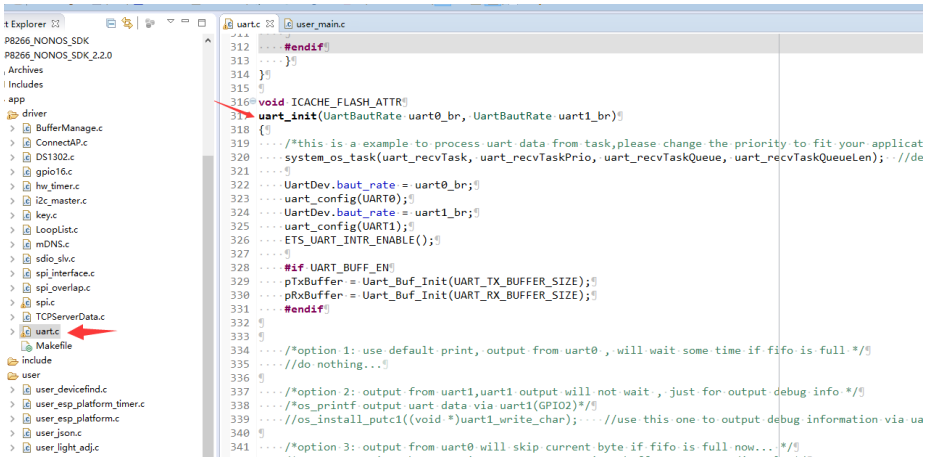
1. C#委托+回调详解(9)
2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(8)
3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
4. ESP8266使用详解(AT,LUA,SDK)(6)
5. 关于TCP和MQTT之间的转换(5)

内部FIFO满了以后进入FIFO满中断

串口打开了串口超时(空闲)中断:超过两个字节的时间没有接受到数据,进入串口超时(空闲)中断

NONOS(先看串口接收)

1.初始化串口是使用下面的函数



里面只写了设置波特率,如果需要设置其它参数可以参考代码的最下面

uart.c | user_main.c

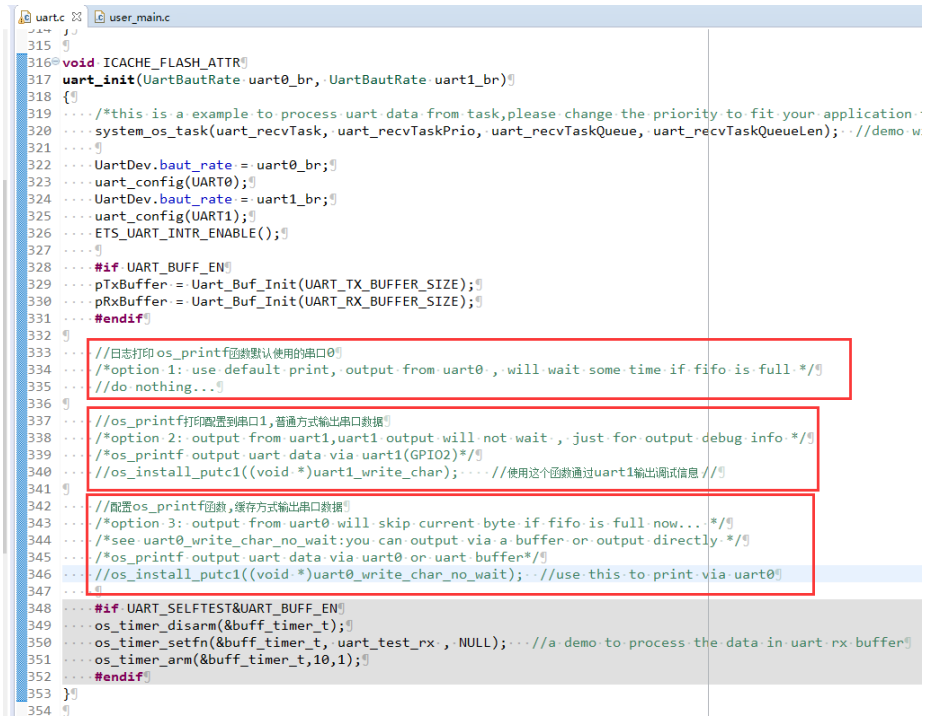
```
void ICACHE_FLASH_ATTR
UART_SetPrintPort(uint8 uart_no)
{
    if(uart_no==1){
        os_install_putc1(uart1_write_char);
    }else{
        /*option 1: do not wait if uart fifo is full,drop current character*/
        os_install_putc1(uart0_write_char_no_wait);
    /*option 2: wait for a while if uart fifo is full*/
    os_install_putc1(uart0_write_char);
    }
}


//=========================================================
/*test code*/
void ICACHE_FLASH_ATTR
uart_init_2(UartBautRate uart0_br, UartBautRate uart1_br)
{
    UartDev.baut_rate = uart0_br;
//    UartDev.exist_parity = STICK_PARITY_EN;
    UartDev.parity = NONE_BITS;//无奇偶校验
    UartDev.stop_bits = ONE_STOP_BIT;//1位停止位
    UartDev.data_bits = EIGHT_BITS;//8位数据

    uart_config(UART0);
    UartDev.baut_rate = uart1_br;
    uart_config(UART1);
    ETS_UART_INTR_ENABLE();

    // install uart1 putc callback
//    os_install_putc1((void *)uart1_write_char);//os_printf使用串口1打印(GPIO2)

    os_install_putc1((void *)uart0_write_char);//os_printf使用串口0打印
}


```

uart.c | user_main.c

```
 2 * ESPRESSIF MIT License
24
25 #include "ets_sys.h"
26 #include "osapi.h"
27 #include "user_interface.h"
28 #include "user_devicefind.h"
29 #include "user_webserver.h"
30 #if ESP_PLATFORM
31 #include "user_esp_platform.h"
32 #endif
33 #include "driver/uart.h"
34
35 uint32 priv_param_start_sec;
37 user_rf_cal_sector_set(void)
87 void ICACHE_FLASH_ATTR
88 user_rf_pre_init(void){}
89
90 /******************************************************************
91  * FunctionName : user_init
92  * Description  : entry of user application, init user functior
93  * Parameters   : none
94  * Returns      : none
95  *******************************************************************/
96 void ICACHE_FLASH_ATTR user_init(void){
97     uart_init(BIT_RATE_115200,BIT_RATE_115200);
98 }
99
100
```

