

## 优秀不够，你是否无可替代

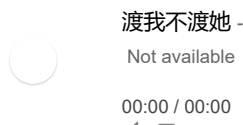
知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人  
QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 748, 文章 - 0, 评论 - 315, 阅读 - 182万

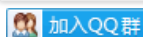
### 导航

博客园  
首页  
新随笔  
联系  
订阅 8ML  
管理

### 公告



- 1 渡我不渡她
- 2 小镇姑娘
- 3 PDD洪荒之力



昵称：杨奉武  
园龄：5年10个月  
粉丝：633  
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

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 LUA开发基础入门篇备份(22)

## 105-ESP32\_SDK开发-串口,485通信

<p><iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnESP32" frameborder="0" scrolling="auto" width="100%" height="1500"></iframe></p>

## 开源ESP32开发(源码见资料源码)

测试板链接:[ESP32测试板链接](#)

资料源码:<https://github.com/yangfengwu45/learn-esp32.git>

【点击加入乐鑫WiFi模组开发交流群】(群号822685419)[https://jq.qq.com/?\\_wv=1027&k=fXgd3UOo](https://jq.qq.com/?_wv=1027&k=fXgd3UOo)

python虚拟机: [python-3.8.4-amd64.exe](#)

ESP-IDF工具安装器: [esp-idf-tools-setup-2.3.exe](#)

- 基础开源教程:ESP32开发(arduino)
  - 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。

- [000-ESP32开发板使用说明](#)
- [ESP32\\_SDK开发](#)
- [001-开发环境搭建\(Windows+VSCode\)](#)
- [002-测试网络摄像头\(OV2640\),实现远程视频监控\(花生壳http映射\)](#)
- [003-学习ESP32资料说明](#)
- [004-新建工程模板和创建新的文件](#)
- [005-新建工程补充-通过官方示例创建工程](#)
- [006-关于操作系统-任务,任务堆栈空间,任务的挂起,恢复,删除](#)
- [007-使用缓存管理传递数据](#)
- 基本外设-----
- [101-ESP32管脚说明](#)
- [102-GPIO](#)
- [103-硬件定时器timer](#)
- [104-软件定时器esp\\_timer](#)
- [105-uart串口,485通信](#)
- 
- 
- 
-

ESP8266 SDK开发(32)  
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)  
软件安装使用: OpenResty(6)  
更多

## 最新评论

1. Re:单片机模块化程序: 看  
看是不是你想要的按键处理  
视频不见了  
--伊森亨特
2. Re:C#开发: 通信篇-TCP客  
户端  
感谢分享，直接就用上了  
--Zfen

## 阅读排行榜

1. ESP8266使用详解(AT,LUA,  
SDK)(172765)
2. 1-安装MQTT服务器(Windo  
ws),并连接测试(98876)
3. ESP8266刷AT固件与node  
mcu固件(64682)
4. 用ESP8266+android,制作  
自己的WIFI小车(ESP8266篇)  
(64233)
5. 有人WIFI模块使用详解(385  
14)

## 说明

模块有3个串口,每个串口管脚可以设置到任意的gpio上

模组出厂默认使用GPIO1,GPIO3作为串口0引脚(日志打印);  
GPIO17,GPIO16作为串口1引脚(AT指令)

