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/

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 / 00:00

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： 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

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

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

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(ZLESP8266A物联网开发(1)
STM32+ESP8266+AIR202/302远程升级方案(16)
STM32+ESP8266+AIR202/302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

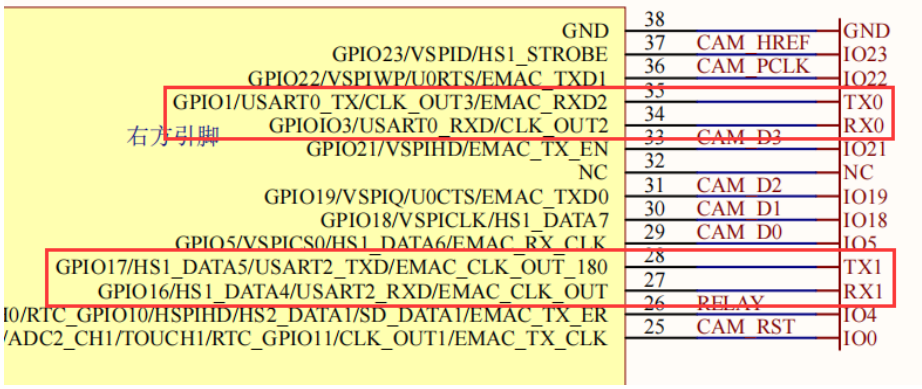
阅读排行榜

1. ESP8266使用详解(AT,LUA,SDK)(172765)
2. 1-安装MQTT服务器(Windows),并连接测试(98876)
3. ESP8266刷AT固件与nodemcu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(38514)

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)

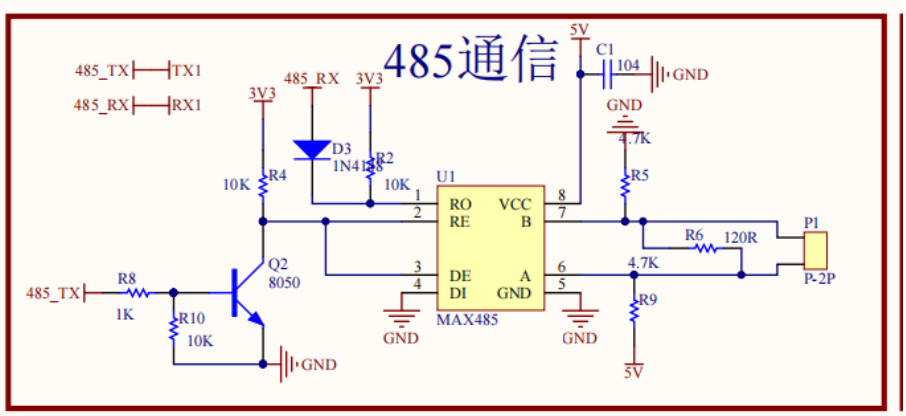


开发板上也把串口1连接了485上.

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客服端+eps8266+单片机+路由器之远程控制系统(31321)