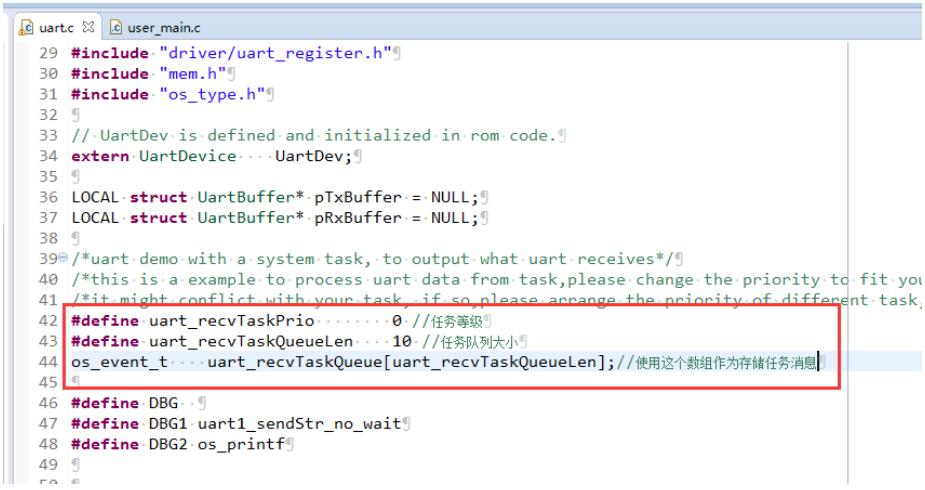
2.默认是使用串口0输出日志

咱先不用修改,咱先把串口基本操作学完

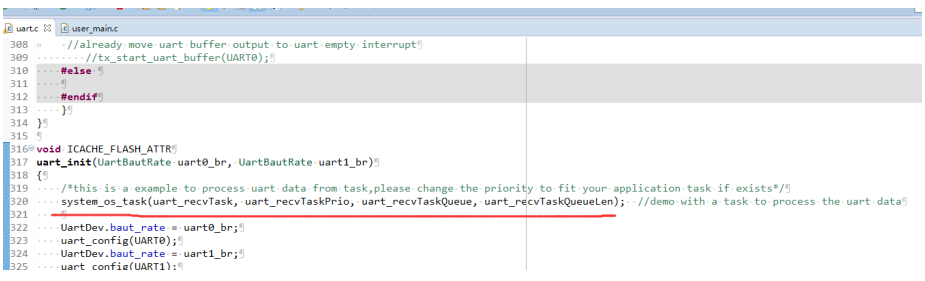


```
void ICACHE_FLASH_ATTR
uart_init(UartBautRate uart0_br, UartBautRate uart1_br)
{
    /*this is a example to process uart data from task,please change the priority to fit your application
    system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen);  //demo w

    UartDev.baut_rate = uart0_br;
    uart_config(UART0);
    UartDev.baut_rate = uart1_br;
    uart_config(UART1);
    ETS_UART_INTR_ENABLE();

    #if UART_BUFF_EN
    pTxBuffer = Uart_Buf_Init(UART_TX_BUFFER_SIZE);
    pRxBuffer = Uart_Buf_Init(UART_RX_BUFFER_SIZE);
    #endif

    //日志打印os_printf函数默认使用的串口0
    /*option 1: use default print, output from uart0 , will wait some time if fifo is full */
    //do nothing...

    //os_printf打印配置到串口1,普通方式输出串口数据
    /*option 2: output from uart1,uart1 output will not wait , just for output debug info */
    /*os_printf output uart data via uart1(GPIO2)*/
    //os_install_putc1((void *)uart1_write_char);    //使用这个函数通过uart1输出调试信息 //

    //配置os_printf函数,缓存方式输出串口数据
    /*option 3: output from uart0 will skip current byte if fifo is full now... */
    /*see uart0_write_char_no_wait:you can output via a buffer or output directly */
    /*os_printf output uart data via uart0 or uart buffer*/
    //os_install_putc1((void *)uart0_write_char_no_wait);  //use this to print via uart0

    #if UART_SELFTEST&UART_BUFF_EN
    os_timer_disarm(&buff_timer_t);
    os_timer_setfn(&buff_timer_t, uart_test_rx , NULL);  //a demo to process the data in uart rx buffer
    os_timer_arm(&buff_timer_t,10,1);
    #endif
}
```

3.提供的串口中断接收里面是使用任务通知的形式(关于任务通知参见上一节系统任务(消息队列,通知))

