

优秀不够，你是否无可替代

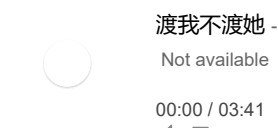
知识从未如此性感。 烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人
QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 770, 文章 - 0, 评论 - 321, 阅读 - 187万

导航

博客园
首页
新随笔
联系
订阅 
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



昵称： 杨奉武
园龄： 5年11个月
粉丝： 654
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(5)
Android(22)
Android 开发(8)
C# 开发(4)
CH395Q学习开发(17)
CH579M物联网开发(7)
CH579M学习开发(7)
ESP32学习开发(19)
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)

106-ESP32_SDK开发-SPI

<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Git下载链

接:<https://github.com/yangfengwu45/learn-esp32.git>

资料源码百度网

盘:<https://pan.baidu.com/s/10SBk0NsvLtJYHpDab9islg>
提取码：25oy

【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](#)

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(华大单片机)学习开
发(5)
NB-IOT Air302 AT指令和LUA
脚本语言开发(27)
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(24)
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)
更多

最新评论

1. Re:102-CH579M学习开
发-基本外设-串口
IN4148是不是接反了呀？
--freemote
2. Re:006-
STM32+ESP8266+AIR202/3
基本控制篇(阿里云物联网平台
在阿里云物联网平台上动态注
设备(基于STM32+ESP8266)
你好，请问下您的这个项目
使用的是阿里的LinkSdk吗
--码农29

阅读排行榜

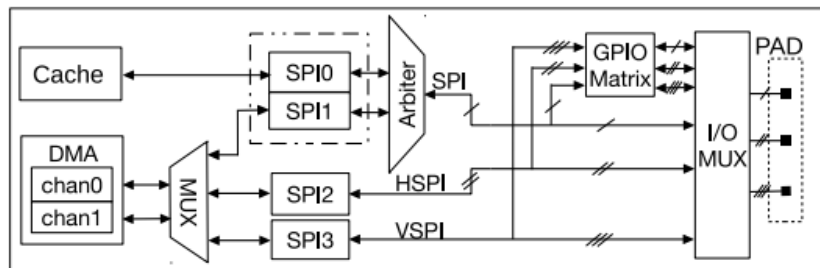
1. ESP8266使用详解(AT,LUA,
SDK)(173177)
2. 1-安装MQTT服务器(Windo
ws),并连接测试(100637)
3. ESP8266刷AT固件与node
mcu固件(65359)
4. 用ESP8266+android,制作
自己的WIFI小车(ESP8266篇)
(65155)
5. 有人WIFI模块使用详解(387
85)

- -----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](#)
- [106-SPI](#)
- [107-flash数据存储nvs](#)
- -----网络通信-----

■ [201-softAP模式配置模组发出的热点](#)
- [202-station模式配置模组连接路由器热
点](#)

说明



ESP32共有4路SPI, SPI0,SPI1,SPI2,SPI3

同时SPI2还取了个别名叫做 HSPI；同时SPI3还取了个别名叫做 VSPI

SPI0,SPI1是只访问缓存芯片使用,可以用来给模组用来扩容ram.

SPI0,SPI1的引脚分布在

GPIO6(SPI_CLK); GPIO7(SPI_Q); GPIO8(SPI_D);

GPIO9(SPI_HD); GPIO10(SPI_WP); GPIO11(SPI_CS0);

用户不要使用这些引脚作为普通IO使用.关于SPI0和SPI1的使用会在以后的教程里面.

这节教程是学习SPI2(HSPI)和SPI3(VSPI); HSPI和VSPI使用是一样的哈.

如果按照下图分配引脚(不包含CS引脚), SPI的传输速率可以达到 80M

- 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)(36209)
- 7. 关于TCP和MQTT之间的转
换(33929)
- 8. C#中public与private与stat
ic(33417)
- 9. android 之TCP客户端编程
(32221)
- 10. android服务端+eps8266
+单片机+路由器之远程控制系
统(31423)

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)

Pin Name	HSPI	VSPI
	GPIO Number	
CS0*	15	5
SCLK	14	18
MISO	12	19
MOSI	13	23
QUADWP	2	22
QUADHD	4	21

咱们就以HSPI为例(基础使用)