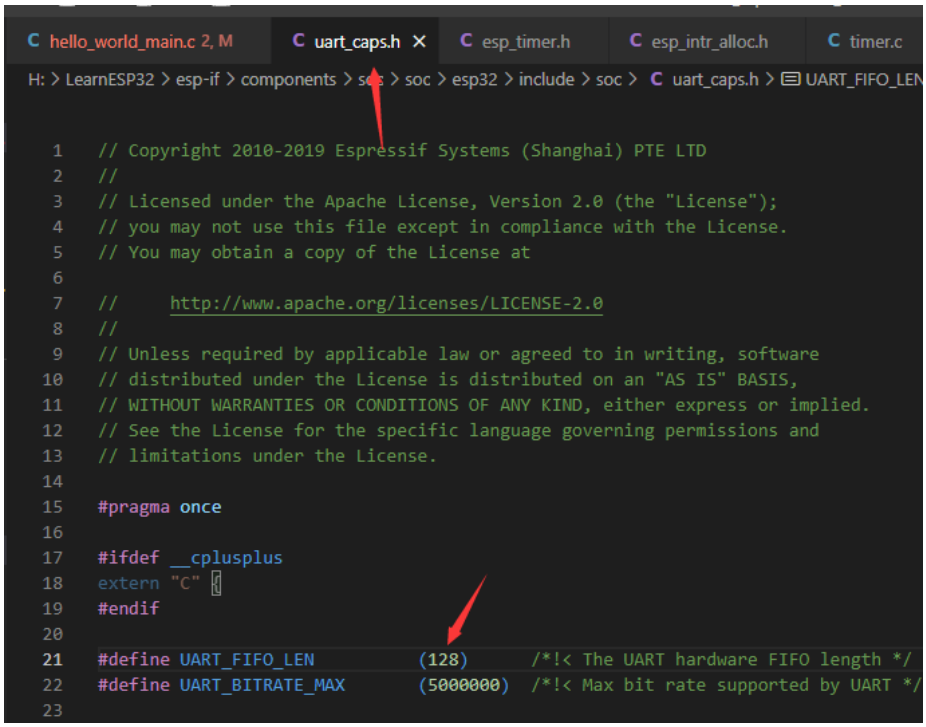
推荐排行榜

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.



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.

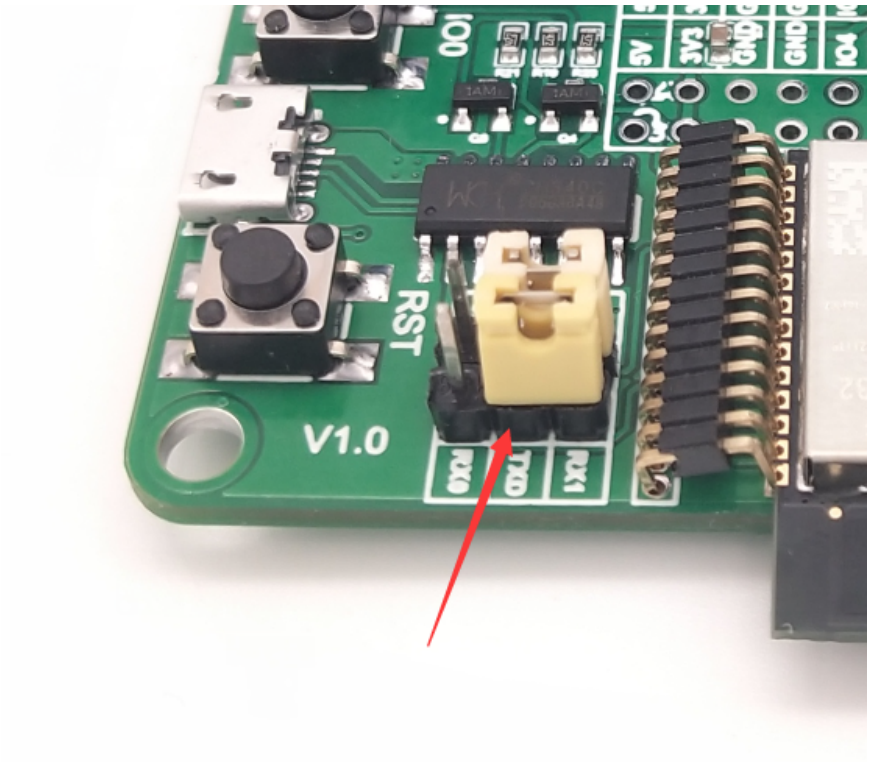
没有设置发送缓存,调用 uart_write_bytes 发送数据的时候是阻塞的.

```c
#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(128)
/*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_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);
    while (1) {
        //接收串口数据                                          //每隔10ms判断一次,可以写成portMAX_DELAY(一直判断)
        int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 10 / portTICK_RATE_MS);
        //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);
}
```