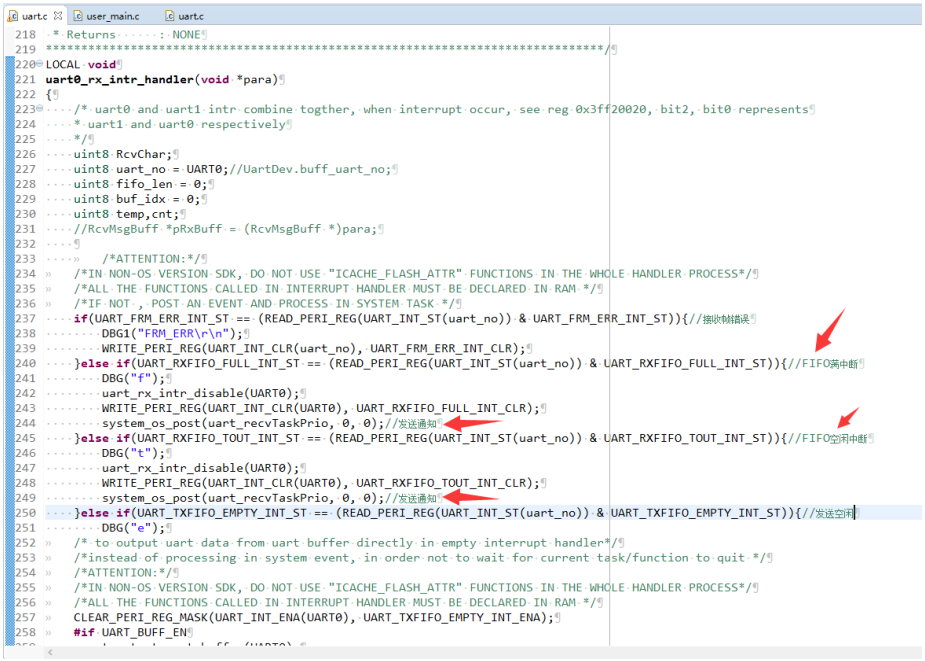


```
#include "driver/uart_register.h"
#include "mem.h"
#include "os_type.h"

// UartDev is defined and initialized in rom code.
extern UartDevice    UartDev;

LOCAL struct UartBuffer* pTxBuffer = NULL;
LOCAL struct UartBuffer* pRxBuffer = NULL;

/*uart demo with a system task, to output what uart receives*/
/*this is a example to process uart data from task,please change the priority to fit you
/*it might conflict with your task, if so,please arrange the priority of different task,
#define uart_recvTaskPrio        0 //任务等级
#define uart_recvTaskQueueLen    10 //任务队列大小
os_event_t    uart_recvTaskQueue[uart_recvTaskQueueLen];//使用这个数组作为存储任务消息

#define DBG
#define DBG1 uart1_sendStr_no_wait
#define DBG2 os_printf
```



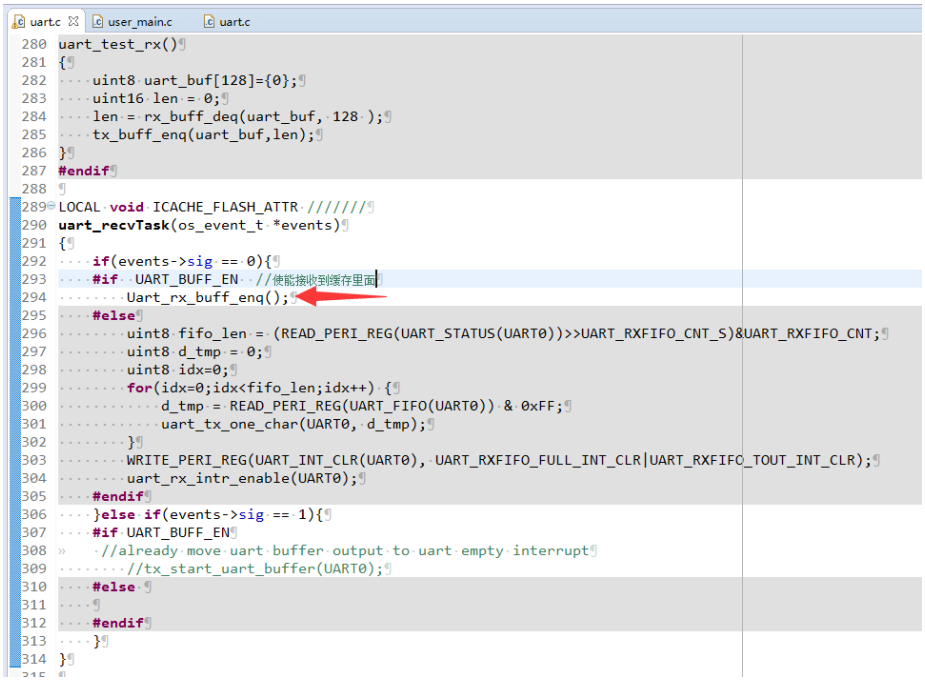
```
        //already move uart buffer output to uart empty interrupt
        //tx_start_uart_buffer(UART0);
    #else
    #endif
    }
}

void ICACHE_FLASH_ATTR
uart_init(UartBautRate uart0_br, UartBautRate uart1_br)
{
    /*this is a example to process uart data from task,please change the priority to fit your application task if exists*/
    system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen);  //demo with a task to process the uart data

    UartDev.baut_rate = uart0_br;
    uart_config(UART0);
    UartDev.baut_rate = uart1_br;
    uart_config(UART1);
```

在内部FIFO接收到数据的时候发送任务消息出去



```
 * Returns      : NONE
*******************************************************************************/
LOCAL void
uart0_rx_intr_handler(void *para)
{
    /* uart0 and uart1 intr combine togther, when interrupt occur, see reg 0x3ff20020, bit2, bit0 represents
    * uart1 and uart0 respectively
    */
    uint8 RcvChar;
    uint8 uart_no = UART0;//UartDev.buff_uart_no;
    uint8 fifo_len = 0;
    uint8 buf_idx = 0;
    uint8 temp,cnt;
    //RcvMsgBuff *pRxBuff = (RcvMsgBuff *)para;

    /*ATTENTION:*/
   /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
   /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM */
   /*IF NOT , POST AN EVENT AND PROCESS IN SYSTEM TASK */
    if(UART_FRM_ERR_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_FRM_ERR_INT_ST)){//接收帧错误
        DBG1("FRM_ERR\r\n");
        WRITE_PERI_REG(UART_INT_CLR(uart_no), UART_FRM_ERR_INT_CLR);
    }else if(UART_RXFIFO_FULL_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_RXFIFO_FULL_INT_ST)){//FIFO满中断
        DBG("f");
        uart_rx_intr_disable(UART0);
        WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR);
        system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
    }else if(UART_RXFIFO_TOUT_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_RXFIFO_TOUT_INT_ST)){//FIFO空闲中断
        DBG("t");
        uart_rx_intr_disable(UART0);
        WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_TOUT_INT_CLR);
        system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
    }else if(UART_TXFIFO_EMPTY_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_TXFIFO_EMPTY_INT_ST)){//发送空闲
        DBG("e");
   /* to output uart data from uart buffer directly in empty interrupt handler*/
   /*instead of processing in system event, in order not to wait for current task/function to quit */
   /*ATTENTION:*/
   /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
   /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM */
   CLEAR_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);
   #if UART_BUFF_EN
```

