

淘宝店铺

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 - 692, 文章 - 0, 评论 - 311, 阅读 - 173万

导航

博客园

首页

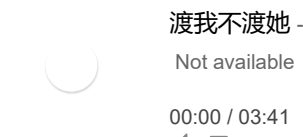
新随笔

联系

订阅 

管理


公告



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8个月

粉丝：606

关注：1

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(4)
ESP32学习开发(8)
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)

2-网络芯片CH395Q学习开发-学习资料说明,测试通信,获取硬件版本,代码移植说明

<p><iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnCH395Q" frameborder="0" scrolling="auto" width="100%" height="1500"></iframe></p>

网络芯片CH395Q学习开发

开发板链接:[开发板链接](#)

模组原理图:[模组原理图](#)

资料源码下载链

接:<https://github.com/yangfengwu45/CH395Q.c>

- [学习Android](#)
教程中搭配的Android，C#等教程如上，各个教程正在整理。
- [1-网络芯片CH395Q学习开发-硬件测试使用说明](#)
- [2-网络芯片CH395Q学习开发-学习资料说明,测试通信,获取硬件版本](#)
- [3-网络芯片CH395Q学习开发-芯片初始化,网线连接检测实验](#)
- [4-网络芯片CH395Q学习开发-关于中断检测和DHCP实验](#)

ESP8266 LUA开发基础入门篇
备份(22)
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+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(58)
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(10)
感想(6)
软件安装使用: MQTT(8)
软件安装使用: OpenResty(6)
数据处理思想和程序架构(24)
数据库学习开发(12)
更多

最新评论

1. Re:C#委托+回调详解
好文，撒也不说了，直接收
藏！
--杨咩咩plus
2. Re:2-STM32 替换说明-
CKS32, HK32, MM32,
APM32, CH32, GD32,
BLM32, AT32(推荐), N32,
HC华大系列
有用，谢谢！
--你跟游戏过吧

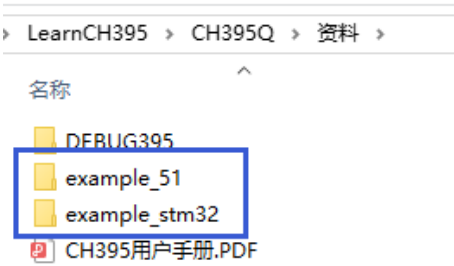
阅读排行榜

1. ESP8266使用详解(AT,LUA,
SDK)(172063)
2. 1-安装MQTT服务器(Windo
ws),并连接测试(96467)
3. ESP8266刷AT固件与node
mcu固件(63734)

资料说明

首先说明一下学习资料源码,在资料中有51单片
机,stm32相关的例程.

用户可根据自己的情况选择使用.



说明

我提供的例程是作为具体的讲解使用.默认以SPI通信为
主.

4. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(62506)
5. 有人WIFI模块使用详解(38092)
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)(35367)
7. 关于TCP和MQTT之间的转换(32220)
8. android 之TCP客户端编程(31276)
9. android服务端+eps8266+单片机+路由器之远程控制系统(31125)
10. C#中public与private与static(30918)

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)

提供的代码是使用STM32F103单片机编写的.