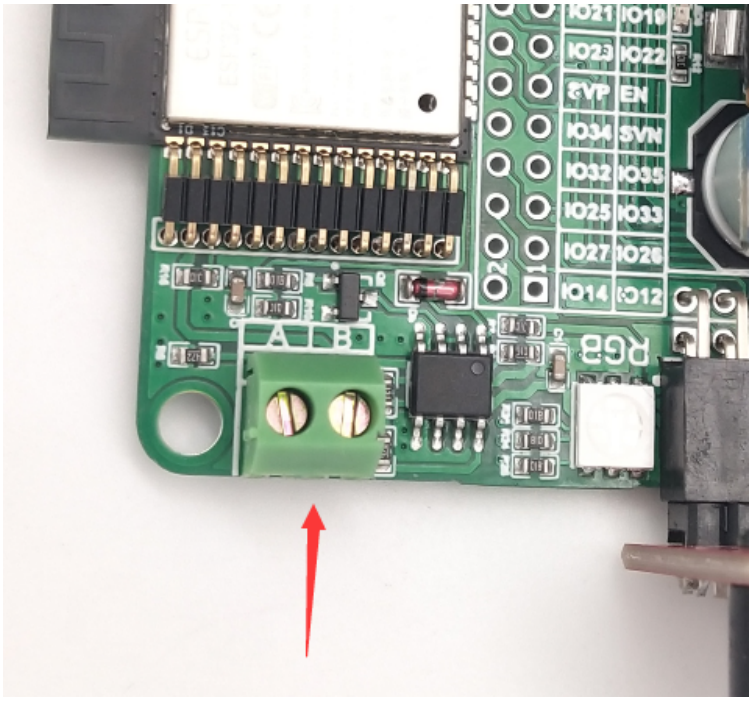
```c
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"
#include "driver/timer.h"
#include "esp_timer.h"
#include "driver/uart.h"

#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);
    }
```

