

优秀不够，你是否无可替代

知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人
QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 751, 文章 - 0, 评论 - 317, 阅读 - 183万

导航

博客园
首页
新随笔
联系
订阅 
管理

公告



 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](#)

ESP8266 LUA开发基础入门篇
备份(22)
ESP8266 SDK开发(33)
ESP8266 SDK开发基础入门篇
备份(30)
GPRS Air202 LUA开发(11)
HC32F460(华大) +
BC260Y(NB-IOT) 物联网开发
(5)
NB-IOT Air302 AT指令和LUA
脚本语言开发(25)
PLC(三菱PLC)基础入门篇(2)
STM32+Air724UG(4G模组)
物联网开发(43)
STM32+BC26/260Y物联网开
发(37)
STM32+CH395Q(以太网)物
联网开发(21)
STM32+ESP8266(ZLESP8266/
物联网开发(1)
STM32+ESP8266+AIR202/30:
远程升级方案(16)
STM32+ESP8266+AIR202/30:
终端管理方案(6)
STM32+ESP8266+Air302物
联网开发(64)
STM32+W5500+AIR202/302
基本控制方案(25)
STM32+W5500+AIR202/302
远程升级方案(6)
UCOSii操作系统(1)
W5500 学习开发(8)
编程语言C#(11)
编程语言Lua脚本语言基础入
门篇(6)
编程语言Python(1)
单片机(LPC1778)LPC1778(2)
单片机(MSP430)开发基础入门
篇(4)
单片机(STC89C51)单片机开发
板学习入门篇(3)
单片机(STM32)基础入门篇(3)
单片机(STM32)综合应用系列
(16)
电路模块使用说明(11)
感想(6)
软件安装使用: MQTT(8)
更多

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

1. ESP8266使用详解(AT,LUA,
SDK)(172847)
2. 1-安装MQTT服务器(Windo
ws),并连接测试(99168)
3. ESP8266刷AT固件与node
mcu固件(64823)
4. 用ESP8266+android,制作
自己的WIFI小车(ESP8266篇)
(64354)
5. 有人WIFI模块使用详解(385
48)

[4.11-阿里云物联网平台篇-ESP8266连接阿里云物联网平台使用自定义Topic实现自定义数据的上报和数据下发](#)

[4.12-阿里云物联网平台篇-ESP8266连接阿里云物联网平台使用物理模型Topic实现温湿度数据显示](#)

[4.13-阿里云物联网平台篇-阿里云物联网平台加入规则引擎\(云产品流转\),让MQTT设备之间实现通信](#)

[4.14-阿里云物联网平台篇-Android和ESP8266连接阿里云物联网平台,并通过云平台实现远程温湿度采集和继电器控制](#)

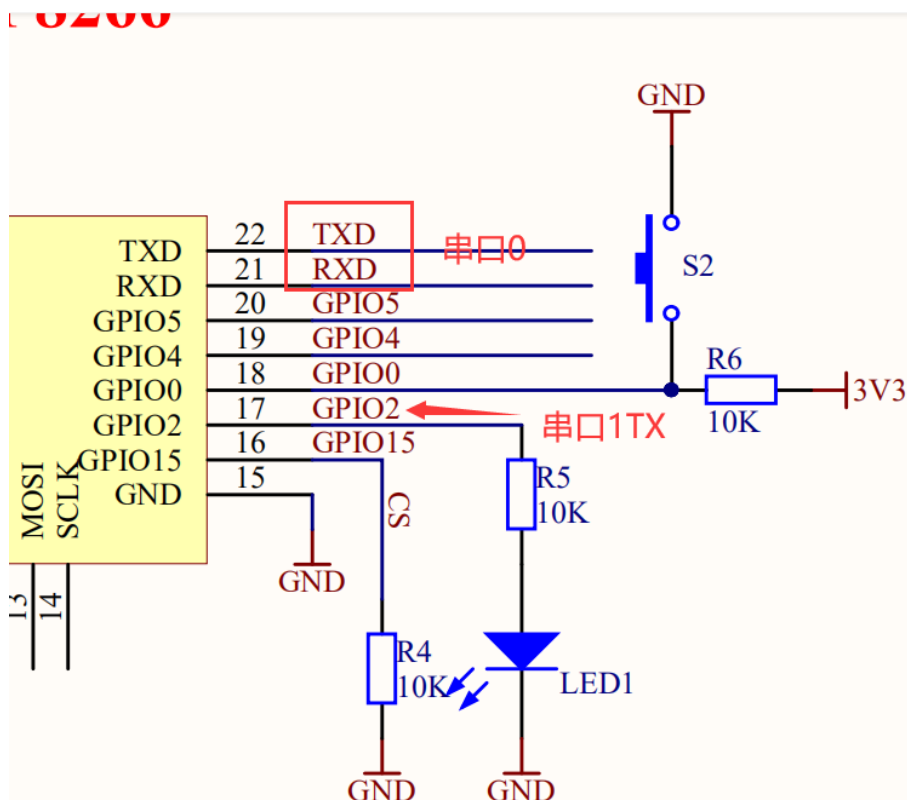
[6.01-综合实战篇-C#上位机串口通信控制ESP8266\(RTOS 2.2.0\)](#)