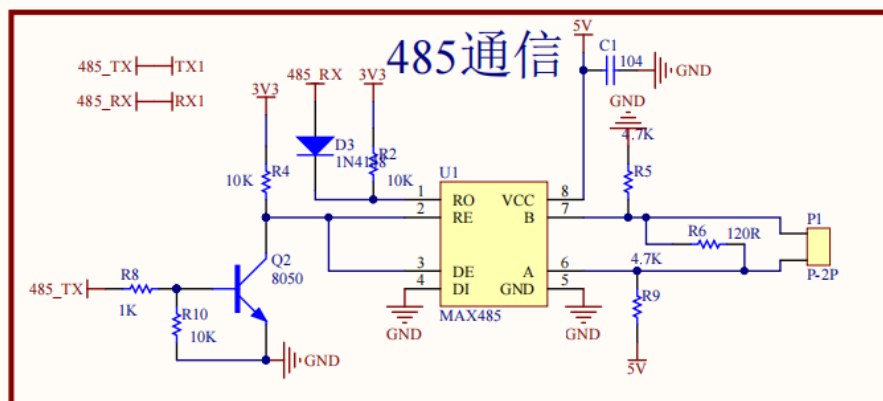
	GND	38		GND
	GPIO23/VSPID/HS1_STROBE	37	CAM_HREF	IO23
	GPIO22/VSPWP/U0RTS/EMAC_TXD1	36	CAM_PCLK	IO22
	GPIO1/USART0_TX/CLK_OUT3/EMAC_RXD2	35		TX0
	GPIO3/USART0_RXD/CLK_OUT2	34		RX0
右方引脚	GPIO21/VSPHD/EMAC_TX_EN	33	CAM_D3	IO21
	NC	32		NC
	GPIO19/VSPHQ/U0CTS/EMAC_TXD0	31	CAM_D2	IO19
	GPIO18/VSPICLK/HS1_DATA7	30	CAM_D1	IO18
	GPIO5/VSPICS0/HS1_DATA6/EMAC_RX_CLK	29	CAM_D0	IO5
	GPIO17/HS1_DATA5/USART2_TXD/EMAC_CLK_OUT_180	28		TX1
	GPIO16/HS1_DATA4/USART2_RXD/EMAC_CLK_OUT	27		RX1
	IO/RTC_GPIO10/HSPIHD/HS2_DATA1/SD_DATA1/EMAC_TX_ER	26	RELAY	IO4
	ADC2_CH1/TOUCH1/RTC_GPIO11/CLK_OUT1/EMAC_TX_CLK	25	CAM_RST	IO0

开发板上也把串口1连接了485上.

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)(35927)
7. 关于TCP和MQTT之间的转换(33212)
8. C#中public与private与static(32431)
9. android 之TCP客户端编程(31925)
10. android服务端+esp8266+单片机+路由器之远程控制系统(31321)

#### 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)



## 说明2

每个串口都有一个128字节的FIFO缓存区,知道这个就可以.

```
C hello_world_main.c 2, M  C uart_caps.h X  C esp_timer.h  C esp_intr_alloc.h  C timer.c
H: > LearnESP32 > esp-if > components > soc > soc > esp32 > include > soc > C uart_caps.h > UART_FIFO_LEN

1 // Copyright 2010-2019 Espressif Systems (Shanghai) PTE LTD
2 //
3 // Licensed under the Apache License, Version 2.0 (the "License");
4 // you may not use this file except in compliance with the License.
5 // You may obtain a copy of the License at
6
7 // http://www.apache.org/licenses/LICENSE-2.0
8 //
9 // Unless required by applicable law or agreed to in writing, software
10 // distributed under the License is distributed on an "AS IS" BASIS,
11 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 // See the License for the specific language governing permissions and
13 // limitations under the License.
14
15 #pragma once
16
17 #ifdef __cplusplus
18 extern "C" {
19 #endif
20
21 #define UART_FIFO_LEN (128) /*!< The UART hardware FIFO length */
22 #define UART_BITRATE_MAX (5000000) /*!< Max bit rate supported by UART */
23
```

设置串口1,带接收缓存,不带发送缓存区的方式  
(最简洁的方式)

设置GPIO17,GPIO16作为串口1引脚.