测试下这节的程序

1.用户可以使用杜邦线根据自己的情况设置和连接引脚

```
ch395cmd.h  CH395INC.H  CH395SPI.C  usart.c  delay.c  timer.c  main.c  delay.h  CH395SPLH
2  #ifndef _CH395SPI_H_
3  #define _CH395SPI_H_
4
5  #include "CH395INC.H"
6
7  /*****配置GPIO (根据自己的修改)*****/
8  //时钟
9  #define CH395_CONFIG_SPI_CLK() ( RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2,ENABLE) )
10 #define CH395_CONFIG_GPIO_CLK() ( RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Peri
11 //设置使用的SPI
12 #define USE_SPI SPI2
13 //SPI_CS -- 连接模块SCS引脚
14 #define CH395_CS_PORT GPIOB
15 #define CH395_CS_PIN GPIO_Pin_12
16 //SPI_CLK -- 连接模块SCK引脚
17 #define CH395_CLK_PORT GPIOB
18 #define CH395_CLK_PIN GPIO_Pin_13
19 //SPI_MISO -- 连接模块SDO引脚
20 #define CH395_MISO_PORT GPIOB
21 #define CH395_MISO_PIN GPIO_Pin_14
22 //SPI_MOSI -- 连接模块SDI引脚
23 #define CH395_MOSI_PORT GPIOB
24 #define CH395_MOSI_PIN GPIO_Pin_15
25 //RST -- 连接模块RST引脚
26 #define CH395_RST_PORT GPIOA
27 #define CH395_RST_PIN GPIO_Pin_8
28 //TX -- 连接模块Tx引脚
29 #define CH395_TX_PORT GPIOA
30 #define CH395_TX_PIN GPIO_Pin_3
31 //INT -- 连接模块INT引脚 (检测到该引脚低电平信号之后再获取数据)
32 #define CH395_INT_PORT GPIOA
33 #define CH395_INT_PIN GPIO_Pin_0
34 /*****/
```

2,注意!

要想模块使用SPI通信,模块的TX引脚需要在模块重启之前设置为低电平.

上面的引脚分配把模块的TX引脚接到了单片机的PA3上,也就是串口2的RX上,如果用户使用了串口2,请注意!

CH395 与单片机之间支持三种通讯接口：8 位并行接口、SPI 同步串行接口、异步串口。在芯片上电复位时，CH395 将采样 SEL 和 TXD 引脚的状态，根据这 2 个引脚状态的组合选择通讯接口，参考下表（表中 X 代表不关心此位，0 代表低电平，1 代表高电平或者悬空）。