[6.02-综合实战篇-8266TCP服务器\(LWIP,RAW模式,PCB控制块实现\)\(RTOS 2.2.0\)与C#TCP客户端实现无线网络通信控制](#)

[6.03-综合实战篇-8266TCP服务器\(espconn实现\)\(NONOS 2.2.0\)与Android TCP客户端实现无线网络通信控制](#)

[9.01-常见问题及程序BUG修复](#)

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

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服务端+esp8266+单片机+路由器之远程控制系统(31338)

推荐排行榜

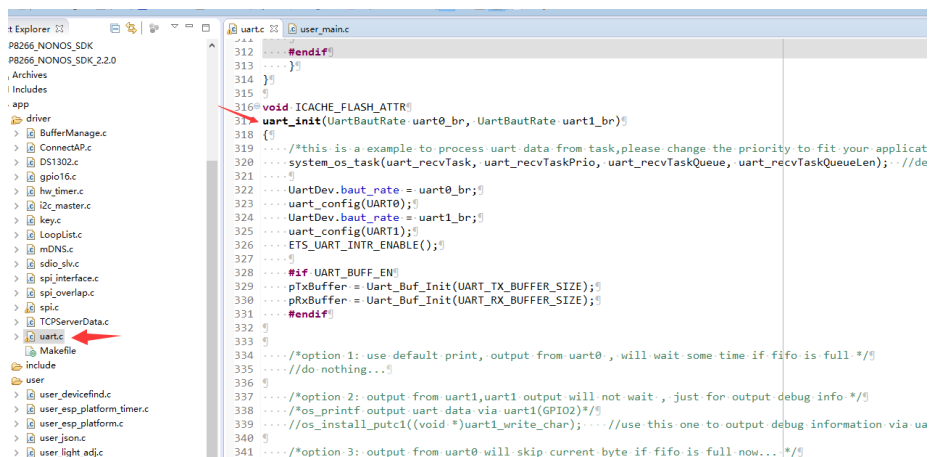
1. C#委托+回调详解(9)
2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(8)
3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
4. ESP8266使用详解(AT,LUA,SDK)(6)
5. 关于TCP和MQTT之间的转换(5)