1.配置HSPI信号线的参数

```
C hello_world_main.c M X C spi_caps.h
main > C hello_world_main.c > app_main(void)
17 #include "esp_timer.h"
18 #include "driver/uart.h"
19 #include "esp_log.h"
20 #include "driver/spi_master.h"
21
22
23 #define PIN_NUM_MISO 12
24 #define PIN_NUM_MOSI 13
25 #define PIN_NUM_CLK 14
26 #define PIN_NUM_CS 15
27
28 spi_device_handle_t spi_device;
29
30 void app_main(void)
31 {
32     esp_err_t ret;
33
34     //配置SPI总线参数
35     spi_bus_config_t buscfg={
36         .miso_io_num = PIN_NUM_MISO, //设置主机输入从机输出引脚(接收数据引脚)
37         .mosi_io_num = PIN_NUM_MOSI, //设置主机输出从机输入引脚(发送数据引脚)
38         .sclk_io_num = PIN_NUM_CLK, //设置时钟引脚
39         .quadwp_io_num = -1, //不使用wp信号线
40         .quadhd_io_num = -1, //不使用hd信号线
41         .max_transfer_sz = SOC_SPI_MAXIMUM_BUFFER_SIZE, //一次性最大传输字节个数;默认为SOC_SPI_MAXIMUM_BUFFER_SIZE(启用DMA时需要设置为0)
42     };
43     //初始化配置SPI总线(配置HSPI);最后设置为0为不启用DMA
44     ret = spi_bus_initialize(HSPI_HOST, &buscfg, 0);
45     ESP_ERROR_CHECK(ret);
46
47     //配置SPI数据传输参数(可以新建多个这种结构体变量,相当于挂载SPI设备,一个SPI最多挂载3个设备,需要设置不同的片选引脚)
48     spi_device_interface_config_t devcfg={
49
```

2.配置数据传输参数

```
C hello_world_main.c X C spi_master.h
main > C hello_world_main.c > app_main(void)
20 #include "driver/spi_master.h"
21
22
23 #define PIN_NUM_MISO 12
24 #define PIN_NUM_MOSI 13
25 #define PIN_NUM_CLK 14
26 #define PIN_NUM_CS 15
27
28 spi_device_handle_t spi_device; //存储所配置的SPI句柄
29
30 void app_main(void)
31 {
32     esp_err_t ret;
33
34     //配置SPI总线参数
35     spi_bus_config_t buscfg={
36         .miso_io_num = PIN_NUM_MISO, //设置主机输入从机输出引脚(接收数据引脚)
37         .mosi_io_num = PIN_NUM_MOSI, //设置主机输出从机输入引脚(发送数据引脚)
38         .sclk_io_num = PIN_NUM_CLK, //设置时钟引脚
39         .quadwp_io_num = -1, //不使用wp信号线
40         .quadhd_io_num = -1, //不使用hd信号线
41         .max_transfer_sz = SOC_SPI_MAXIMUM_BUFFER_SIZE, //一次性最大传输字节个数;默认为SOC_SPI_MAXIMUM_BUFFER_SIZE(启用DMA时需要设置为0)
42     };
43     //初始化配置SPI总线(配置HSPI);最后设置为0为不启用DMA
44     ret = spi_bus_initialize(HSPI_HOST, &buscfg, 0);
45     ESP_ERROR_CHECK(ret);
46
47     //配置SPI数据传输参数(可以新建多个这种结构体变量,相当于挂载SPI设备,一个SPI最多挂载3个设备,需要设置不同的片选引脚)
48     spi_device_interface_config_t devcfg={
49         .address_bits=0, //不需要地址数据
50         .clock_speed_hz=40*1000*1000, //频率40M
51         .command_bits=0, //不需要命令数据
52         .mode=0, //SPI模式(0,1,2,3)
53         .spics_io_num=PIN_NUM_CS, //设置CS引脚
54         .queue_size=1, //传输(事务)队列大小(必须设置,先设置1,后面会说明这个有什么用)
55     };
56
57     //把传输参数配置进SPI总线
58     ret = spi_bus_add_device(HSPI_HOST, &devcfg, &spi_device);
59     ESP_ERROR_CHECK(ret);
60 }
```