SEL 引脚	TXD 引脚	选择通讯接口
1	1	异步串口
1	0	SPI 接口
0	1	8 位并口
0	0	错误接口

3.打开这节的程序

这一节测试使用SPI方式和模块进行通信,然后获取模块的芯片版本

- 1-硬件测试和使用说明
- 2-测试通信-获取芯片版本
- 调试助手
- 资料

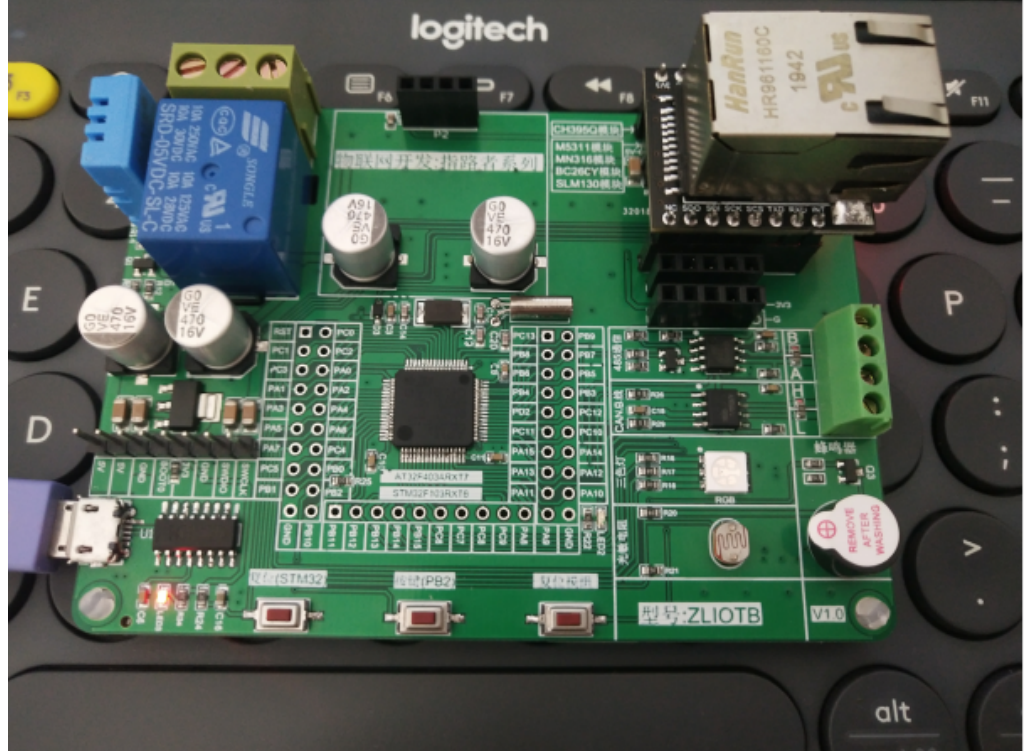
STM32F10xSPI

STM32F10xSPI

- CH395
- CMSIS
- Libraries
- mem
- Project
- Startup
- User

- Listing 20
- output 20
- Project 20
- JLinkLog.txt 20
- JLinkSettings.ini 20
- Project.uvgui.Administrator 20
- Project.uvgui.yang 20
- Project.uvgui_yang.bak 20
- Project.uvguix.yang 20
- Project.uvopt 20
- Project.uvoptx 20
- Project.uvprojx 20
- Project_Target 1.dep 20
- Project_uvopt.bak 20
- Project_uvproj.bak 20

4,我使用测试板进行测试



5,把程序下载到开发板,监控下单片机串口1打印的信息

注:也可能打印46..... 版本不一样

ATX
COM XCOM V2.0

```
CH395VER : 42
CH395VER : 42
CH395VER : 42
```

关于版本号

不同的版本号功能上有差异,版本号越高越好



50910E227 代表 版本2

50910E228 代表 版本4

推荐购买228或大于此数字的芯片.也就是4版本及其以上

版本功能主要差异:

假设使用芯片作为TCP服务器.

4版本及其以上的芯片支持多路TCP客户端连接其TCP服务器.