没有设置发送缓存,调用 `uart_write_bytes` 发送数据的时候是阻塞的.

```

6 #include "driver/timer.h"
7 #include "esp_timer.h"
8 #include "driver/uart.h"
9
10 #define TXD1_PIN (GPIO_NUM_17) //串口1的发送数据引脚
11 #define RXD1_PIN (GPIO_NUM_16) //串口1的接收数据引脚
12 #define BUF_SIZE (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN(128)
13 /*串口任务*/
14 static void uart_task(void *arg)
15 {
16     /*配置串口参数*/
17     uart_config_t uart_config = {
18         .baud_rate = 115200, //波特率
19         .data_bits = UART_DATA_8_BITS, //数据位8位
20         .parity = UART_PARITY_DISABLE, //无奇偶校验
21         .stop_bits = UART_STOP_BITS_1, //停止位1位
22         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
23         .source_clk = UART_SCLK_APB, //串口使用的时钟
24     };
25     /*初始化串口1*/
26     uart_driver_install(UART_NUM_1,
27         BUF_SIZE, //串口1接收缓存大小
28         0, //不使用发送缓存(发送数据的时候便会阻塞发送)
29         0, //队列大小为0;没有使用freertos内部缓存管理
30         NULL, //不使用QueueHandle_t 内部缓存管理,设置为空
31         0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
32     );
33     /*设置串口参数*/
34     uart_param_config(UART_NUM_1, &uart_config);
35     /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
36     uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
37     /*申请一块内存,用于临时存储接收的数据*/
38     uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
39     while (1) {
40         //接收串口数据 //每隔10ms判断一次,可以写成portMAX_DELAY(一直判断)
41         int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 10 / portTICK_RATE_MS);
42         //把接收的数据发送出去
43         uart_write_bytes(UART_NUM_1, (const char *) data, len);
44     }
45 }