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



```
114  ... #endif
115  ... }
116  ... }
117  ... }
118  ... }
119  ... }
120  ... }
121  ... }
122  ... }
123  ... }
124  ... }
125  ... }
126  ... }
127  ... }
128  ... }
129  ... }
130  ... }
131  ... }
132  ... }
133  ... }
134  ... }
135  ... }
136  ... }
137  ... }
138  ... }
139  ... }
140  ... }
141  ... }
142  ... }
143  ... }
144  ... }
145  ... }
146  ... }
147  ... }
148  ... }
149  ... }
150  ... }
151  ... }
152  ... }
153  ... }
154  ... }
155  ... }
156  ... }
157  ... }
158  ... }
159  ... }
160  ... }
161  ... }
162  ... }
163  ... }
164  ... }
165  ... }
166  ... }
167  ... }
168  ... }
169  ... }
170  ... }
171  ... }
172  ... }
173  ... }
174  ... }
175  ... }
176  ... }
177  ... }
178  ... }
179  ... }
180  ... }
181  ... }
182  ... }
183  ... }
184  ... }
185  ... }
186  ... }
187  ... }
188  ... }
189  ... }
190  ... }
191  ... }
192  ... }
193  ... }
194  ... }
195  ... }
196  ... }
197  ... }
198  ... }
199  ... }
200  ... }
201  ... }
202  ... }
203  ... }
204  ... }
205  ... }
206  ... }
207  ... }
208  ... }
209  ... }
210  ... }
211  ... }
212  ... }
213  ... }
214  ... }
215  ... }
216  ... }
217  ... }
218  ... }
219  ... }
220  ... }
221  ... }
222  ... }
223  ... }
224  ... }
225  ... }
226  ... }
227  ... }
228  ... }
229  ... }
230  ... }
231  ... }
232  ... }
233  ... }
234  ... }
235  ... }
236  ... }
237  ... }
238  ... }
239  ... }
240  ... }
241  ... }
242  ... }
243  ... }
244  ... }
245  ... }
246  ... }
247  ... }
248  ... }
249  ... }
250  ... }
251  ... }
252  ... }
253  ... }
254  ... }
255  ... }
256  ... }
257  ... }
258  ... }
259  ... }
260  ... }
261  ... }
262  ... }
263  ... }
264  ... }
265  ... }
266  ... }
267  ... }
268  ... }
269  ... }
270  ... }
271  ... }
272  ... }
273  ... }
274  ... }
275  ... }
276  ... }
277  ... }
278  ... }
279  ... }
280  ... }
281  ... }
282  ... }
283  ... }
284  ... }
285  ... }
286  ... }
287  ... }
288  ... }
289  ... }
290  ... }
291  ... }
292  ... }
293  ... }
294  ... }
295  ... }
296  ... }
297  ... }
298  ... }
299  ... }
300  ... }
301  ... }
302  ... }
303  ... }
304  ... }
305  ... }
306  ... }
307  ... }
308  ... }
309  ... }
310  ... }
311  ... }
312  ... }
313  ... }
314  ... }
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 applicat
320  ... system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen); //de
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  ... }
334  ... /*option 1: use default print, output from uart0 , will wait some time if fifo is full */
335  ... //do nothing...
336  ... }
337  ... /*option 2: output from uart1,uart1 output will not wait , just for output debug info */
338  ... /*os_printf output uart data via uart1(GPI02)*/
339  ... //os_install_putci((void *)uart1_write_char);... //use this one to output debug information via ua
340  ... }
341  ... /*option 3: output from uart0 will skip current byte if fifo is full now... */
```

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

```
uartc user_main.c
756
757
758 void ICACHE_FLASH_ATTR
759 UART_SetPrintPort(uint8 uart_no)
760 {
761     if(uart_no==1){
762         os_install_putc1(uart1_write_char);
763     }else{
764         /*option 1: do not wait if uart fifo is full, drop current character*/
765         os_install_putc1(uart0_write_char_no_wait);
766         /*option 2: wait for a while if uart fifo is full*/
767         os_install_putc1(uart0_write_char);
768     }
769 }
770
771
772 //=====
773
774 /*test code*/
775 void ICACHE_FLASH_ATTR
776 uart_init_2(UartBaudRate uart0_br, UartBaudRate uart1_br)
777 {
778     UartDev.baud_rate = uart0_br;
779     //UartDev.exist_parity = STICK_PARITY_EN;
780     UartDev.parity = NONE_BITS; //无奇偶校验
781     UartDev.stop_bits = ONE_STOP_BIT; //1位停止位
782     UartDev.data_bits = EIGHT_BITS; //8位数据
783
784     uart_config(UART0);
785     UartDev.baud_rate = uart1_br;
786     uart_config(UART1);
787     ETS_UART_INTR_ENABLE();
788
789     //install uart1 putc callback
790     //os_install_putc1((void *)uart1_write_char); //os_printf使用串口1打印(GPIO2)
791
792     os_install_putc1((void *)uart0_write_char); //os_printf使用串口0打印
793 }
794
795
796
```

```
uartc user_main.c
24 * ESPRESSIF MIT License
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;
36 user_rf_cal_sector_set(void)
37 void ICACHE_FLASH_ATTR
38 user_rf_pre_init(void){}
39
40 /**
41  * FunctionName: user_init
42  * Description: entry of user application, init user function
43  * Parameters: none
44  * Returns: none
45  */
46 void ICACHE_FLASH_ATTR user_init(void){
47     uart_init(BIT_RATE_115200, BIT_RATE_115200);
48 }
49
50
```