4版本以下的芯片只支持单路TCP客户端连接其TCP服务器

CH395Q获取可以通

过 `CMD_GET_GLOB_INT_STATUS` 和 `CMD_GET_GLOB_INT_STATUS_ALL` 两个命令来获取中断状态，

前者只能获取到低 8 位的中断状态，后者可以获取全部的中断状态，使用时需要注意，任何版本的芯片都支持

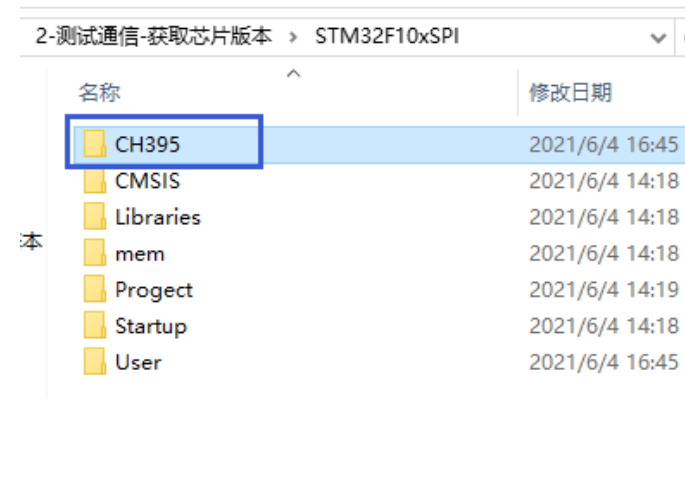
`CMD_GET_GLOB_INT_STATUS` 命令，

如果芯片版本号大于等于 0X44 且使用了Socket4- Socket7 则只能用 `CMD_GET_GLOB_INT_STATUS_ALL`。

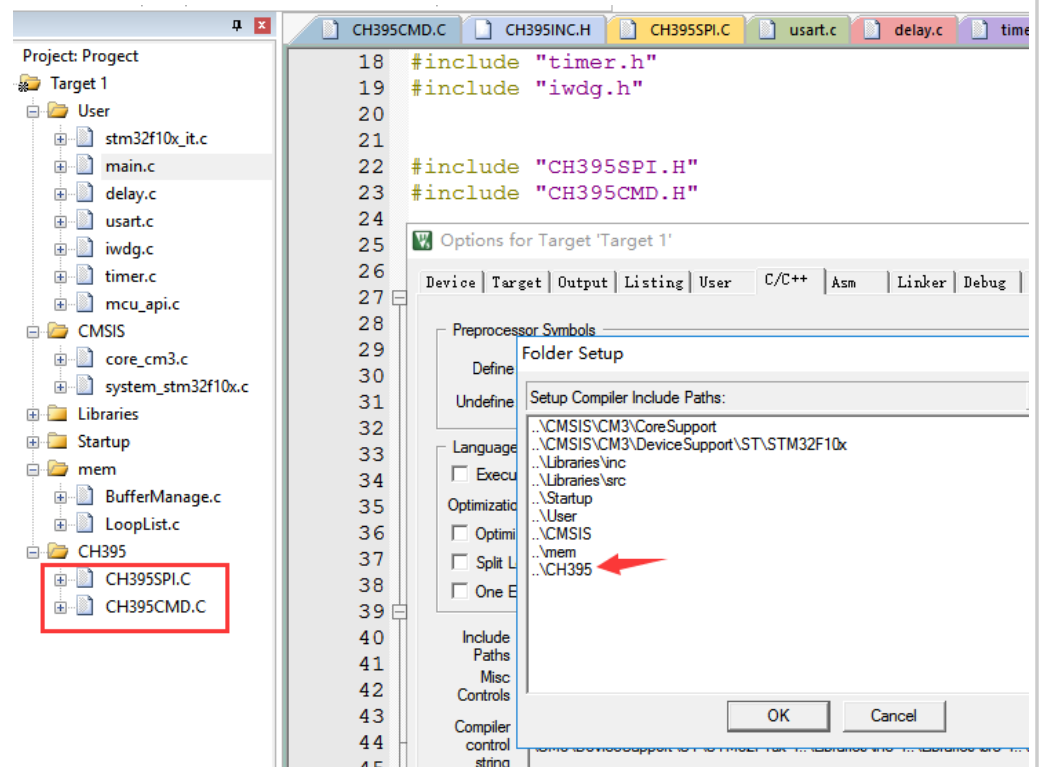
芯片版本号小于 0X44 不支持CMD_GET_GLOB_INT_STATUS_ALL 命令。

程序移植使用说明

1.把以下文件夹放到自己的工程



2.添加个分组,分组里面添加CH395SPI.C和CH395CMD.C文件; 包含头文件路径



3.根据自己的情况替换延时函数



```
1  /***** (C) COPYRIGHT *****/
2  * File Name      : SPI_HW.C
3  * Author         : WXF
4  * Version        : V1.0
5  * Date           : 2013/12/19
6  * Description    : CH395芯片 CH395芯片 硬件标准SPI串行
7  *                : 提供I/O接口子程序
8  *****/
9  #include "CH395SPI.H"
10 #include "delay.h"
11 #include "CH395INC.H"
12
13
14 /*****
15 * Function Name   : Delay_us
16 * Description     : 微秒级延时函数(基本准确)
17 * Input          : delay---延时值
18 * Output         : None
19 * Return         : None
20 *****/
21 void mDelayus( UINT8 delay )
22 {
23     //替换自己的延时us函数
24     delay_us(delay);
25 }
26
27 /*****
28 * Function Name   : Delay_ms
29 * Description     : 毫秒级延时函数(基本准确)
30 * Input          : delay---延时值
31 * Output         : None
32 * Return         : None
33 *****/
34 void mDelayms( UINT8 delay )
35 {
36     //替换自己的延时ms函数
37     delay_ms(delay);
38 }
39
```