46 void app_main(void)
47 {
48     xTaskCreate(uart_task, "uart_task", 2048, NULL, 10, NULL);
49 }

```



```

#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"
#include "driver/timer.h"
#include "esp_timer.h"
#include "driver/uart.h"

#define TXD1_PIN (GPIO_NUM_17) //串口1的发送数据引脚
#define RXD1_PIN (GPIO_NUM_16) //串口1的接收数据引脚
#define BUF_SIZE (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN
/*串口任务*/
static void uart_task(void *arg)
{
    /*配置串口参数*/
    uart_config_t uart_config = {
        .baud_rate = 115200, //波特率
        .data_bits = UART_DATA_8_BITS, //数据位8位
        .parity = UART_PARITY_DISABLE, //无奇偶校验
        .stop_bits = UART_STOP_BITS_1, //停止位1位
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
        .source_clk = UART_SCLK_APB, //串口使用的时钟
    };
    /*初始化串口1*/
    uart_driver_install(UART_NUM_1,
        BUF_SIZE, //串口1接收缓存大小
        0, //不使用发送缓存(发送数据的时候便会阻塞发送)

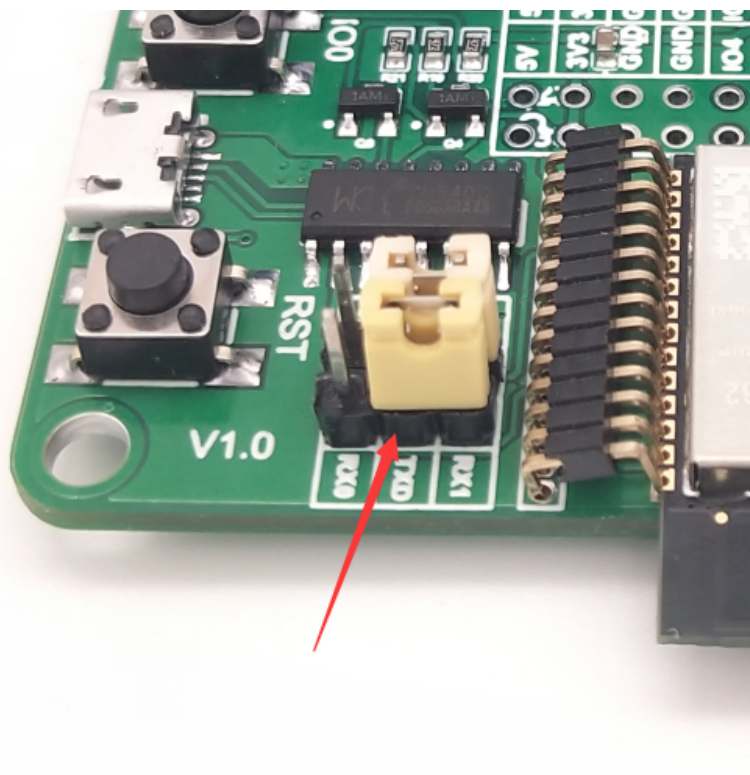
```

```

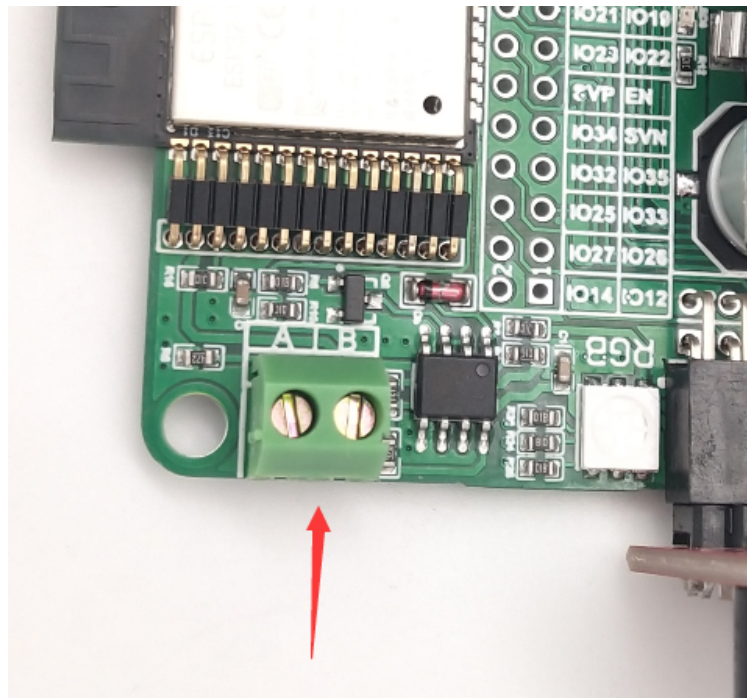
0, //队列大小为0;没有使用freertos内部缓存管理
NULL, //不使用QueueHandle_t 内部缓存管理,设置为空
0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
);
/*设置串口参数*/
uart_param_config(UART_NUM_1, &uart_config);
/*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_
/*申请一块内存,用于临时存储接收的数据*/
uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
while (1) {
    //接收串口数据 //每隔10ms判断一次
    int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 10 / portTICK_R
    //把接收的数据发送出去
    uart_write_bytes(UART_NUM_1, (const char *) data, len);
}
}
void app_main(void)
{
    xTaskCreate(uart_task, "uart_task", 2048, NULL, 10, NULL);
}