2.默认是使用串口0输出日志

咱先用修改,咱先把串口基本操作学完

```
uartc  user_main.c
315
316 void ICACHE_FLASH_ATTR
317 uart_init(UartBaudRate uart0_br, UartBaudRate uart1_br)
318 {
319     /*this is a example to process uart data from task, please change the priority to fit your application.
320     system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen); //demo with
321     */
322     UartDev.baud_rate = uart0_br;
323     uart_config(UART0);
324     UartDev.baud_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
```

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))

```
uartc  user_main.c
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 your
41 /*it might conflict with your task, if so, please arrange the priority of different task.
42 #define uart_recvTaskPrio 0 //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
```

```
uartc  user_main.c
308 //already move uart buffer output to uart empty interrupt
309 //tx_start_uart_buffer(UART0);
310 #else
311
312 #endif
313 }
314
315
316 void ICACHE_FLASH_ATTR
317 uart_init(UartBaudRate uart0_br, UartBaudRate uart1_br)
318 {
319     /*this is a example to process uart data from task, please change the priority to fit your application task if exists*/
320     system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen); //demo with a task to process the uart data
321
322     UartDev.baud_rate = uart0_br;
323     uart_config(UART0);
324     UartDev.baud_rate = uart1_br;
325     uart_config(UART1);

```

在内部FIFO接收到数据的时候发送任务消息出去

```
uartc  user_mainc  uartc
218  /* Returns .....: NONE */
219  *****
220  LOCAL void
221  uart0_rx_intr_handler(void *para)
222  {
223  /* uart0 and uart1 intr. combine together, when interrupt occur, see reg 0x3ff20020, bit2, bit0 represents */
224  /* uart1 and uart0 respectively */
225  /* */
226  uint8 RcvChar;
227  uint8 uart_no = UART0; //UartDev.buff_uart_no;
228  uint8 fifo_len = 0;
229  uint8 buf_idx = 0;
230  uint8 temp_cnt;
231  //RcvMsgBuff *pRxBuff = (RcvMsgBuff *)para;
232  /*
233  /*ATTENTION:*/
234  /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
235  /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM */
236  /*IF NOT, POST AN EVENT AND PROCESS IN SYSTEM TASK */
237  if(UART_FRM_ERR_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_FRM_ERR_INT_ST)){//接收帧错误
238  DBG("FRM_ERR\n");
239  WRITE_PERI_REG(UART_INT_CLR(uart_no), UART_FRM_ERR_INT_CLR);
240  }else if(UART_RXFIFO_FULL_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_RXFIFO_FULL_INT_ST)){//FIFO满中断
241  DBG("f");
242  uart_rx_intr_disable(UART0);
243  WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR);
244  system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
245  }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
246  DBG("t");
247  uart_rx_intr_disable(UART0);
248  WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_TOUT_INT_CLR);
249  system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
250  }else if(UART_TXFIFO_EMPTY_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_TXFIFO_EMPTY_INT_ST)){//发送空闲
251  DBG("e");
252  /* to output uart data from uart buffer directly in empty interrupt handler */
253  /*instead of processing in system event, in order not to wait for current task/function to quit */
254  /*ATTENTION:*/
255  /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
256  /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM */
257  CLEAR_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);
258  #if UART_BUFF_EN
259  }
```

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)

```
uartc  user_mainc  uartc
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 //
290  uart_recvTask(os_event_t *events)
291  {
292  if(events->sig == 0){
293  #if UART_BUFF_EN //使能接收到缓存里面
294  Uart_rx_buff_enq();
295  #else
296  uint8 fifo_len = (READ_PERI_REG(UART_STATUS(UART0)) >> UART_RXFIFO_CNT_S) & UART_RXFIFO_CNT;
297  uint8 d_tmp = 0;
298  uint8 idx = 0;
299  for(idx=0; idx<fifo_len; idx++){
300  d_tmp = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;
301  uart_tx_one_char(UART0, d_tmp);
302  }
303  WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR|UART_RXFIFO_TOUT_INT_CLR);
304  uart_rx_intr_enable(UART0);
305  #endif
306  }else if(events->sig == 1){
307  #if UART_BUFF_EN
308  //already move uart buffer output to uart empty interrupt
309  //tx_start_uart_buffer(UART0);
310  #else
311  }
312  #endif
313  }
314  }
```

```

507 //move data from uart_fifo to rx buffer
508 void Uart_rx_buff_enq()
509 {
510     uint8 fifo_len,buf_idx;
511     uint8 fifo_data;
512     #if 1
513     //从FIFO中获取串口接收的数据个数
514     fifo_len = (READ_PERI_REG(UART_STATUS(UART0))>>UART_RXFIFO_CNT_S)&UART_RXFIFO_CNT;
515     if(fifo_len>= pRxBuffer->Space){
516         os_printf("buf full!!!\n\r");
517     }else{
518         buf_idx=0;
519         while(buf_idx< fifo_len){
520             buf_idx++;
521             fifo_data = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;//读取数据
522             *(pRxBuffer->pInPos++) = fifo_data;
523             if(pRxBuffer->pInPos == (pRxBuffer->pUartBuff + pRxBuffer->UartBuffSize)){
524                 pRxBuffer->pInPos = pRxBuffer->pUartBuff;
525             }
526         }
527         pRxBuffer->Space -= fifo_len;
528         if(pRxBuffer->Space >= UART_FIFO_LEN){
529             //os_printf("after rx enq buf enough\n\r");
530             uart_rx_intr_enable(UART0);
531         }
532     }
533     #endif
534 }