4.如果用户使用的STM32F103系列的单片机的是硬件SPI,只需要修改这个地方即可


```
CH395CMD.C CH395INC.H CH395SPI.C usart.c delay.c timer.c main.c delay.h CH395SPLH
1
2 #ifndef CH395SPI_H
3 #define CH395SPI_H
4
5 #include "CH395INC.H"
6
7 /*****配置GPIO (根据自己的修改) *****/
8 //时钟
9 #define CH395_CONFIG_SPI_CLK() ( RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2,ENABLE) )
10 #define CH395_CONFIG_GPIO_CLK() ( RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB,ENABLE) )
11 //设置使用的SPI
12 #define USE_SPI SPI2
13 //SPI_CS -- 连接模块SCS引脚
14 #define CH395_CS_PORT GPIOB
15 #define CH395_CS_PIN GPIO_Pin_12
16 //SPI_CLK -- 连接模块SCK引脚
17 #define CH395_CLK_PORT GPIOB
18 #define CH395_CLK_PIN GPIO_Pin_13
19 //SPI_MISO -- 连接模块SDO引脚
20 #define CH395_MISO_PORT GPIOB
21 #define CH395_MISO_PIN GPIO_Pin_14
22 //SPI_MOSI -- 连接模块SDI引脚
23 #define CH395_MOSI_PORT GPIOB
24 #define CH395_MOSI_PIN GPIO_Pin_15
25 //RST -- 连接模块RST引脚
26 #define CH395_RST_PORT GPIOA
27 #define CH395_RST_PIN GPIO_Pin_8
28 //TX -- 连接模块TX引脚
29 #define CH395_TX_PORT GPIOA
30 #define CH395_TX_PIN GPIO_Pin_3
31 //INT -- 连接模块INT引脚 (检测到中断信号之后再获取数据)
32 #define CH395_INT_PORT GPIOA
33 #define CH395_INT_PIN GPIO_Pin_0
34 /*****/
```

5.如果用户使用的其它型号的单片机,还需要替换后面程序

```
CH395CMD.C CH395INC.H CH395SPI.C usart.c delay.c timer.c main.c delay.h CH395SPLH
7 /*****配置GPIO (根据自己的修改) *****/
8 //时钟
9 #define CH395_CONFIG_SPI_CLK() ( RCC_APB1PeriphClockCmd( RCC_APB1Periph_SPI2,ENABLE) )
10 #define CH395_CONFIG_GPIO_CLK() ( RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB,ENAB
11 //设置使用的SPI
12 #define USE_SPI SPI2
13 //SPI_CS -- 连接模块SCS引脚
14 #define CH395_CS_PORT GPIOB
15 #define CH395_CS_PIN GPIO_Pin_12
16 //SPI_CLK -- 连接模块SCK引脚
17 #define CH395_CLK_PORT GPIOB
18 #define CH395_CLK_PIN GPIO_Pin_13
19 //SPI_MISO -- 连接模块SDO引脚
20 #define CH395_MISO_PORT GPIOB
21 #define CH395_MISO_PIN GPIO_Pin_14
22 //SPI_MOSI -- 连接模块SDI引脚
23 #define CH395_MOSI_PORT GPIOB
24 #define CH395_MOSI_PIN GPIO_Pin_15
25 //RST -- 连接模块RST引脚
26 #define CH395_RST_PORT GPIOA
27 #define CH395_RST_PIN GPIO_Pin_8
28 //TX -- 连接模块TX引脚
29 #define CH395_TX_PORT GPIOA
30 #define CH395_TX_PIN GPIO_Pin_3
31 //INT -- 连接模块INT引脚 (检测到中断信号之后再获取数据)
32 #define CH395_INT_PORT GPIOA
33 #define CH395_INT_PIN GPIO_Pin_0
34 /*****/
35
36 #define CH395_SPI_CS_LOW() (CH395_CS_PORT->BRR = CH395_CS_PIN) /*CS输出低*/
37 #define CH395_SPI_CS_HIGH() (CH395_CS_PORT->BSRR = CH395_CS_PIN) /*CS输出高*/
38
39 #define CH395_RST_PIN_HIGH() (CH395_RST_PORT->BSRR = CH395_RST_PIN) /*RST输出高*/
40 #define CH395_RST_PIN_LOW() (CH395_RST_PORT->BRR = CH395_RST_PIN) /*RST输出低*/
41
42 #define CH395_TX_PIN_HIGH() (CH395_TX_PORT->BSRR = CH395_TX_PIN) /*TX输出高*/
43 #define CH395_TX_PIN_LOW() (CH395_TX_PORT->BRR = CH395_TX_PIN) /*TX输出低*/
44
45 #define CH395_INT_PIN_INPUT() (CH395_INT_PORT->IDR & CH395_INT_PIN) /* 获取INT电平 */
46
47 /*****/
```