```

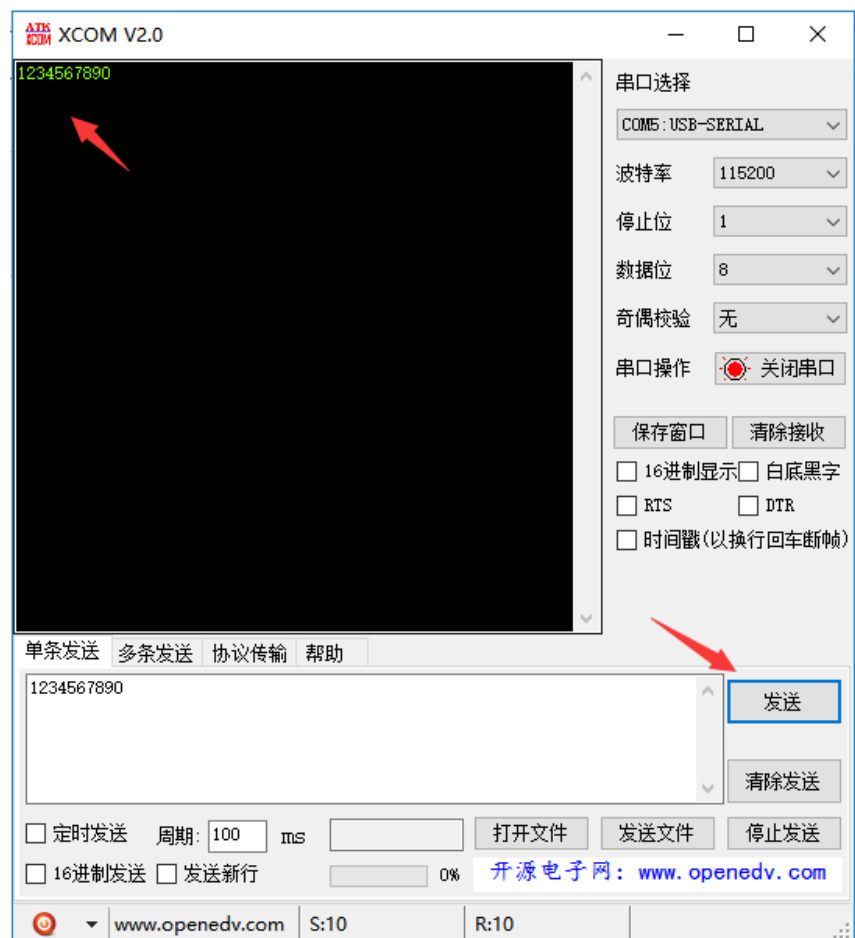
下载程序到开发板以后可以通过串口测试



485接口默认连接串口1,也可以使用485进行通讯



发送什么数据将会返回什么数据



设置串口1,带接收缓存,带发送缓存区的方式



设置上发送缓存区以后,调用 `uart_write_bytes` 发送数据的时候,将不会阻塞在那里.

```
18 #include "driver/uart.h"
19
20 #define TXD1_PIN (GPIO_NUM_17) //串口1的发送数据引脚
21 #define RXD1_PIN (GPIO_NUM_16) //串口1的接收数据引脚
22 #define BUF_SIZE (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN(128)
23
24 #define BUF_SEND_SIZE (1024) //发送数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN(128)
25
26 /*串口任务*/
27 static void uart_task(void *arg)
28 {
29     /*配置串口参数*/
30     uart_config_t uart_config = {
31         .baud_rate = 115200, //波特率
32         .data_bits = UART_DATA_8_BITS, //数据位8位
33         .parity = UART_PARITY_DISABLE, //无奇偶校验
34         .stop_bits = UART_STOP_BITS_1, //停止位1位
35         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
36         .source_clk = UART_SCLK_APB, //串口使用的时钟
37     };
38     /*初始化串口1*/
39     uart_driver_install(UART_NUM_1,
40         BUF_SIZE, //串口1接收缓存大小
41         BUF_SEND_SIZE, //串口1发送缓存大小
42         0, //队列大小为0;没有使用freertos内部缓存管理
43         NULL, //不使用QueueHandle_t 内部缓存管理,设置为空
44         0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
45     );
46     /*设置串口参数*/
47     uart_param_config(UART_NUM_1, &uart_config);
48     /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
49     uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
50     /*申请一块内存,用于临时存储接收的数据*/
51     uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
52     while (1) {
53         //接收串口数据 //每隔10ms判断一次,可以写成portMAX_DELAY(一直判断)
54         int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 10 / portTICK_RATE_MS);
55         //把接收的数据发送出去
56         uart_write_bytes(UART_NUM_1, (const char *) data, len);
57     }
58 }
59 void app_main(void)
60 {
61     xTaskCreate(uart_task, "uart_task", 2048, NULL, 10, NULL);
62 }
```

设置串口1,带接收缓存,带发送缓存区,并使用上freertos内部的缓存管理的方式

加上缓存管理

```

3
4 #define BUF_SEND_SIZE (1024) //发送数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN(128)
5
6
7 static QueueHandle_t QueueHandle_t_uart1;
8
9 /*串口任务*/
10 static void uart_task(void *arg)
11 {
12     /*配置串口参数*/
13     uart_config_t uart_config = {
14         .baud_rate = 115200, //波特率
15         .data_bits = UART_DATA_8_BITS, //数据位8位
16         .parity = UART_PARITY_DISABLE, //无奇偶校验
17         .stop_bits = UART_STOP_BITS_1, //停止位1位
18         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
19         .source_clk = UART_SCLK_APB, //串口使用的时钟
20     };
21     /*初始化串口1*/
22     uart_driver_install(UART_NUM_1,
23         BUF_SIZE, //串口1接收缓存大小
24         BUF_SEND_SIZE, //串口1发送缓存大小
25         10, //队列大小为10
26         &QueueHandle_t_uart1, //缓存管理
27         0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
28     );
29     /*设置串口参数*/
30     uart_param_config(UART_NUM_1, &uart_config);
31     /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
32     uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
33
34     /*申请一块内存,用于临时存储接收的数据*/
35     uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
36     uart_event_t event;
37     while (1) {
38         if(xQueueReceive(QueueHandle_t_uart1, (void *) &event, portMAX_DELAY))
39         {