```

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

```

464 os_free(pBuff->pUartBuff);
465 os_free(pBuff);
466 }
467
468 //rx buffer dequeue
470 uint16 ICACHE_FLASH_ATTR
471 rx_buff_deq(char* pdata, uint16 data_len)
472 {
473     uint16 buf_len = (pRxBuffer->UartBuffSize- pRxBuffer->Space);
474     uint16 tail_len = pRxBuffer->pUartBuff + pRxBuffer->UartBuffSize - pRxBuffer->pOutPos;
475     uint16 len_tmp = 0;
476     len_tmp = ((data_len > buf_len)?buf_len:data_len);
477     if(pRxBuffer->pOutPos <= pRxBuffer->pInPos){
478         os_memcpy(pdata, pRxBuffer->pOutPos, len_tmp);
479         pRxBuffer->pOutPos += len_tmp;
480         pRxBuffer->Space += len_tmp;
481     }else{
482         if(len_tmp>tail_len){
483             os_memcpy(pdata, pRxBuffer->pOutPos, tail_len);
484             pRxBuffer->pOutPos += tail_len;
485             pRxBuffer->pOutPos = (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos - pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize);
486             pRxBuffer->Space += tail_len;
487         }
488         os_memcpy(pdata+tail_len, pRxBuffer->pOutPos, len_tmp-tail_len);
489         pRxBuffer->pOutPos += (len_tmp-tail_len);
490         pRxBuffer->pOutPos = (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos - pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize);
491         pRxBuffer->Space += (len_tmp-tail_len);
492     }
493     //os_printf("case 3 in rx deq\n\r");
494     os_memcpy(pdata, pRxBuffer->pOutPos, len_tmp);
495     pRxBuffer->pOutPos += len_tmp;
496     pRxBuffer->pOutPos = (pRxBuffer->pUartBuff + (pRxBuffer->pOutPos - pRxBuffer->pUartBuff) % pRxBuffer->UartBuffSize);
497     pRxBuffer->Space += len_tmp;
498 }
499
500 if(pRxBuffer->Space >= UART_FIFO_LEN){
501     uart_rx_intr_enable(UART0);
502 }
503 return len_tmp;
504 }

```

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


```

315 }
316 void ICACHE_FLASH_ATTR
317 uart_init(UartBaudRate uart0_br, UartBaudRate 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 的, 保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

```

```

407
270 /*
271  * FunctionName : uart_init
272  * Description : user interface for init uart
273  * Parameters : UartBaudRate uart0_br -- uart0 baudrate
274  *             UartBaudRate uart1_br -- uart1 baudrate
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

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

```
uartc user_main.c uarth osapi.h
24 * ESPRESSIF MIT License
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 //set 1: enable uart hw flow control 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.

```
uartc user_main.c uarth 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_t priv_param_start_sec;
37 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_t uart_buf[128]={0};
97     uint16_t 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  uarth  osapi.h
24  * ESPRESSIF MIT License
25
26 #include "ets_sys.h"
27 #include "osapi.h"
28 #include "driver/uart.h"
29 #include "osapi.h"
30 #include "driver/uart_register.h"
31 #include "mem.h"
32 #include "os_type.h"
33
34 // UartDev is defined and initialized in rom code
35 extern UartDevice UartDev;
36
37 LOCAL struct UartBuffer* pTxBuffer = NULL;
38 LOCAL struct UartBuffer* pRxBuffer = NULL;
39
40 /*uart demo with a system task, to output what uart receives*/
41 /*this is a example to process uart data from task, please change the priority to fit yo
42 /*it might conflict with your task, if so, please arrange the priority of different task
43 #define uart_recvTaskPrio 2 //任务等级
44 #define uart_recvTaskQueueLen 10 //任务队列大小
45 os_event_t uart_recvTaskQueue[uart_recvTaskQueueLen]; //使用这个数组作为存储任务消息
46
47 #define DBG
48 #define DBG1 uart1_sendStr_no_wait
49 #define DBG2 os_printf
50
51