6.如果用户使用的其它型号的单片机,根据自己的情况修改引脚初始化

```
CH395INC.H  CH395SPI.C  usart.c  delay.c  timer.c  main.c  delay.h  CH395SPI.h

40  /*****
41  * Function Name   : CH395_Port_Init
42  * Description     : CH395端口初始化
43  *                 : 由于使用SPI读写时序,所以进行初始化
44  * Input          : None
45  * Output         : None
46  * Return         : None
47  *****/
48  void CH395_PORT_INIT( void )
49  {
50      //替换自己的端口初始化函数
51      SPI_InitTypeDef SPI_InitStructure;
52      GPIO_InitTypeDef GPIO_InitStructure;
53
54      /* 初始化SPI接口 */
55      CH395_CONFIG_SPI_CLK();
56      CH395_CONFIG_GPIO_CLK();
57
58      // Configure pins: SCK, MISO and MOSI
59      GPIO_InitStructure.GPIO_Pin = CH395_CLK_PIN;
60      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
61      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; /* 推拉输出备用功能 */
62      GPIO_Init( CH395_CLK_PORT, &GPIO_InitStructure );
63
64
65      GPIO_InitStructure.GPIO_Pin = CH395_MISO_PIN;
66      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; /* 推拉输出备用功能 */
67      GPIO_Init( CH395_MISO_PORT, &GPIO_InitStructure );
68
69
70      GPIO_Init( CH395_CS_PORT, &GPIO_InitStructure );
71
72      // Configure pins: TX
73      GPIO_InitStructure.GPIO_Pin = CH395_TX_PIN;
74      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
75      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; /* 推拉输出 */
76      GPIO_Init( CH395_TX_PORT, &GPIO_InitStructure );
77
78      // Configure pins: RST
79      GPIO_InitStructure.GPIO_Pin = CH395_RST_PIN;
80      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
81      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; /* 推拉输出 */
82      GPIO_Init( CH395_RST_PORT, &GPIO_InitStructure );
83
84      //TX low
85      CH395_TX_PIN_LOW();
86      /*CS high */
87      CH395_SPI_CS_HIGH();
88
89      /* SPI configuration */
90      SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex; /* SPI配置成两线的单向全双工通信 */
91      SPI_InitStructure.SPI_Mode = SPI_Mode_Master; /* SPI主机 */
92      SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b; /* SPI8位数据格式传输 */
93      SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low; /* 时钟低时活动 */
94      SPI_InitStructure.SPI_CPHA = SPI_CPHA_1Edge; /* 数据在时钟第二个边沿时捕获 */
95      SPI_InitStructure.SPI_NSS = SPI_NSS_Soft; /* 内部NSS信号由SSI控制 */
96      SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_4; /* 波特率预分频数为4 */
97      SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB; /* 传输时高位在前 */
98      SPI_InitStructure.SPI_CRCPolynomial = 7;
99
100     SPI_Init( USE_SPI, &SPI_InitStructure );
101
102     /* Enable SPI */
103     SPI_Cmd( USE_SPI, ENABLE );
104
105     /* 初始化中断引脚 */
106     GPIO_InitStructure.GPIO_Pin = CH395_INT_PIN;
107     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; /* 上拉输入 */
108     GPIO_Init( CH395_INT_PORT, &GPIO_InitStructure );
109 }
110
```