```

## 从缓存管理中获取数据

```

main > C:\hello_world_main.c > uart_task(void *)
44     BUF_SEND_SIZE, //串口1发送缓存大小
45     10, //队列大小为10
46     &QueueHandle_t_uart1, //缓存管理
47     0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
48 );
49 /*设置串口参数*/
50 uart_param_config(UART_NUM_1, &uart_config);
51 /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
52 uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
53
54 /*申请一块内存,用于临时存储接收的数据*/
55 uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
56 uart_event_t event;
57 while (1) {
58     if(xQueueReceive(QueueHandle_t_uart1, (void *) &event, portMAX_DELAY))
59     {
60         switch(event.type) {
61             case UART_DATA: //接收到数据
62                 /*读取接收的数据*/
63                 uart_read_bytes(UART_NUM_1, data, event.size, portMAX_DELAY);
64                 /*返回接收的数据*/
65                 uart_write_bytes(UART_NUM_1, (const char *) data, event.size);
66                 break;
67             case UART_FIFO_OVF: //FIFO溢出(建议加上数据流控制)
68                 uart_flush_input(UART_NUM_1);
69                 xQueueReset(QueueHandle_t_uart1);
70                 break;
71             case UART_BUFFER_FULL: //接收缓存满(建议加大缓存 BUF_SIZE)
72                 uart_flush_input(UART_NUM_1);
73                 xQueueReset(QueueHandle_t_uart1);
74                 break;
75             case UART_BREAK: //检测到接收数据中断
76                 break;
77             case UART_PARITY_ERR: //数据校验错误
78                 break;
79             case UART_FRAME_ERR: //数据帧错误
80                 break;
81             case UART_PATTERN_DET: //接收到相匹配的字符(没用到)
82                 break;
83             default:
84                 break;
85         }
86     }
87 }
88 free(data);
89 data = NULL;
90 vTaskDelete(NULL);
91 }
92 void app_main(void)