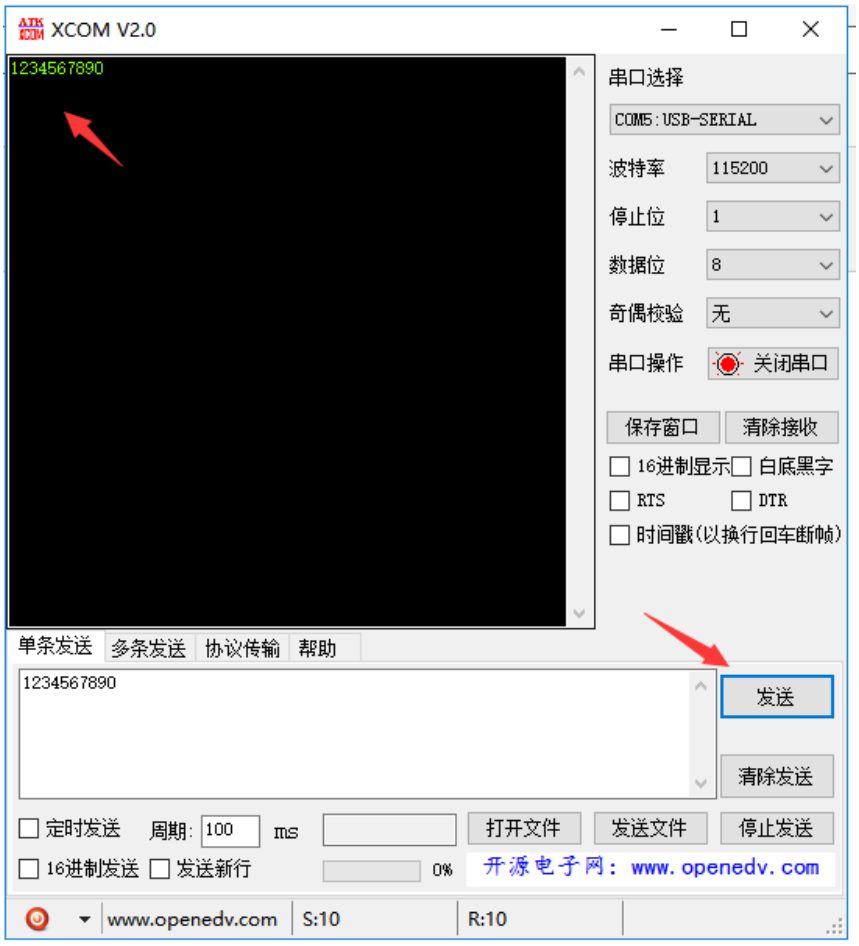
下载程序到开发板以后可以通过串口测试



485接口默认连接串口1,也可以使用485进行通讯

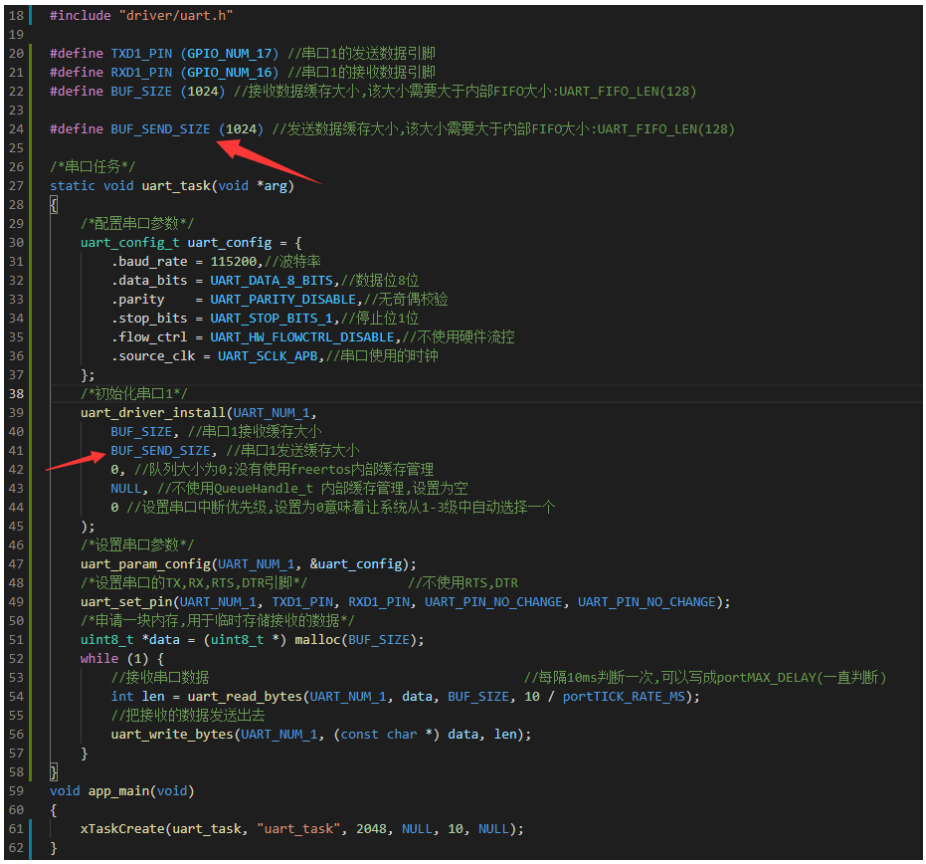


发送什么数据将会返回什么数据



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.



```c
#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(128)

#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)

/*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
        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_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);
    while (1) {
        //接收串口数据                                          //每隔10ms判断一次,可以写成portMAX_DELAY(一直判断)
        int len = uart_read_bytes(UART_NUM_1, data, BUF_SIZE, 10 / portTICK_RATE_MS);
        //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);
}
```