在任务中读取数据(读取数据默认提供的是存储到数组缓存里面)

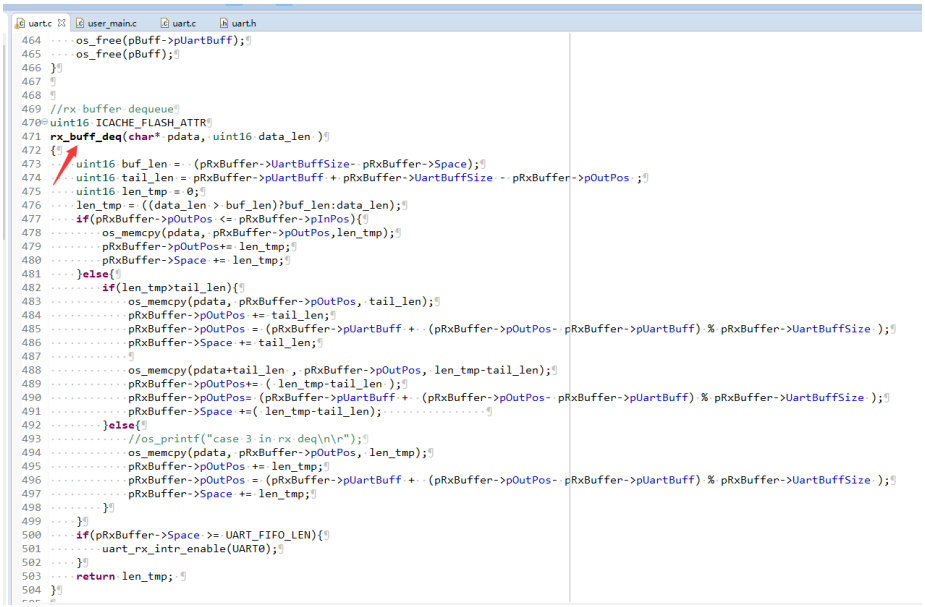


```
uart_test_rx()
{
    uint8 uart_buf[128]={0};
    uint16 len = 0;
    len = rx_buff_deq(uart_buf, 128 );
    tx_buff_enq(uart_buf,len);
}
#endif

LOCAL void ICACHE_FLASH_ATTR ///////
uart_recvTask(os_event_t *events)
{
    if(events->sig == 0){
    #if UART_BUFF_EN //使能接收到缓存里面
        Uart_rx_buff_enq();
    #else
        uint8 fifo_len = (READ_PERI_REG(UART_STATUS(UART0))>>UART_RXFIFO_CNT_S)&UART_RXFIFO_CNT;
        uint8 d_tmp = 0;
        uint8 idx=0;
        for(idx=0;idx<fifo_len;idx++) {
            d_tmp = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;
            uart_tx_one_char(UART0, d_tmp);
        }
        WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR|UART_RXFIFO_TOUT_INT_CLR);
        uart_rx_intr_enable(UART0);
    #endif
    }else if(events->sig == 1){
    #if UART_BUFF_EN
        //already move uart buffer output to uart empty interrupt
        //tx_start_uart_buffer(UART0);
    #else

    #endif
    }
}
```

```
//move data from uart fifo to rx buffer
void Uart_rx_buff_enq()
{
    uint8 fifo_len,buf_idx;
    uint8 fifo_data;
    #if 1
    //从FIFO中获取串口接收的数据个数
    fifo_len = (READ_PERI_REG(UART_STATUS(UART0))>>UART_RXFIFO_CNT_S)&UART_RXFIFO_CNT;
    if(fifo_len >= pRxBuffer->Space){
        os_printf("buf full!!!\n\r");
    }else{
        buf_idx=0;
        while(buf_idx < fifo_len){
            buf_idx++;
            fifo_data = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;//读取数据
            *(pRxBuffer->pInPos++) = fifo_data;
            if(pRxBuffer->pInPos == (pRxBuffer->pUartBuff + pRxBuffer->UartBuffSize)){
                pRxBuffer->pInPos = pRxBuffer->pUartBuff;
            }
        }
        pRxBuffer->Space -= fifo_len ;
        if(pRxBuffer->Space >= UART_FIFO_LEN){
            //os_printf("after rx enq buf enough\n\r");
            uart_rx_intr_enable(UART0);
        }
    }
    #endif
}
```

4.咱们获取数据呢是使用下面的函数

函数返回值是获取的数据个数;形参1是咱要把数据拷贝到的数组地址;形参2是咱想获取的数据个数



```
os_free(pBuff->pUartBuff);
os_free(pBuff);
}

//rx buffer dequeue
uint16 ICACHE_FLASH_ATTR
rx_buff_deq(char* pdata, uint16 data_len )
{
    uint16 buf_len = (pRxBuffer->UartBuffSize- pRxBuffer->Space);
    uint16 tail_len = pRxBuffer->pUartBuff + pRxBuffer->UartBuffSize - pRxBuffer->pOutPos ;
    uint16 len_tmp = 0;
    len_tmp = ((data_len > buf_len)?buf_len:data_len);
    if(pRxBuffer->pOutPos <= pRxBuffer->pInPos){
        os_memcpy(pdata, pRxBuffer->pOutPos,len_tmp);
        pRxBuffer->pOutPos+= len_tmp;
        pRxBuffer->Space += len_tmp;
    }else{
        if(len_tmp>tail_len){
            os_memcpy(pdata, pRxBuffer->pOutPos, tail_len);
            pRxBuffer->pOutPos += tail_len;
            pRxBuffer->pOutPos = (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos- pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize );
            pRxBuffer->Space += tail_len;

            os_memcpy(pdata+tail_len , pRxBuffer->pOutPos, len_tmp-tail_len);
            pRxBuffer->pOutPos+= ( len_tmp-tail_len );
            pRxBuffer->pOutPos= (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos- pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize );
            pRxBuffer->Space +=( len_tmp-tail_len);
        }else{
            //os_printf("case 3 in rx deq\n\r");
            os_memcpy(pdata, pRxBuffer->pOutPos, len_tmp);
            pRxBuffer->pOutPos += len_tmp;
            pRxBuffer->pOutPos = (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos- pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize );
            pRxBuffer->Space += len_tmp;
        }
    }
    if(pRxBuffer->Space >= UART_FIFO_LEN){
        uart_rx_intr_enable(UART0);
    }
    return len_tmp;
}
```