3.设置传输的数据,和发送数据

```
60
61
62 /*设置传输的数据*/
63 const char spi_data[2]={0xaa,0x55}; //2字节数据
64 /*传输事务*/
65 spi_transaction_t spi_transaction;
66 memset(&spi_transaction, 0, sizeof(spi_transaction));
67
68 spi_transaction.tx_buffer=spi_data; //发送的数据地址
69 spi_transaction.length=2*8; //传输数据大小(以数据位为单位);传两字节数据所以是2*8
70
71 while(1){
72     spi_device_transmit(spi_device, &spi_transaction); //使用SPI发送数据
73
74     vTaskDelay(1000 / portTICK_PERIOD_MS);
75 }
76
77
```

```
#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"
#include "esp_log.h"
#include "driver/spi_master.h"

#define PIN_NUM_MISO 12
#define PIN_NUM_MOSI 13
#define PIN_NUM_CLK 14
#define PIN_NUM_CS 15
```

```

spi_device_handle_t spi_device; //存储所配置的SPI句柄

void app_main(void)
{
    esp_err_t ret;

    //配置SPI总线参数
    spi_bus_config_t buscfg={
        .miso_io_num = PIN_NUM_MISO, //设置主机输入从机输出引脚(接收数据引脚)
        .mosi_io_num = PIN_NUM_MOSI, //设置主机输出从机输入引脚(发送数据引脚)
        .sclk_io_num = PIN_NUM_CLK, //设置时钟引脚
        .quadwp_io_num = -1, //不使用wp信号线
        .quadhd_io_num = -1, //不使用hd信号线
        .max_transfer_sz = SOC_SPI_MAXIMUM_BUFFER_SIZE, //一次性最大传输字节个数;最大
    };
    //初始化配置SPI总线(配置HSPI);最后设置为0为不启用DMA
    ret = spi_bus_initialize(HSPI_HOST, &buscfg, 0);
    ESP_ERROR_CHECK(ret);

    //配置SPI数据传输参数(可以新建多个这种结构体变量,相当于挂载SPI设备,一个SPI最多挂载3个)
    spi_device_interface_config_t devcfg={
        .address_bits=0, //不需要地址数据
        .clock_speed_hz=40*1000*1000, //频率40M
        .command_bits=0, //不需要命令数据
        .mode=0, //SPI模式(0,1,2,3)
        .spics_io_num=PIN_NUM_CS, //设置CS引脚
        .queue_size=1, //传输(事务)队列大小(必须设置,先设
    };
    //把传输参数配置进SPI总线
    ret = spi_bus_add_device(HSPI_HOST, &devcfg, &spi_device);
    ESP_ERROR_CHECK(ret);

    /*设置传输的数据*/
    const char spi_data[2]={0xaa,0x55}; //2字节数据
    /*传输事务*/
    spi_transaction_t spi_transaction;
    memset(&spi_transaction, 0, sizeof(spi_transaction));

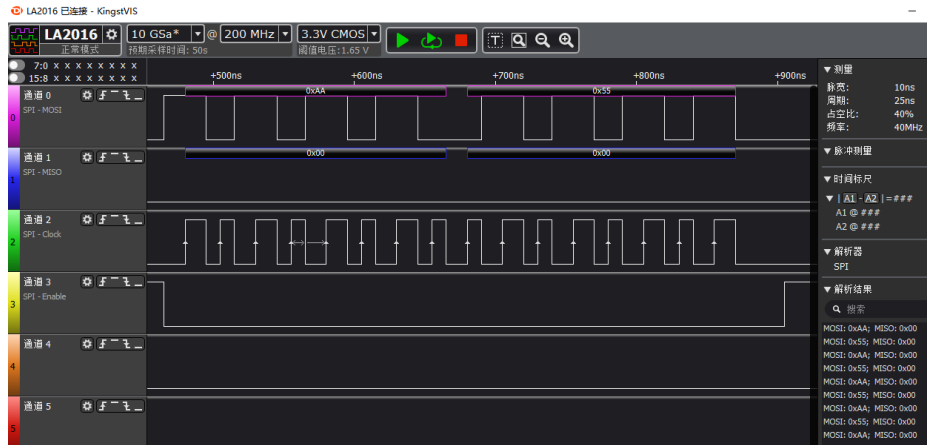
    spi_transaction.tx_buffer=spi_data; //发送的数据地址
    spi_transaction.length=2*8; //传输数据大小(以数据位为单位);传两字节数据

    while(1){
        spi_device_transmit(spi_device, &spi_transaction); //使用SPI发送数据

        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