```

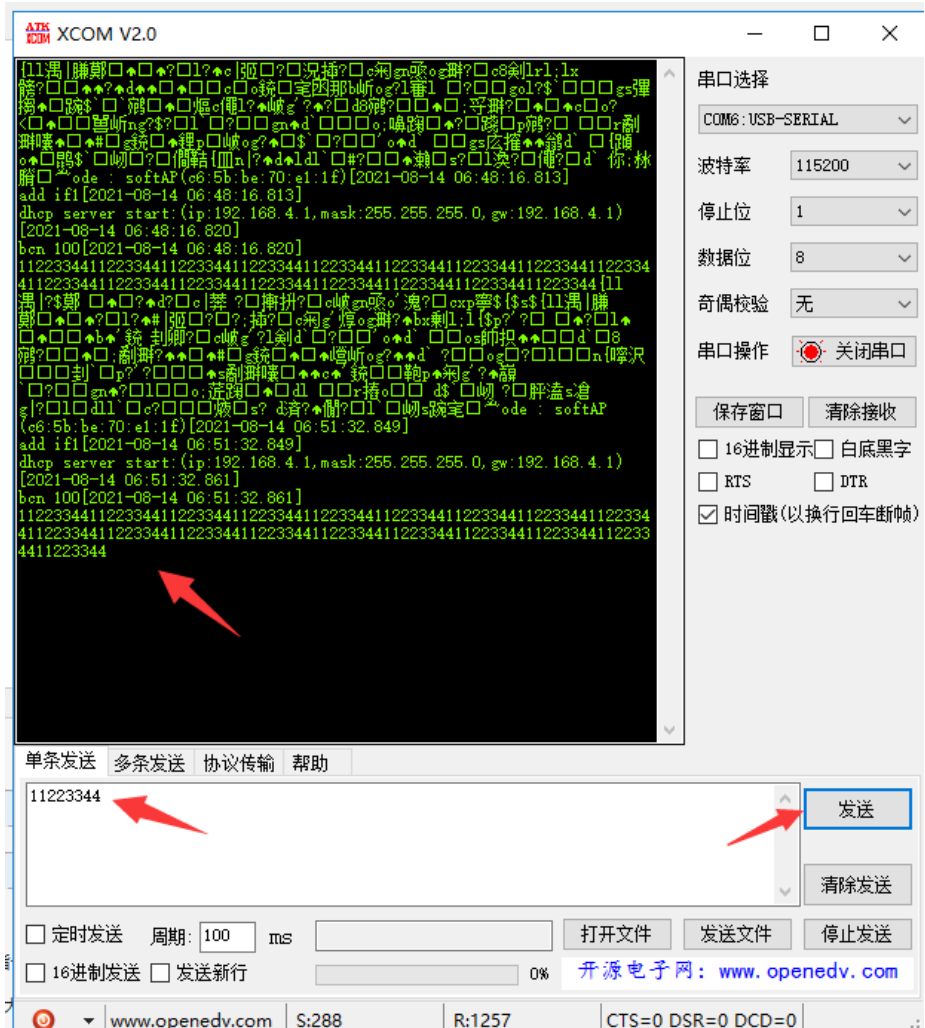
然后在主函数加一个优先级别低的任务

```

uart.c  user_main.c  uarth  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
41 /******串口接收缓存******/
42 uint8 uart_buf[UART_RX_BUFFER_SIZE]={0};
43 uint16 len = 0;
44
45 uint32 priv_param_start_sec;
46 #user_rf_cal_sector_set(void)
47 #void ICACHE_FLASH_ATTR
48 user_rf_pre_init(void){}
49
50
51 void os_task_t_callback(os_event_t *events){
52     if(events->sig == 0 && events->par == 0){
53         system_os_post(TaskPrio, 0, 0);
54     }
55     os_task_t_delay++;
56     if(os_task_t_delay>300){ //大约1ms
57         os_task_t_delay=0;
58         len = rx_buff_deq(uart_buf, 128); //尝试从缓存中获取128字节串口数据
59         tx_buff_eng(uart_buf, len); //把接收的数据送回
60     }
61 }
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570