5.官方还贴心的给了例子

定时器每隔10ms轮训,每次尝试获取128字节数据,获取到数据之后,直接返回接收的数据

uart.c | user_main.c | uart.h

```
315 }
316 void ICACHE_FLASH_ATTR
317 uart_init(UartBautRate uart0_br, UartBautRate uart1_br)
318 {
319     /*this is a example to process uart data from task,please change the priority to fit your application task if
320     system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen); //demo with a t
321     
322     UartDev.baut_rate = uart0_br;
323     uart_config(UART0);
324     UartDev.baut_rate = uart1_br;
325     uart_config(UART1);
326     ETS_UART_INTR_ENABLE();
327     
328     #if UART_BUFF_EN
329     pTxBuffer = Uart_Buf_Init(UART_TX_BUFFER_SIZE);
330     pRxBuffer = Uart_Buf_Init(UART_RX_BUFFER_SIZE);
331     #endif
332 
333     //日志打印 os_printf函数默认使用的串口0
334     /*option 1: use default print, output from uart0 , will wait some time if fifo is full */
335     //do nothing...
336 
337     //os_printf打印配置到串口1,普通方式输出串口数据
338     /*option 2: output from uart1,uart1 output will not wait , just for output debug info */
339     /*os_printf output uart data via uart1(GPIO2)*/
340     //os_install_putc1((void *)uart1_write_char);     //使用这个函数通过uart1输出调试信息 //
341 
342     //配置os_printf函数,缓存方式输出串口数据
343     /*option 3: output from uart0 will skip current byte if fifo is full now... */
344     /*see uart0_write_char_no_wait:you can output via a buffer or output directly */
345     /*os_printf output uart data via uart0 or uart buffer*/
346     //os_install_putc1((void *)uart0_write_char_no_wait); //use this to print via uart0
347     
348     #if UART_SELFTEST&UART_BUFF_EN
349     os_timer_disarm(&buff_timer_t);
350     os_timer_setfn(&buff_timer_t, uart_test_rx , NULL);   //a demo to process the data in uart rx buffer
351     os_timer_arm(&buff_timer_t,10,1);
352     #endif
353 }
354 
355 void ICACHE_FLASH_ATTR
```

```
270 /******************************************************************************
271  * FunctionName : uart_init
272  * Description  : user interface for init uart
273  * Parameters   : UartBautRate uart0_br - uart0 bautrate
274  *                UartBautRate uart1_br - uart1 bautrate
275  * Returns      : NONE
276 *******************************************************************************/
277 #if UART_SELFTEST&UART_BUFF_EN
278 os_timer_t buff_timer_t;
279 void ICACHE_FLASH_ATTR
280 uart_test_rx()
281 {
282     uint8 uart_buf[128]={0};
283     uint16 len = 0;
284     len = rx_buff_deq(uart_buf, 128 );
285     tx_buff_enq(uart_buf,len);
286 }
287 #endif
288 
289 LOCAL void ICACHE_FLASH_ATTR ///////
```

6.咱们就直接拷贝到主函数试一试

```
uart.c | user_main.c | uart.h | osapi.h
#include "user_interface.h"
#include "user_devicefind.h"
#include "user_webserver.h"
#if ESP_PLATFORM
#include "user_esp_platform.h"
#endif
#include "driver/uart.h"


uint32 priv_param_start_sec;
user_rf_cal_sector_set(void)
void ICACHE_FLASH_ATTR
user_rf_pre_init(void){}


os_timer_t buff_timer_t;
void ICACHE_FLASH_ATTR
uart_test_rx()
{
    uint8 uart_buf[128]={0};
    uint16 len = 0;
    len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据
    tx_buff_enq(uart_buf,len);//把接收的数据返回
}

/******************************************************************************
 * FunctionName : user_init
 * Description  : entry of user application, init user function here
 * Parameters   : none
 * Returns      : none
 *******************************************************************************/
void ICACHE_FLASH_ATTR user_init(void){
    uart_init(BIT_RATE_115200,BIT_RATE_115200);

    os_timer_disarm(&buff_timer_t);
    os_timer_setfn(&buff_timer_t, uart_test_rx , NULL);    //a demo to process the data in uart rx buffer
    os_timer_arm(&buff_timer_t,10,1);
}
```

7.串口接收缓存默认是256字节,如果数据的每一帧的数据量大于这个值,就需要增加

uart.c | user_main.c | uart.h | osapi.h

```
 2 * ESPRESSIF MIT License
24
25 #ifndef UART_APP_H
26 #define UART_APP_H
27
28 #include "uart_register.h"
29 #include "eagle_soc.h"
30 #include "c_types.h"
31
32 #define UART_TX_BUFFER_SIZE 256  //Ring buffer length of tx buffer
33 #define UART_RX_BUFFER_SIZE 256 //Ring buffer length of rx buffer
34
35 #define UART_BUFF_EN  1   //use uart buffer  , FOR UART0
36 #define UART_SELFTEST  0  //set 1:enable the loop test demo for uart buffer, FOR UART0
37
38 #define UART_HW_RTS   0   //set 1: enable uart hw flow control RTS, PIN MTDO, FOR UART0
39 #define UART_HW_CTS  0    //set1: enable uart hw flow contrl CTS , PIN MTCK, FOR UART0
40
41
```

8.再提醒下哈,下面只是尝试获取128字节,假设缓存里面有10个数据,那么也只会返回10个.

假设缓存里面有500个,那么会返回128个.