```





```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"
#include "driver/timer.h"
#include "esp_timer.h"
#include "driver/uart.h"

#define TXD1_PIN (GPIO_NUM_17) //串口1的发送数据引脚
#define RXD1_PIN (GPIO_NUM_16) //串口1的接收数据引脚
#define BUF_SIZE (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN

#define BUF_SEND_SIZE (1024) //发送数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN

static QueueHandle_t QueueHandle_t_uart1;

/*串口任务*/
static void uart_task(void *arg)
{
    /*配置串口参数*/
    uart_config_t uart_config = {
        .baud_rate = 115200, //波特率
        .data_bits = UART_DATA_8_BITS, //数据位8位
        .parity = UART_PARITY_DISABLE, //无奇偶校验
        .stop_bits = UART_STOP_BITS_1, //停止位1位
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
        .source_clk = UART_SCLK_APB, //串口使用的时钟
    };
    /*初始化串口1*/
    uart_driver_install(UART_NUM_1,
        BUF_SIZE, //串口1接收缓存大小
        BUF_SEND_SIZE, //串口1发送缓存大小
        10, //队列大小为10
        &QueueHandle_t_uart1, //缓存管理
        0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
    );
    /*设置串口参数*/
    uart_param_config(UART_NUM_1, &uart_config);
    /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
    uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);

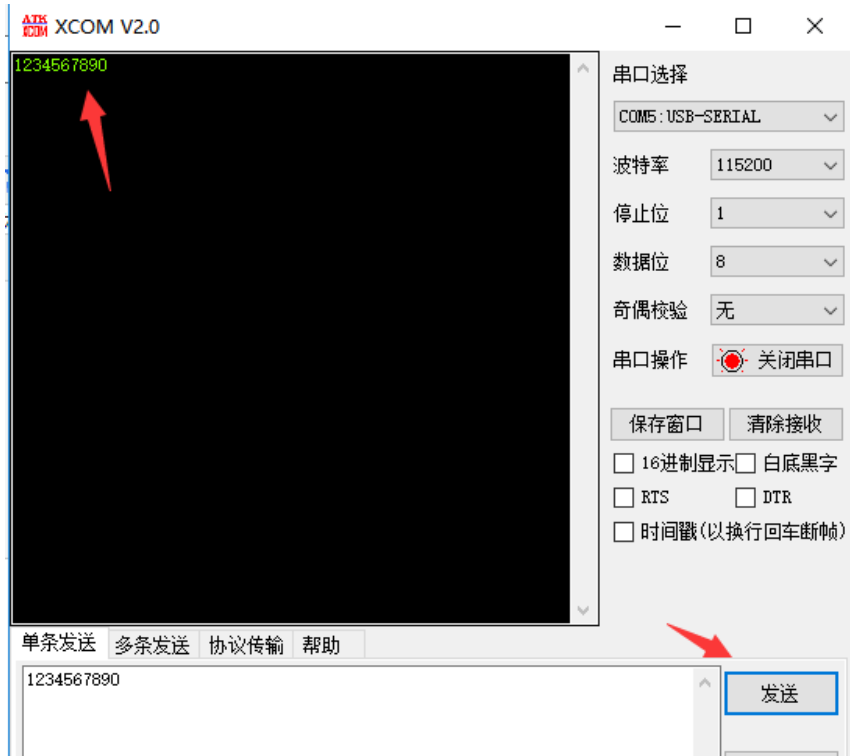
    /*申请一块内存,用于临时存储接收的数据*/
    uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
    uart_event_t event;
    while (1) {
        if(xQueueReceive(QueueHandle_t_uart1, (void *) &event, portMAX_DELAY) == pdPASS)
        {
            switch(event.type) {
                case UART_DATA: //接收到数据
                    //读取接收的数据
                    uart_read_bytes(UART_NUM_1, data, event.size, portMAX_DELAY);
                    //返回接收的数据
                    uart_write_bytes(UART_NUM_1, (const char *) data, event.size);
                    break;
                case UART_FIFO_OVF: //FIFO溢出(建议加上数据流控制)
```

```

        uart_flush_input(UART_NUM_1);
        xQueueReset(QueueHandle_t_uart1);
        break;
    case UART_BUFFER_FULL://接收缓存满(建议加大缓存 BUF_SIZE)
        uart_flush_input(UART_NUM_1);
        xQueueReset(QueueHandle_t_uart1);
        break;
    case UART_BREAK://检测到接收数据中断
        break;
    case UART_PARITY_ERR://数据校验错误
        break;
    case UART_FRAME_ERR://数据帧错误
        break;
    case UART_PATTERN_DET://接收到相匹配的字符(没用到)
        break;
    default:
        break;
    }
}
}
free(data);
data = NULL;
vTaskDelete(NULL);
}

void app_main(void)
{
    xTaskCreate(uart_task, "uart_task", 2048, NULL, 10, NULL);
}

```



**如果想配置串口0或者串口2**

把以下变量的最后一个数字改为0或者2即可

```
hello_world_main.c > uart_task(void *)
31 {
32     /*配置串口参数*/
33     uart_config_t uart_config = {
34         .baud_rate = 115200, //波特率
35         .data_bits = UART_DATA_8_BITS, //数据位8位
36         .parity = UART_PARITY_DISABLE, //无奇偶校验
37         .stop_bits = UART_STOP_BITS_1, //停止位1位
38         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE, //不使用硬件流控
39         .source_clk = UART_SCLK_APB, //串口使用的时钟
40     };
41     /*初始化串口1*/
42     uart_driver_install(UART_NUM_1,
43         BUF_SIZE, //串口1接收缓存大小
44         BUF_SEND_SIZE, //串口1发送缓存大小
45         10, //队列大小为10
46         &QueueHandle_t_uart1, //缓存管理
47         0 //设置串口中断优先级,设置为0意味着让系统从1-3级中自动选择一个
48     );
49     /*设置串口参数*/
50     uart_param_config(UART_NUM_1, &uart_config);
51     /*设置串口的TX,RX,RTS,DTR引脚*/ //不使用RTS,DTR
52     uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
53
54     /*申请一块内存,用于临时存储接收的数据*/
55     uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
56     uart_event_t event;
57     while (1) {
58         if(xQueueReceive(QueueHandle_t_uart1, (void *) &event, portMAX_DELAY))
59         {
60             switch(event.type) {
61                 case UART_DATA: //接收到数据
62                     //读取接收的数据
63                     uart_read_bytes(UART_NUM_1, data, event.size, portMAX_DELAY);
64                     //返回接收的数据
```