```


◀ ▶



当然一般串口10ms已经适应了基本上所有的项目了

```
43
44 uint32_t 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 > 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
```

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

```
uartc user_main.c uarth 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 0 //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_MTD0, FOR UART0
39 #define UART_HW_CTS 0 //set 1: enable uart hw flow control CTS, PIN_MTD0, FOR UART0
40
```

```
uartc user_main.c uarth osapi.h
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_t uart_buf[128] = {0};
283         ... uint16_t 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 //
290     uart_recvTask(os_event_t *events)
291     {
292         ... if(events->sig == 0){
293             ... #if UART_BUFF_EN //使能接收到缓存里面
294                 ... Uart_rx_buff_enq();
295             ... #else
296                 ... uint8_t fifo_len = (READ_PERI_REG(UART_STATUS(UART0)) >> UART_RXFIFO_CNT_S) & UART_RXFIFO_CNT;
297                 ... uint8_t d_tmp = 0;
298                 ... uint8_t idx = 0;
299                 ... for(idx = 0; idx < fifo_len; idx++){
300                     ... d_tmp = READ_PERI_REG(UART_FIFO(UART0)) & 0xFF;
301                     ... uart_tx_one_char(UART0, d_tmp);
302                 }
303                 ... WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR | UART_RXFIFO_TOUT_INT_CLR);
304                 ... uart_rx_intr_enable(UART0);
305             ... #endif
306         } ... else if(events->sig == 1){
307             ... #if UART_BUFF_EN
308                 ... //already move uart buffer output to uart empty interrupt
309                 ... //tx_start_uart_buffer(UART0);
310             ... #else
311                 ...

```

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\nnull, create buffer struct\n\n");
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\n");
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\n");
556         }
557     }
558     #if 0
559     if(pTxBuffer->Space <= UART_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
```

把数据储存在缓存

使能中断发送

```

uartc  user_mainc  uarth  osapih
229 .....uint8 buf_idx = 0;
230 .....uint8 temp_cnt;
231 .....//RcvMsgBuff *pRxBuff = (RcvMsgBuff *)para;
232 .....
233 ...../*ATTENTION:*/
234 » /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
235 » /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM.*/
236 » /*IF NOT, POST AN EVENT AND PROCESS IN SYSTEM TASK.*/
237 .....if(UART_FRM_ERR_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_FRM_ERR_INT_ST)){//接收帧错误
238 .....DBG1("FRM_ERR\r\n");
239 .....WRITE_PERI_REG(UART_INT_CLR(uart_no), UART_FRM_ERR_INT_CLR);
240 .....}else if(UART_RXFIFO_FULL_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_RXFIFO_FULL_INT_ST)){//FIFO满中断
241 .....DBG1("f");
242 .....uart_rx_intr_disable(UART0);
243 .....WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_FULL_INT_CLR);
244 .....system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
245 .....}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
246 .....DBG1("t");
247 .....uart_rx_intr_disable(UART0);
248 .....WRITE_PERI_REG(UART_INT_CLR(UART0), UART_RXFIFO_TOUT_INT_CLR);
249 .....system_os_post(uart_recvTaskPrio, 0, 0);//发送通知
250 .....}else if(UART_TXFIFO_EMPTY_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_TXFIFO_EMPTY_INT_ST)){//发送空闲
251 .....DBG1("e");
252 » /*to output uart data from uart buffer directly in empty interrupt handler*/
253 » /*instead of processing in system event, in order not to wait for current task/function to quit.*/
254 » /*ATTENTION:*/
255 » /*IN NON-OS VERSION SDK, DO NOT USE "ICACHE_FLASH_ATTR" FUNCTIONS IN THE WHOLE HANDLER PROCESS*/
256 » /*ALL THE FUNCTIONS CALLED IN INTERRUPT HANDLER MUST BE DECLARED IN RAM.*/
257 .....CLEAR_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);
258 .....#if UART_BUFF_EN
259 .....» tx_start_uart_buffer(UART0);
260 .....#endif
261 .....//system_os_post(uart_recvTaskPrio, 1, 0);
262 .....WRITE_PERI_REG(UART_INT_CLR(uart_no), UART_TXFIFO_EMPTY_INT_CLR);
263 .....
264 .....}else if(UART_RXFIFO_OVF_INT_ST == (READ_PERI_REG(UART_INT_ST(uart_no)) & UART_RXFIFO_OVF_INT_ST)){
265 .....WRITE_PERI_REG(UART_INT_CLR(uart_no), UART_RXFIFO_OVF_INT_CLR);
266 .....DBG1("RX_OVF!!\r\n");
267 .....}
268 .....}
269 .....
89 *****/
90 void tx_start_uart_buffer(uint8 uart_no)
91 {
92 .....uint8 tx_fifo_len = (READ_PERI_REG(UART_STATUS(uart_no)) >> UART_TXFIFO_CNT_S) & UART_TXFIFO_CNT;
93 .....uint8 fifo_remain = UART_FIFO_LEN - tx_fifo_len;
94 .....uint8 len_tmp;
95 .....uint16 tail_ptx_len, head_ptx_len, data_len;
96 .....//struct UartBuffer *pTxBuff = *get_buff_ptr();
97 .....
98 .....if(pTxBuffer){
99 .....» data_len = (pTxBuffer->UartBuffSize - pTxBuffer->Space);
100 .....» if(data_len > fifo_remain){
101 .....» len_tmp = fifo_remain;
102 .....» tx_fifo_insert(pTxBuffer, len_tmp, uart_no);
103 .....» SET_PERI_REG_MASK(UART_INT_ENA(UART0), UART_TXFIFO_EMPTY_INT_ENA);
104 .....» }else{
105 .....» len_tmp = data_len;
106 .....» tx_fifo_insert(pTxBuffer, len_tmp, uart_no);
107 .....» }
108 .....}else{
109 .....» DBG1("pTxBuff: null \n\r");
110 .....}
111 .....}
112 .....
113 #endif