提示: SPI通信方式 CPOL=0, CPHA=1

CLK空闲状态为低电平,在第二个沿开始采集数据

7.如果用户使用的其它型号的单片机,根据自己的情况修改SPI函数

```
CH395INC.H  CH395SPI.C  usart.c  delay.c  timer.c  main.c  delay.h  CH395SPI.h

118 }
119
120 /*****
121 * Function Name : Spi395Exchange
122 * Description   : 硬件SPI输出且输入8个位数据
123 * Input        : d---将要送入到CH395的数据
124 * Output       : None
125 * Return       : SPI接收的数据
126 *****/
127 UINT8 Spi395Exchange( UINT8 d )
128 {
129     /* Loop while DR register in not empty */
130     // while(SPI_I2S_GetFlagStatus(USE_SPI, SPI_I2S_FLAG_TXE) == RESET);
131     while( ( USE_SPI->SR & SPI_I2S_FLAG_TXE ) == RESET );
132
133     /* Send byte through the SPI1 peripheral */
134     // SPI_I2S_SendData(USE_SPI, byte);
135     USE_SPI->DR = d;
136
137     /* Wait to receive a byte */
138     // while(SPI_I2S_GetFlagStatus(USE_SPI, SPI_I2S_FLAG_RXNE) == RESET);
139     while( ( USE_SPI->SR & SPI_I2S_FLAG_RXNE ) == RESET );
140
141     /* Return the byte read from the SPI bus */
142     // return SPI_I2S_ReceiveData(USE_SPI);
143     return( USE_SPI->DR );
144 }
145
```

8.根据下面的步骤测试即可

```
CH395CMD.C  CH395INC.H  CH395SPI.C  usart.c  delay.c  timer.c  main.c

18 #include "timer.h"
19 #include "iwdg.h"
20
21
22 #include "CH395SPI.H"
23 #include "CH395CMD.H"
24
25
26 int main(void)
27 {
28     NVIC_Configuration();
29     uart_init(115200); //串口初始化为115200
30     delay_init();
31     //初始化CH395使用的GPIO
32     CH395_PORT_INIT();
33     //复位 CH395
34     CH395_RST();
35
36     IWDG_Init(IWDG_Prescaler_256,156*10);
37     printf("\r\nstart\r\n");
38     while(1)
39     {
40         IWDG_Feed(); //喂狗
41
42         delay_ms(1000);
43         printf("CH395VER : %2x\n",CH395CMDGetVer());
44     }
45 }
46
47
```

9.关于文件

红框内的是通用文件,咱和模块通信调用的就是CH395CMD.C里面的函数


















然后CH395CMD.C里面的函数再调用绿框里面的SPI接口文件和模组进行通信.

蓝框是代表不同的通信方式封装的文件,当前并没有用到.也并未编写完整并不能使用...