```





3.关于上面的 配置SPI数据传输参数 里面的 命令和地址数据个数

在SPI传输的时候就是使用SPI传输的数据, 有时候咱想访问一个芯片, 假设是读取芯片上的某个地址上的数据;

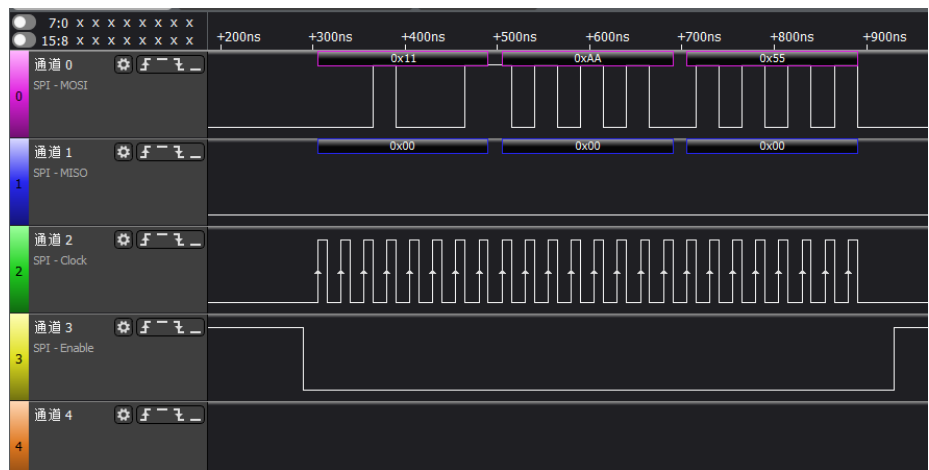
那个芯片的手册上一般会说要先使用SPI发送个什么命令,然后再发送个什么地址,然后就可以读取到数据了.

实际上就是使用SPI发送个数据然后再使用SPI发送个数据嘛.只不过发送的第一个数据叫做xxxx命令

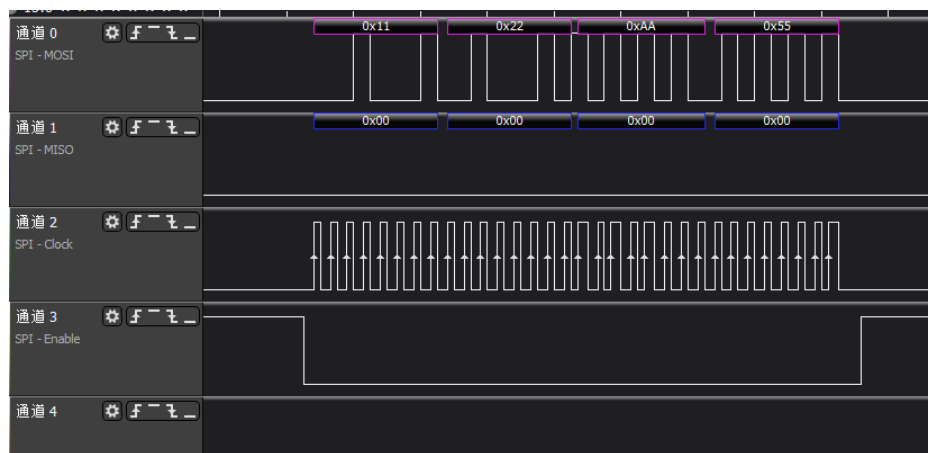
第二个数据就做xxx地址.....