3 //-----
3 LOCAL void tx_fifo_insert(struct UartBuffer *pTxBuff, uint8 data_len, uint8 uart_no)
1 {
2 .....uint8 i;
3 .....for(i = 0; i < data_len; i++){
1 .....» WRITE_PERI_REG(UART_FIFO(uart_no), *(pTxBuff->pOutPos++));
5 .....» if(pTxBuff->pOutPos == (pTxBuff->pUartBuff + pTxBuff->UartBuffSize)){
7 .....» pTxBuff->pOutPos = pTxBuff->pUartBuff;
3 .....» }
3 .....» pTxBuff->pOutPos = (pTxBuff->pUartBuff + (pTxBuff->pOutPos - pTxBuff->pUartBuff) % pTxBuff->UartBuffSize);
3 .....» pTxBuff->Space += data_len;
1 .....}
2 .....}

```

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)


```

316@ void ICACHE_FLASH_ATTR
317 uart_init(UartBaudRate uart0_br, UartBaudRate uart1_br)
318 {
319     /*this is a example to process uart data from task, please change the priority to fit your applica
320     system_os_task(uart_recvTask, uart_recvTaskPrio, uart_recvTaskQueue, uart_recvTaskQueueLen);...//d
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(GPI02)*/
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 buf
351 os_timer_arm(&buff_timer_t, 10, 1);
352 #endif
353 }
354 }
355@ 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

```

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 {
119     uart_init(BIT_RATE_115200, BIT_RATE_115200);
120     system_set_os_print(0); //0:关闭打印 1:开启打印
121     system_os_task(os_task_t_callback, TaskPrio, os_event_t_buff, os_event_t_buff_len);
122     system_os_post(TaskPrio, 0, 0);
123 }
124

```

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

分类: ESP8266 SDK开发

好文要顶

关注我

收藏该文



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](#)

B    

支持 Markdown

 自动补全

[提交评论](#) [退出](#)

[Ctrl+Enter快捷键提交]

【推荐】百度智能云2021普惠上云节：新用户首购云服务器低至0.7折

【推荐】阿里云云大使特惠：新用户购ECS服务器1核2G最低价87元/年

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

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...

Powered by:

博客园

Copyright © 2021 杨奉武

Powered by .NET 5.0 on Kubernetes



单片机,物联网,上位机,...

扫一扫二维码, 入群聊。