一般这个值取最大可能返回的数据个数.

uart.c | user_main.c | uart.h | osapi.h

```
26 #include "osapi.h"
27 #include "user_interface.h"
28 #include "user_devicefind.h"
29 #include "user_webserver.h"
30 #if ESP_PLATFORM
31 #include "user_esp_platform.h"
32 #endif
33 #include "driver/uart.h"
34
35
36 uint32 priv_param_start_sec;
38 user_rf_cal_sector_set(void)
88 void ICACHE_FLASH_ATTR
89 user_rf_pre_init(void){}
90
91
92 os_timer_t buff_timer_t;
93 void ICACHE_FLASH_ATTR
94 uart_test_rx()
95 {
96     uint8 uart_buf[128]={0};
97     uint16 len = 0;
98     len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据
99     tx_buff_enq(uart_buf,len);//把接收的数据返回
100 }
101
102 /*****************************************************************
```

9.注意:一般是接收到一条完整的数据之后再去处理数据,但是特殊场合需要快速的通信,如果每条数据的时间间隔小于10ms,

那么便会出现粘包.可以使用任务把轮训间隔缩小到1ms;

首先把串口的任务优先级改为最高

```
uart.c  user_main.c  uart.h  osapi.h
 2 * ESPRESSIF MIT License
24
25 #include "ets_sys.h"
26 #include "osapi.h"
27 #include "driver/uart.h"
28 #include "osapi.h"
29 #include "driver/uart_register.h"
30 #include "mem.h"
31 #include "os_type.h"
32
33 // UartDev is defined and initialized in rom code.
34 extern UartDevice    UartDev;
35
36 LOCAL struct UartBuffer* pTxBuffer = NULL;
37 LOCAL struct UartBuffer* pRxBuffer = NULL;
38
39 /*uart demo with a system task, to output what uart receives*/
40 /*this is a example to process uart data from task,please change the priority to fit yo
41 /*it might conflict with your task, if so,please arrange the priority of different task
42 #define uart_recvTaskPrio        2 //任务等级
43 #define uart_recvTaskQueueLen    10 //任务队列大小
44 os_event_t    uart_recvTaskQueue[uart_recvTaskQueueLen];//使用这个数组作为存储任务消息
45
46 #define DBG
47 #define DBG1 uart1_sendStr_no_wait
48 #define DBG2 os_printf
49
50
```

然后在主函数加一个优先级别低的任务

```
uart.c  user_main.c  uart.h  osapi.h
32 #endif
33 #include "driver/uart.h"
34
35 /************任务*************/
36 #define os_event_t_buff_len 255 /*消息队列长度;最大255*/
37 os_event_t    os_event_t_buff[os_event_t_buff_len]; //存储消息的数组
38 #define TaskPrio 0 //任务等级(0,1,2(最高))
39 uint32 os_task_t_delay=0;//在任务里面做延时
40 /************串口接收缓存*************/
41 uint8 uart_buf[UART_RX_BUFFER_SIZE]={0};
42 uint16 len = 0;
43
44 uint32 priv_param_start_sec;
46 user_rf_cal_sector_set(void)
96 void ICACHE_FLASH_ATTR
97 user_rf_pre_init(void){}
98
99 void os_task_t_callback(os_event_t *events){
100     if(events->sig == 0 && events->par ==0){
101         system_os_post(TaskPrio, 0, 0);
102     }
103     os_task_t_delay++;
104     if(os_task_t_delay>300){//大约1ms
105         os_task_t_delay=0;
106         len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据
107         tx_buff_enq(uart_buf,len);//把接收的数据返回
108     }
109 }
110
111 /******************************************************************************
112  * FunctionName : user_init
113  * Description  : entry of user application, init user function here
114  * Parameters   : none
115  * Returns      : none
116 *******************************************************************************/
117 void ICACHE_FLASH_ATTR user_init(void){
118     uart_init(BIT_RATE_115200,BIT_RATE_115200);
119
120     system_os_task(os_task_t_callback, TaskPrio, os_event_t_buff, os_event_t_buff_len);
121     system_os_post(TaskPrio, 0, 0);
122 }
```

```
/************任务*************/
#define os_event_t_buff_len 255 /*消息队列长度;最大255*/

os_event_t    os_event_t_buff[os_event_t_buff_len]; //存储消息的数组
#define TaskPrio 0 //任务等级(0,1,2(最高))
```

```
uint32 os_task_t_delay=0;//在任务里面做延时
/************串口接收缓存**************/
uint8 uart_buf[UART_RX_BUFFER_SIZE]={0};
uint16 len = 0;
```

```
void os_task_t_callback(os_event_t *events){
    if(events->sig == 0 && events->par ==0){
        system_os_post(TaskPrio, 0, 0);
    }
    os_task_t_delay++;
    if(os_task_t_delay>300){//大约1ms
        os_task_t_delay=0;
        len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据
        tx_buff_enq(uart_buf,len);//把接收的数据返回
    }
}
```

```
system_os_task(os_task_t_callback, TaskPrio, os_event_t_buff, os_event_t_buff
    system_os_post(TaskPrio, 0, 0);
```

当然一般串口10ms已经适应了基本上所有的项目了

```
uint32 priv_param_start_sec;
user_rf_cal_sector_set(void)
void ICACHE_FLASH_ATTR
user_rf_pre_init(void){}

void os_task_t_callback(os_event_t *events){
    if(events->sig == 0 && events->par ==0){
        system_os_post(TaskPrio, 0, 0);
    }
    os_task_t_delay++;
    if(os_task_t_delay>3000){//大约10ms
        os_task_t_delay=0;
        len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据
        tx_buff_enq(uart_buf,len);//把接收的数据返回
    }
}
```