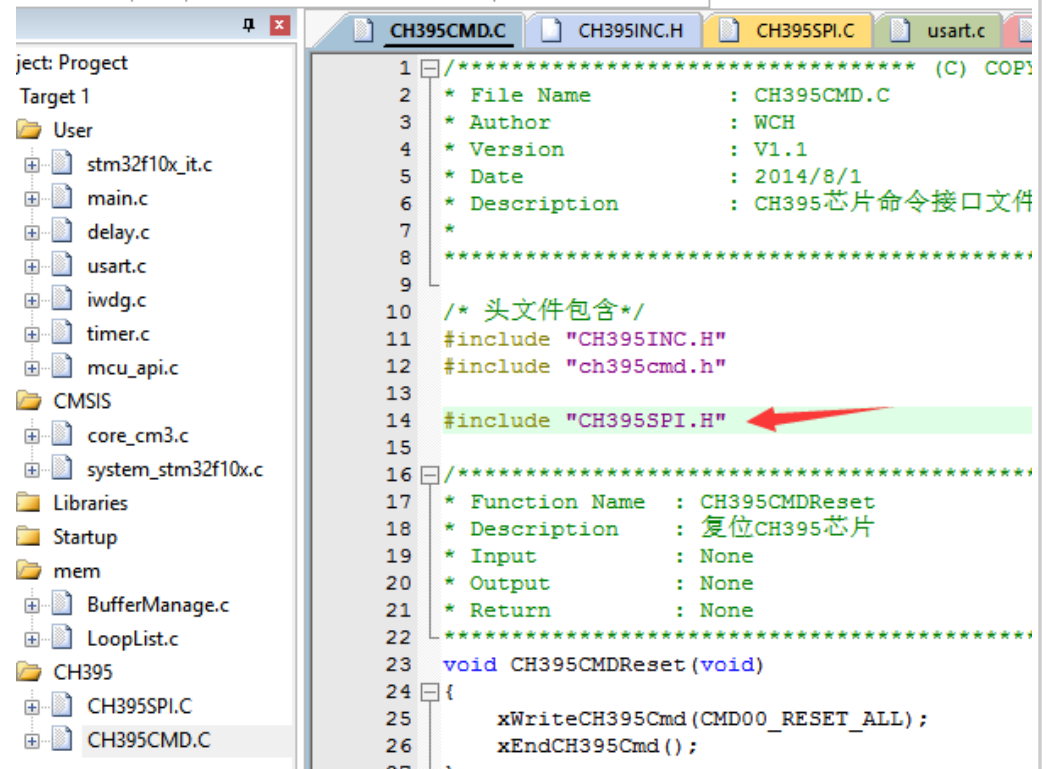
PARA : 并口

SPI_SW: 模拟SPI

UART: 串口

 CH395CMD.C	2021/6/4 10:36	C 文件
 CH395CMD.C~RF2ac6e89.TMP	2021/5/30 21:20	TMP 彡
 CH395CMD.C~RF2be9360.TMP	2021/6/3 23:56	TMP 彡
 CH395CMD.C~RF4b2b53f.TMP	2021/6/4 0:16	TMP 彡
 CH395CMD.C~RF4f5f7b1.TMP	2021/6/4 9:22	TMP 彡
 CH395CMD.C~RF6d6bb8f.TMP	2021/5/29 15:53	TMP 彡
 CH395CMD.C~RF28c31ca.TMP	2014/8/22 15:54	TMP 彡
 CH395CMD.C~RF301beee.TMP	2020/8/15 2:53	TMP 彡
 CH395CMD.H	2021/6/4 11:33	H 文件
 CH395INC.H	2020/8/15 1:43	H 文件
 CH395INC.H~RF28e42f7.TMP	2014/8/22 15:32	TMP 彡
 CH395PARA_SW.C	2021/5/30 15:23	C 文件
 CH395SPI.C	2021/6/4 17:25	C 文件
 CH395SPI.H	2021/6/4 17:18	H 文件
 CH395SPI_SW.C	2021/6/4 14:06	C 文件
 CH395UART.C	2021/6/4 13:57	C 文件
 CH395UART.H	2021/6/4 13:46	H 文件

如果使用其它通信方式,修改完通信方式文件以后,记得在CH395CMD.C里面包含接口函数头文件



分类: [CH395Q学习开发](#)

好文要顶

关注我

收藏该文



杨奉武

关注 - 1

粉丝 - 606

0

0

« 上一篇: [001-STM32+Air724UG基本控制篇\(华为云物联网平台\)--测试STM32+Air724UG\(4G模组\),Android,微信小程序等连接华为云物联网平台](#)

» 下一篇: [3-网络芯片CH395Q学习开发-芯片初始化,网线连接检测实验\(轮训和中断方式\)](#)

posted on 2021-06-04 17:28 杨奉武 阅读(16) 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]快捷提交

【推荐】百度智能云618年中大促，限时抢购，新老用户同享超值折扣
【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!
【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折
【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台
【推荐】华为应用软件专题日 | 生态市场企业特惠GO

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 阿里接上 “新大腿”
 - 完全无线化！小米工程师：明年或有厂商取消手机充电口
 - 华为携手极星汽车 举办首个中国车机黑客松大赛
 - HarmonyOS+骁龙865加持！华为MatePad 11上市时间首曝
 - 摩根大通：比特币反弹难以维持 期货市场已发出警报
- » 更多新闻...

历史上的今天：

2021-06-04 2-网络芯片CH395Q学习开发-学习资料说明,测试通信,获取硬件版本,代码移植说明

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, 加入群聊。