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

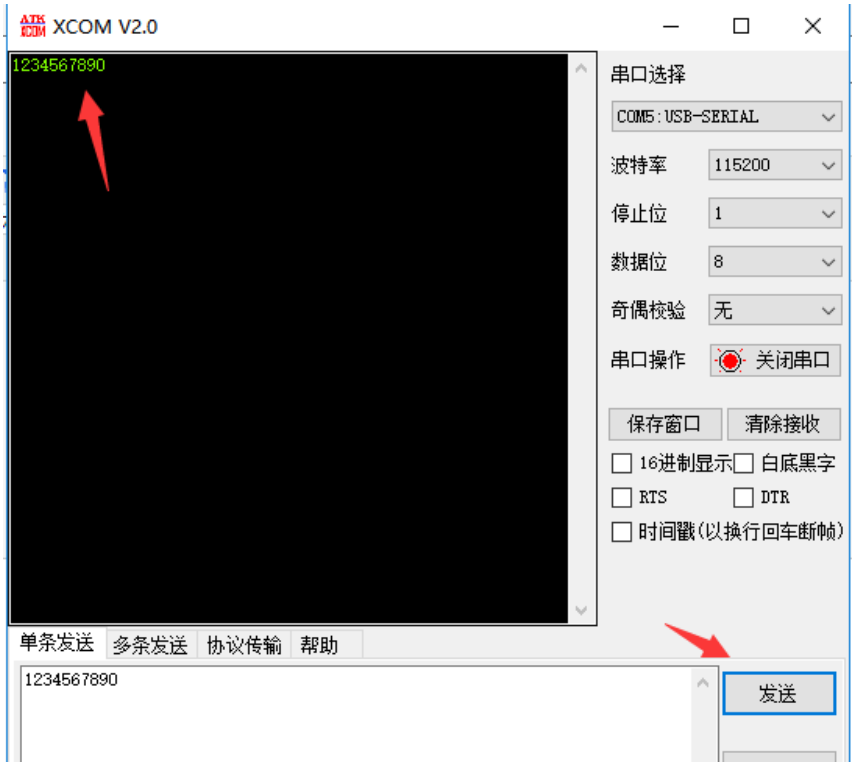
```c
#define BUF_SEND_SIZE (1024) //发送数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN(128)


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))
        {
```

从缓存管理中获取数据

```c
main > C hello_world_main.c > uart_task(void *)
            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))
            {
                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)
```

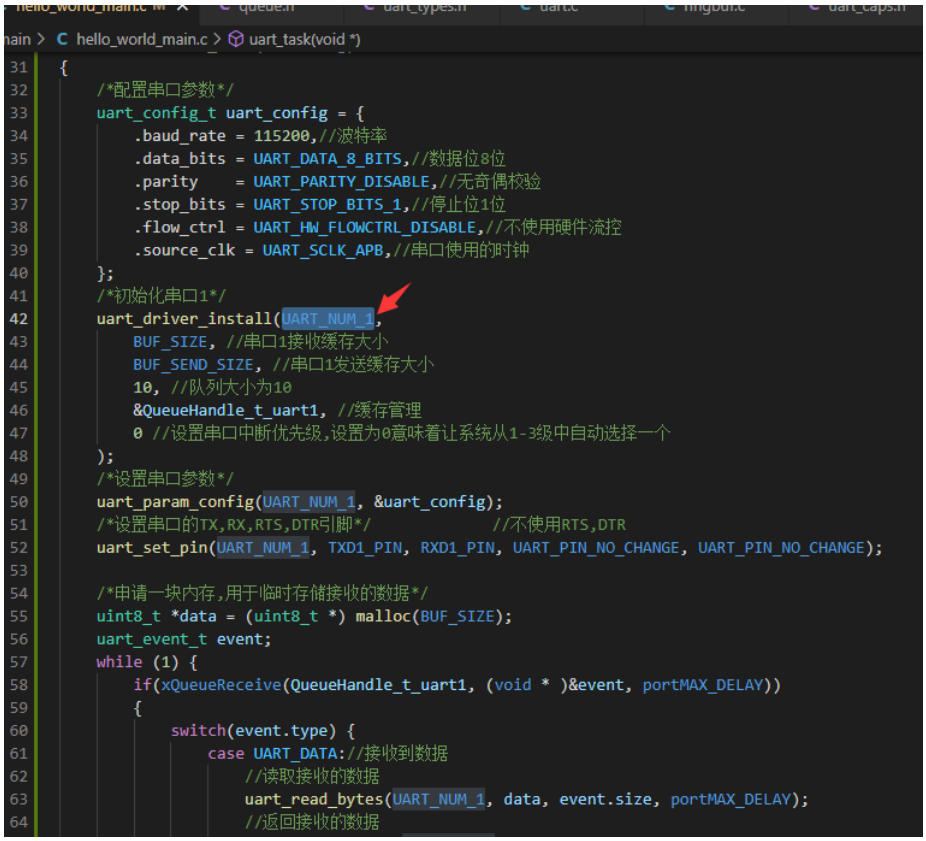
```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"
#include "driver/timer.h"
#include "esp_timer.h"
#include "driver/uart.h"

#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

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

    /*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)
        {
            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_DEL
                    //返回接收的数据
                    uart_write_bytes(UART_NUM_1, (const char*) data, event.si
                    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

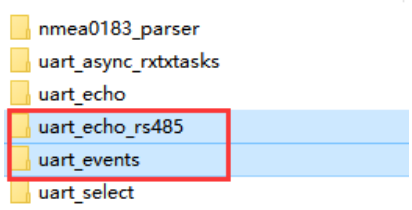
把以下变量的最后一个数字改为0或者2即可