10.如果想把数据直接存储到自己定义数组里面

把UART_BUFF_EN改为0

uart.c | user_main.c | uart.h | osapi.h

```
/* ESPRESSIF MIT License
*/
#ifndef UART_APP_H
#define UART_APP_H

#include "uart_register.h"
#include "eagle_soc.h"
#include "c_types.h"

#define UART_TX_BUFFER_SIZE 256 //Ring buffer length of tx buffer
#define UART_RX_BUFFER_SIZE 256 //Ring buffer length of rx buffer

#define UART_BUFF_EN  0  //use uart buffer , FOR UART0
#define UART_SELFTEST  0  //set 1:enable the loop test demo for uart buffer, FOR UART0

#define UART_HW_RTS   0   //set 1: enable uart hw flow control RTS, PIN MTDO, FOR UART0
#define UART_HW_CTS   0   //set1: enable uart hw flow contrl CTS , PIN MTCK, FOR UART0
```

uart.c | user_main.c | uart.h | osapi.h

```
#if UART_SELFTEST&UART_BUFF_EN
os_timer_t buff_timer_t;
void ICACHE_FLASH_ATTR
uart_test_rx()
{
    uint8 uart_buf[128]={0};
    uint16 len = 0;
    len = rx_buff_deq(uart_buf, 128 );
    tx_buff_enq(uart_buf,len);
}
#endif

LOCAL void ICACHE_FLASH_ATTR ///////
uart_recvTask(os_event_t *events)
{
    if(events->sig == 0){
        #if UART_BUFF_EN  //使能接收到缓存里面
            Uart_rx_buff_enq();
        #else
            uint8 fifo_len = (READ_PERI_REG(UART_STATUS(UART0))>>UART_RXFIFO_CNT_S)&UART_RXFIFO_CNT;
            uint8 d_tmp = 0;
            uint8 idx=0;
            for(idx=0;idx<fifo_len;idx++) {
                d_tmp = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;
                uart_tx_one_char(UART0, d_tmp);
            }
            WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR|UART_RXFIFO_TOUT_INT_CLR);
            uart_rx_intr_enable(UART0);
        #endif
    }else if(events->sig == 1){
    #if UART_BUFF_EN
      //already move uart buffer output to uart empty interrupt
        //tx_start_uart_buffer(UART0);
    #else
```

屏蔽掉这句,在这里把数据存放到自己的数

NONOS(串口发送数据)

默认的发送是使用的缓存+中断发送

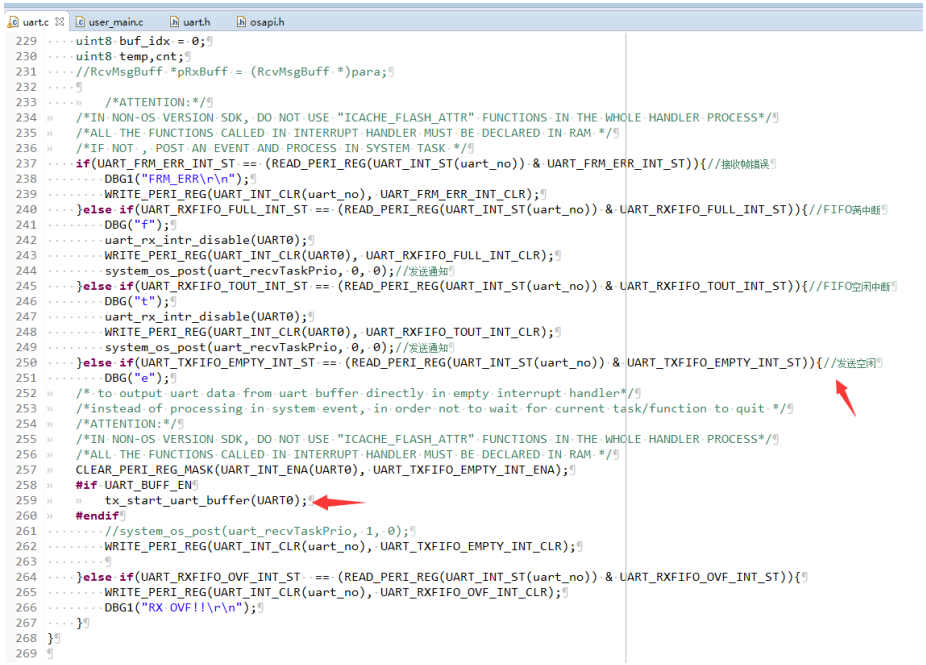
建议用户直接使用这种方式.因为发送数据的时候不会阻塞.

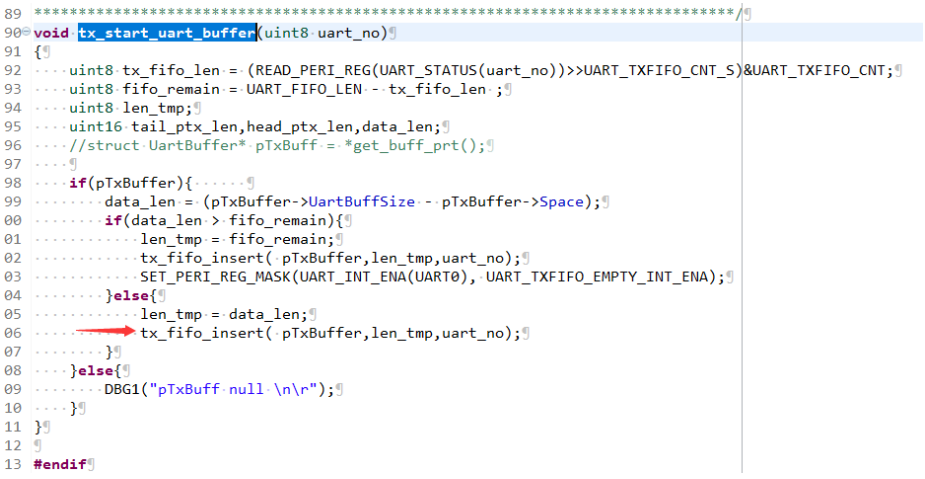
```
98 ¶
99 void os_task_t_callback(os_event_t *events){¶
100 ····if(events->sig == 0 && events->par ==0){¶
101 ····»    system_os_post(TaskPrio, 0, 0);¶
102 ····}¶
103 ····os_task_t_delay++;¶
104 ····if(os_task_t_delay>3000){//大约10ms¶
105 ····»    os_task_t_delay=0;¶
106 ····»    len = rx_buff_deq(uart_buf, 128 );//尝试从缓存中获取128字节串口数据¶
107 ····»    tx_buff_enq(uart_buf,len);//把接收的数据返回¶
108 ····}¶
109 }¶
110 ¶
111 /*****************************************************************************
```