## 关于模式匹配和485方向控制,参考

[https://docs.espressif.com/projects/esp-idf/zh\\_CN/latest/esp32/api-reference/peripherals/uart.html?highlight=uart\\_pattern\\_det#](https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32/api-reference/peripherals/uart.html?highlight=uart_pattern_det#)

- nmea0183\_parser
- uart\_async\_rxtxtasks
- uart\_echo
- **uart\_echo\_rs485**
- **uart\_events**
- uart\_select

```
C uart_events_example_main.c 8 x
main > C uart_events_example_main.c > app_main(void)
133     };
134     //Install UART driver, and get the queue.
135     uart_driver_install(EX_UART_NUM, BUF_SIZE * 2, BUF_SIZE * 2, 20, &uart0_queue, 0);
136     uart_param_config(EX_UART_NUM, &uart_config);
137
138     //Set UART log level
139     esp_log_level_set(TAG, ESP_LOG_INFO);
140     //Set UART pins (using UART0 default pins ie no changes.)
141     uart_set_pin(EX_UART_NUM, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
142
143     //Set uart pattern detect function.
144     uart_enable_pattern_det_baud_intr(EX_UART_NUM, '+', PATTERN_CHR_NUM, 9, 0, 0);
145     //Reset the pattern queue length to record at most 20 pattern positions.
146     uart_pattern_queue_reset(EX_UART_NUM, 20); 接收数据匹配
147
148     //Create a task to handler UART event from ISR
149     xTaskCreate(uart_event_task, "uart_event_task", 2048, NULL, 12, NULL);
150
151
```

## 官方文档

SDMMC Host

SD SPI Host

SDIO Slave

Sigma-delta Modulation

SPI Master

SPI Slave

Secure Element

触摸传感器

TWAI

UART

Overview

Functional Overview

Overview of RS485 specific communication options

Application Examples

API Reference

协议

配网

存储

System

Configuration Options

Error Codes Reference

H/W 参考

API 指南

Libraries and Frameworks

» API 参考 » 外设 API » UART

Edit on GitHub

## UART

### Overview

A Universal Asynchronous Receiver/Transmitter (UART) is a hardware feature that handles communication (i.e., timing requirements and data framing) using widely-adopted asynchronous serial communication interfaces, such as RS232, RS422, RS485. A UART provides a widely adopted and cheap method to realize full-duplex or half-duplex data exchange among different devices.

The ESP32 chip has three UART controllers (UART0, UART1, and UART2), each featuring an identical set of registers to simplify programming and for more flexibility.

Each UART controller is independently configurable with parameters such as baud rate, data bit length, bit ordering, number of stop bits, parity bit etc. All the controllers are compatible with UART-enabled devices from various manufacturers and can also support Infrared Data Association protocols (IrDA).

### Functional Overview

The following overview describes how to establish communication between an ESP32 and other UART devices using the functions and data types of the UART driver. The overview reflects a typical programming workflow and is broken down into the sections provided below:

- Setting Communication Parameters - Setting baud rate, data bits, stop bits, etc.

分类: [ESP32学习开发](#)

好文要顶

关注我

收藏该文

杨奉武

关注 - 1

粉丝 - 633

0

0

« 上一篇: [104-ESP32\\_SDK开发-软件定时器esp\\_timer](#)

posted on 2021-08-07 15:45 杨奉武 阅读(0) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

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：

- [NET大牛之路 006] 了解 Roslyn 编译器
- 源码 | “@Value 注入失败”引发的一系列骚操作
- Redis挂了，流量把数据库也打挂了，怎么办？
- 五个 .NET 性能小贴士
- Web动画 | 科技感十足的暗黑字符雨动画



#### 最新新闻：

- 任正非：将军是打出来的，不是培养出来的，也不是分配出来的
  - 穷小子击败世袭财阀？这位韩国新首富有点东西
  - BAT “圈地战争”简史：巨头如何改变互联网？
  - “飞高了”矩阵不做第二个美团
  - 苹果要偷看你手机电脑上的照片了
- » 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。