```
main > C hello_world_main.c > uart_task(void *)
{
    /*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))
        {
            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
```

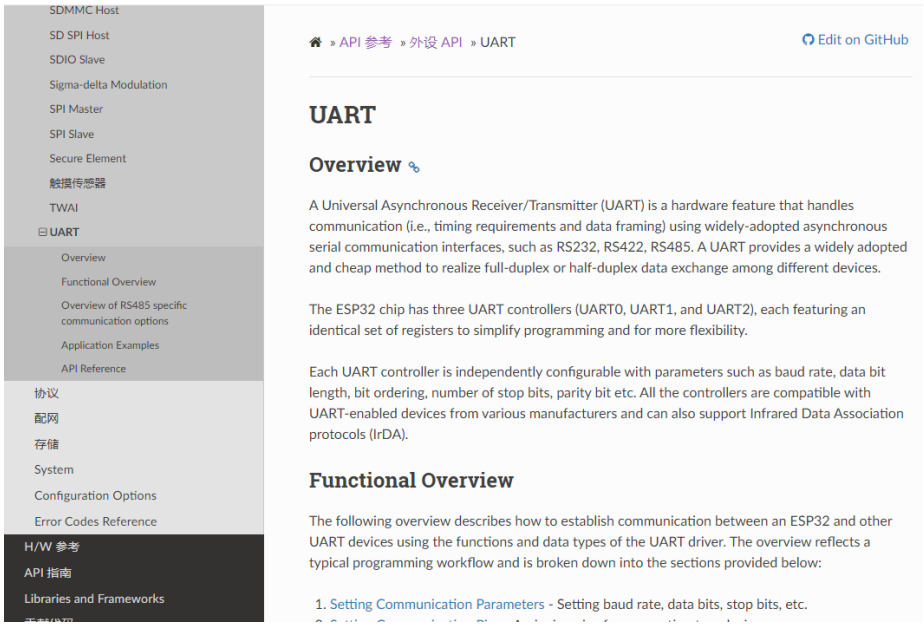
关于模式匹配和485方向控制,参考

https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32/api-reference/peripherals/uart.html?highlight=uart_pattern_det#



```c
uart_events_example_main.c 8 ×
main > C uart_events_example_main.c > app_main(void)
    };
    //Install UART driver, and get the queue.
    uart_driver_install(EX_UART_NUM, BUF_SIZE * 2, BUF_SIZE * 2, 20, &uart0_queue, 0);
    uart_param_config(EX_UART_NUM, &uart_config);

    //Set UART log level
    esp_log_level_set(TAG, ESP_LOG_INFO);
    //Set UART pins (using UART0 default pins ie no changes.)
    uart_set_pin(EX_UART_NUM, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, UART_PIN

    //Set uart pattern detect function.
    uart_enable_pattern_det_baud_intr(EX_UART_NUM, '+', PATTERN_CHR_NUM, 9, 0, 0);
    //Reset the pattern queue length to record at most 20 pattern positions.
    uart_pattern_queue_reset(EX_UART_NUM, 20);    接收数据匹配

    //Create a task to handler UART event from ISR
    xTaskCreate(uart_event_task, "uart_event_task", 2048, NULL, 12, NULL);
}
```

官方文档



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

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...

Copyright © 2021 杨奉武

单片机,物联网,上位机,…

扫一扫二维码，加入群聊。