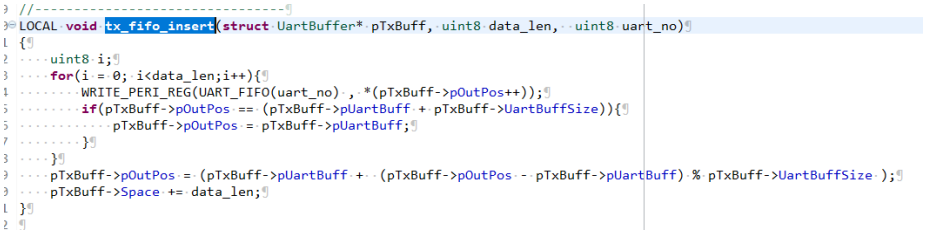
```
536 ¶
537 //fill the uart tx buffer¶
538 void ICACHE_FLASH_ATTR¶
539 tx_buff_enq(char* pdata, uint16 data_len )¶
540 {¶
541 ····CLEAR_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);¶
542 ¶
543 ····if(pTxBuffer == NULL){¶
544 ····»    DBG1("\n\rnull, create buffer struct\n\r");¶
545 ····»    pTxBuffer = Uart_Buf_Init(UART_TX_BUFFER_SIZE);¶
546 ····»    if(pTxBuffer!= NULL){¶
547 ····»    »    Uart_Buf_Cpy(pTxBuffer , pdata, data_len );¶
548 ····»    }else{¶
549 ····»    »    DBG1("uart tx MALLOC no buf \n\r");¶
550 ····»    }¶
551 ····}else{¶
552 ····»    if(data_len <= pTxBuffer->Space){¶
553 ····»    Uart_Buf_Cpy(pTxBuffer , pdata, data_len);¶
554 ····»    }else{¶
555 ····»    »    DBG1("UART TX BUF FULL!!!!\n\r");¶
556 ····»    }¶
557 ····}¶
558 ····#if 0¶
559 ····if(pTxBuffer->Space <= URAT_TX_LOWER_SIZE){¶
560 »    »    set_tcp_block();    ¶
561 ····}¶
562 ····#endif¶
563 ····SET_PERI_REG_MASK(UART_CONF1(UART0), (UART_TX_EMPTY_THRESH_VAL & UART_TXFIFO_EMPTY_THRHD)<<UART_TXFIFO_EMPTY_THRHD_S);¶
564 ····SET_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);¶
565 }¶
566 ¶
567 ¶
```

把数据储存到缓存

使能中断发送







关于 os_printf

os_printf函数一般是做日志打印的,默认是使用串口0打印;可以配置到串口1上(gpio2口上)

```c
uart.c | user_main.c | uart.h | osapi.h

void ICACHE_FLASH_ATTR
uart_init(UartBautRate uart0_br, UartBautRate uart1_br)
{
    /*this is a example to process uart data from task,please change the priority to fit your applica
    system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen); //d

    UartDev.baut_rate = uart0_br;
    uart_config(UART0);
    UartDev.baut_rate = uart1_br;
    uart_config(UART1);
    ETS_UART_INTR_ENABLE();

    #if UART_BUFF_EN
    pTxBuffer = Uart_Buf_Init(UART_TX_BUFFER_SIZE);
    pRxBuffer = Uart_Buf_Init(UART_RX_BUFFER_SIZE);
    #endif

    //日志打印 os_printf函数默认使用的串口0
    /*option 1: use default print, output from uart0 , will wait some time if fifo is full */
    //do nothing...

    //os_printf打印配置到串口1,普通方式输出串口数据
    /*option 2: output from uart1,uart1 output will not wait , just for output debug info */
    /*os_printf output uart data via uart1(GPIO2)*/
    os_install_putc1((void *)uart1_write_char);    //使用这个函数通过uart1输出调试信息 //

    //配置os_printf函数,缓存方式输出串口数据
    /*option 3: output from uart0 will skip current byte if fifo is full now... */
    /*see uart0_write_char_no_wait:you can output via a buffer or output directly */
    /*os_printf output uart data via uart0 or uart buffer*/
    //os_install_putc1((void *)uart0_write_char_no_wait);  //use this to print via uart0

    #if UART_SELFTEST&UART_BUFF_EN
    os_timer_disarm(&buff_timer_t);
    os_timer_setfn(&buff_timer_t, uart_test_rx , NULL);  //a demo to process the data in uart rx buf
    os_timer_arm(&buff_timer_t,10,1);
    #endif
}

void ICACHE_FLASH_ATTR
```

os_printf函数可以使用下面的函数关闭或者开启打印

```c

/******************************************************************************
 * FunctionName : user_init
 * Description  : entry of user application, init user function here
 * Parameters   : none
 * Returns      : none
*******************************************************************************/
void ICACHE_FLASH_ATTR user_init(void){
    uart_init(BIT_RATE_115200,BIT_RATE_115200);
    system_set_os_print (0);//0:关闭打印 1:开启打印
    system_os_task(os_task_t_callback, TaskPrio, os_event_t_buff, os_event_t_buff_len);
    system_os_post(TaskPrio, 0, 0);
}


```

system_set_os_print (0);//0:关闭打印 1:开启打印

分类: ESP8266 SDK开发

杨奉武
关注 - 1
粉丝 - 637

0 0

« 上一篇： ESP8266 SDK开发: 外设篇-定时器,延时
» 下一篇： C#开发: 准备工作-Visual Studio 安装

posted on 2020-02-27 23:55 杨奉武 阅读(1598) 评论(0) 编辑 收藏 举报

发表评论

编辑 预览

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]

编辑推荐：
· C# 10 完整特性介绍
· 不是技术也能看懂云原生
· 记一次接口慢查排查
· 一个故事看懂HTTPS
· 人人都能看懂系列：分布式系统改造方案之数据篇

最新新闻：
· 你还抢购华为吗？门店可能没有存货了
· 字节新消费版图大起底：投资自营双管齐下
· 上市破发、资金受困，理想“勇争第一”空成口号
· 百度二季度财报点评：以更高维的ESG识别其价值
· 锂电专利战争：欧美、日韩围剿，中国换道超车
» 更多新闻...

单片机,物联网,上位机,…

扫一扫二维码，加入群聊。