假设在传输上面的0xAA和0x55之前,想先发送个0x11命令数据

```
C hello_world_main.c M x C spi_master.h
main > C hello_world_main.c > app_main(void)
39 .quadwp_io_num = -1, //不使用WP信号线
40 .quadhd_io_num = -1, //不使用HD信号线
41 .max_transfer_sz = SOC_SPI_MAXIMUM_BUFFER_SIZE, //一次性最大传输字节个数,默认为SOC_SPI_MAXIMUM_BUFFER_SIZE(扇
42 );
43 //初始化配置SPI总线(配置HSPI);最后设置为0为不启用DMA
44 ret = spi_bus_initialize(HSPI_HOST, &buscfg, 0);
45 ESP_ERROR_CHECK(ret);
46
47
48 //配置SPI数据传输参数(可以新建多个这种结构体变量,相当于挂载SPI设备,一个SPI最多挂载3个设备,需要设置不同的片选引脚)
49 spi_device_interface_config_t devcfg={
50     .address_bits=0, //不需要地址数据
51     .clock_speed_hz=40*1000*1000, //频率40M
52     .command_bits=8, //位为单位,最长16位,即两字节
53     .mode=0, //SPI模式(0,1,2,3)
54     .spics_io_num=PIN_NUM_CS, //设置CS引脚
55     .queue_size=1, //传输(事务)队列大小(必须设置,先设置1,后面会说明这个有什么用)
56 };
57 //把传输参数配置进SPI总线
58 ret = spi_bus_add_device(HSPI_HOST, &devcfg, &spi_device);
59 ESP_ERROR_CHECK(ret);
60
61
62 /*设置传输的数据*/
63 const char spi_data[2]={0xaa,0x55}; //2字节数据
64 /*传输事务*/
65 spi_transaction_t spi_transaction;
66 memset(&spi_transaction, 0, sizeof(spi_transaction));
67 spi_transaction.cmd=0x11; //发送的数据地址
68 spi_transaction.tx_buffer=spi_data; //发送的数据地址
69 spi_transaction.length=2*8; //传输数据大小(以数据位为单位);传两字节数据所以是2*8
70
71 while(1){
72     spi_device_transmit(spi_device, &spi_transaction); //使用SPI发送数据
73
74     vTaskDelay(1000 / portTICK_PERIOD_MS);
75 }
76
77
78 /**
```



假设在传输上面的0xAA和0x55之前,想先发送个0x11命令数据,然后再发送个地址数据0x22



4.关于传输事务大小

```
40
47
48 //配置SPI数据传输参数(可以新建多个这种结构体变量,相当于挂载SPI设备,一个SPI最多挂载3个设备,需要设置不同的片选引脚)
49 spi_device_interface_config_t devcfg={
50     .address_bits=8,                //位为单位,最长64位,即8字节
51     .clock_speed_hz=40*1000*1000,  //频率40M
52     .command_bits=8,               //位为单位,最长16位,即两字节
53     .mode=0,                       //SPI模式(0,1,2,3)
54     .spics_io_num=PIN_NUM_CS,      //设置CS引脚
55     .queue_size=1,                 //传输(事务)队列大小(必须设置,先设置1,后面会说明这个有什么用)
56 };
57 //把传输参数配置进SPI总线
58 ret = spi_bus_add_device(HSPI_HOST, &devcfg, &spi_device);
59 ESP_ERROR_CHECK(ret);
60
61
62 /*设置传输的数据*/
63 const char spi_data[2]={0xaa,0x55}; //2字节数据
64 /*传输事务*/
65 spi_transaction_t spi_transaction;
66 memset(&spi_transaction, 0, sizeof(spi_transaction));
67 spi_transaction.cmd=0x11;
68 spi_transaction.addr=0x22;
69 spi_transaction.tx_buffer=spi_data; //发送的数据地址
70 spi_transaction.length=2*8;         //传输数据大小(以数据位为单位);传两字节数据所以是2*8
71
72 while(1){
73     spi_device_transmit(spi_device, &spi_transaction); //使用SPI发送数据
74     vTaskDelay(1000 / portTICK_PERIOD_MS);
75 }
76
77
78
```

假设我们需要传输多条数据,那个传输事务大小就起作用了

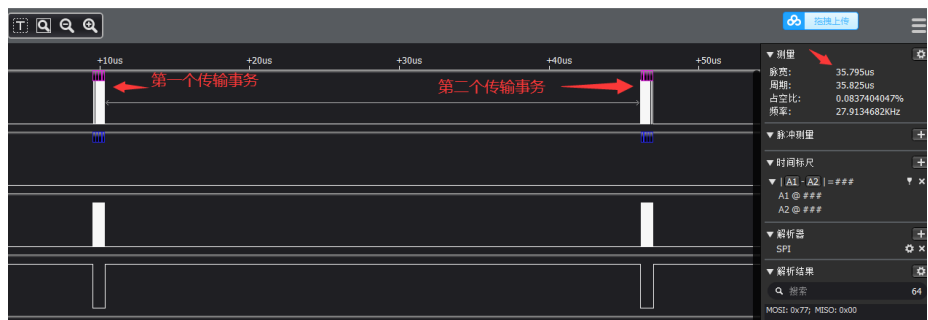
我设置的为8,就是队列最大可以保存8条要传输的事务.

下面增加了一个传输事务.

传输的时候把要传输的事务放到 `spi_device_polling_transmit` 函数就可以了

```
C hello_world_main.c M X C spi_master.c
main > C hello_world_main.c > ...
51     .clock_speed_hz=40*1000*1000,           //频率40M
52     .command_bits=8,                         //位为单位,最长16位,即两字节
53     .mode=0,                                 //SPI模式(0,1,2,3)
54     .spics_io_num=PIN_NUM_CS,                //设置CS引脚
55     .queue_size=8,                           //传输(事务)队列大小
56 };
57 //把传输参数配置进SPI总线
58 ret = spi_bus_add_device(HSPI_HOST, &devcfg, &spi_device);
59 ESP_ERROR_CHECK(ret);
60
61
62 /*设置传输的数据*/
63 const char spi_data[2]={0xaa,0x55}; //2字节数据
64 /*传输事务*/
65 spi_transaction_t spi_transaction;
66 memset(&spi_transaction, 0, sizeof(spi_transaction));
67 spi_transaction.cmd=0x11;
68 spi_transaction.addr=0x22;
69 spi_transaction.tx_buffer=spi_data; //发送的数据地址
70 spi_transaction.length=2*8;         //传输数据大小(以数据位为单位);传两字节数据所以是2*8
71
72
73 /*设置传输的数据*/
74 const char spi_data1[2]={0x66,0x77}; //2字节数据
75 /*传输事务*/
76 spi_transaction_t spi_transaction1;
77 memset(&spi_transaction1, 0, sizeof(spi_transaction1));
78 spi_transaction1.cmd=0x88;
79 spi_transaction1.addr=0x99;
80 spi_transaction1.tx_buffer=spi_data1; //发送的数据地址
81 spi_transaction1.length=2*8;         //传输数据大小(以数据位为单位);传两字节数据所以是2*8
82
83 while(1){
84     //spi_device_transmit(spi_device, &spi_transaction); //使用SPI发送数据
85
86     spi_device_polling_transmit(spi_device, &spi_transaction); //使用SPI发送数据
87     spi_device_polling_transmit(spi_device, &spi_transaction1); //使用SPI发送数据
88
89     vTaskDelay(1000 / portTICK_PERIOD_MS);
90 }
91
92
```

底层默认传输事务之间的时间间隔为35us左右



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 - 654

0

0

« 上一篇: [ESP8266 SDK开发: 外设篇-系统任务\(消息队列,通知\)](#)

» 下一篇: [200-Air724UG\(4G全网通GPRS\)开发-下载和运行第一个lua程序](#)

posted on 2021-08-16 00:59 杨奉武 阅读(79) 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】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】跨平台组态\工控\仿真\CAD 50万行C++源码全开放免费下载！

【推荐】百度智能云超值优惠：新用户首购云服务器1核1G低至69元/年

【推荐】和开发者在一起：华为开发者社区，入驻博客园科技品牌专区

【推广】园子与爱卡汽车爱宝险合作，随手就可以买一份的百万医疗保险



编辑推荐：

- 微前端框架single-spa初探
- 并发编程之：深入解析线程池
- 源码解析 .Net 中 Host 主机的构建过程
- 理解ASP.NET Core - Dependency Injection
- 记一次 .NET 某机械臂智能机器人控制系统 MRS CPU 爆高分析

最新新闻：

- SpaceX飞船和火箭推上发射台 将送全平民机组入轨 (2021-09-13 08:33)
 - 分析：Epic Games诉苹果案结果让开发商更难告赢谷歌 (2021-09-13 08:26)
 - 苹果秋季发布会前瞻：iPhone 13外观没太大变化 (2021-09-13 08:18)
 - 突破性的技术产生了关于硅、亚原子粒子和可能的"第五自然力"的新细节 (2021-09-13 08:10)
 - 新加坡的「自动驾驶」新名片：成于国小，忧也国小 (2021-09-13 08:00)
- » 更多新闻...

历史上的今天：

- 2019-08-16 2-移远GSM/GPRS M26 模块 Mini板 开发板(M26入门)
- 2019-08-16 嘉立创制作PCB流程

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。