

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)  
ESP8266 SDK开发(34)  
ESP8266 SDK开发基础入门篇备份(30)  
GPRS Air202 LUA开发(11)  
HC32F460(华大单片机)物联网开发(17)  
HC32F460(华大单片机)学习开发(8)  
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)  
SLM130(NB-IoT)SDK学习开发(6)  
STM32+Air724UG(4G模组)物联网开发(43)  
STM32+BC26/260Y物联网开发(10)  
STM32+CH395Q(以太网)物联网开发(24)  
STM32+ESP8266(ZLESP8266A)物联网开发(1)  
STM32+ESP8266+AIR202/302远程升级方案(15)  
STM32+ESP8266+AIR202/302终端管理方案(6)  
STM32+ESP8266+Air302物联网开发(54)  
STM32+W5500物联网开发(14)  
STM32F103物联网开发(61)  
STM32F407物联网开发(14)  
STM32G070物联网开发(8)  
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)  
更多

#### 阅读排行榜

1. ESP8266使用详解(AT,LUA,SDK)(175861)
2. 1-安装MQTT服务器(Windows),并连接测试(110137)
3. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(71289)
4. ESP8266刷AT固件与nodemcu固件(68804)
5. 有人WIFI模块使用详解(40323)
6. C#中public与private与static(39216)
7. (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)(38398)
8. 关于TCP和MQTT之间的转换(37187)
9. android 之TCP客户端编程(34222)
10. (一)Lua脚本语言入门(33411)

#### 推荐排行榜

1. C#委托+回调详解(11)
2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(9)
3. 我的大学四年(7)
4. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
5. 关于stm32的正交解码(6)

#### 最新评论

1. Re:ESA2GJK1DH1K基础篇: 阿里云物联网平台: 使用阿里云物联网平台提供的自定义Topic通信控制(ESP8266,TCP透传指令)  
代码在哪里下载？

--题哦咯



## 说明

系统滴答定时器是系统内核(RISC-V内核)自带的定时器.

有人会问:那后面学到的那些通用定时器,高级定时器呢?

那些都是外设,然后依靠总线(导线)和它通信控制其实现具体功能.



又比如说:单片机有串口功能,实际上就是内核+硬件串口的组合,内核通过总线控制硬件串口通信.

## 随便看一个工程

1,官方给了一个不使用中断操作滴答定时器的例子

具体呢不做讨论了,直接使用就可以.

(对于初学者不必深究,即使深究最终也只是配置寄存器而已,先学会用,然后使用遇到问题时再解决)

## 2. Re:ESP8266 SDK开发: 综合篇-C#上位机串口通信控制ESP8266

认真拜阅读您的程序 收获很大 ;但我发现一处问题 就是您在串口接收函数中 没有将空闲标志清零 导致程序最终并没有通过空闲中断来处理 而是每隔10ms处理一次

--觉代疯骚

```
debug.c
22  */
23 void Delay_Init(void)
24 {
25     p_us = SystemCoreClock / 8000000;
26     p_ms = (uint16_t)p_us * 1000;
27 }
28
29 /**
30  * @fn      Delay_Us
31  *
32  * @brief   Microsecond Delay Time.
33  *
34  * @param   n - Microsecond number.
35  *
36  * @return  None
37  */
38 void Delay_Us(uint32_t n)
39 {
40     uint32_t i;
41
42     SysTick->SR &= ~(1 << 0);
43     i = (uint32_t)n * p_us;
44
45     SysTick->CMP = i;
46     SysTick->CTRL |= (1 << 4) | (1 << 5) | (1 << 0);
47
48     while((SysTick->SR & (1 << 0)) != (1 << 0))
49     ;
50     SysTick->CTRL &= ~(1 << 0);
51 }
52
53 /**
54  * @fn      Delay_Ms
55  *
56  * @brief   Millisecond Delay Time.
57  *
58  * @param   n - Millisecond number.
59  *
60  * @return  None
61  */
62 void Delay_Ms(uint32_t n)
63 {
64     uint32_t i;
65
66     SysTick->SR &= ~(1 << 0);
67     i = (uint32_t)n * p_ms;
68 }
```

## 现在看中断方式

```

main.c
10
11 #include "debug.h"
12 #include "ch32v30x.h"
13
14 volatile uint16_t SysTickCnt=0;
15
16 void SysTick_init(void)
17 {
18     /*配置中断优先级*/
19     NVIC_InitTypeDef NVIC_InitStructure = {0};
20     NVIC_InitStructure.NVIC_IRQChannel = SysTick_IRQn;
21     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //抢占式优先级
22     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; //响应式优先级
23     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //使能
24     NVIC_Init(&NVIC_InitStructure);
25
26     /*配置定时器*/
27     SysTick->CTLR = 0;
28     SysTick->SR = 0;
29     SysTick->CNT = 0;
30     SysTick->CMP = SystemCoreClock/1000; //后面的1000代表1000Hz (那就是1ms进一次中断)
31     SysTick->CTLR = 0xf;
32 }
33
34 __attribute__((interrupt("WCH-Interrupt-fast")))
35 void SysTick_Handler(void)
36 {
37     SysTick->SR = 0; //清除中断
38     SysTickCnt++;
39 }
40
41 int main(void)
42 {
43     NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //设置优先级分组为2
44     USART_Printf_Init(115200);
45     SysTick_init();
46     while(1)
47     {
48         if (SysTickCnt>=1000) {
49             SysTickCnt=0;
50             printf("1111111111\n");
51         }
52     }
53 }
54

```



```

#include "debug.h"

#include "ch32v30x.h"

volatile uint16_t SysTickCnt=0;

void SysTick_init(void)
{
    /*配置中断优先级*/
    NVIC_InitTypeDef NVIC_InitStructure = {0};
    NVIC_InitStructure.NVIC_IRQChannel = SysTick_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //抢占式优先级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; //响应式优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //使能
    NVIC_Init(&NVIC_InitStructure);

    /*配置定时器*/
    SysTick->CTLR = 0;
    SysTick->SR = 0;
    SysTick->CNT = 0;
    SysTick->CMP = SystemCoreClock/1000; //后面的1000代表1000Hz (那就是1ms)
    SysTick->CTLR = 0xf;
}

__attribute__((interrupt("WCH-Interrupt-fast")))
void SysTick_Handler(void)
{
    SysTick->SR = 0; //清除中断
    SysTickCnt++;
}

int main(void)

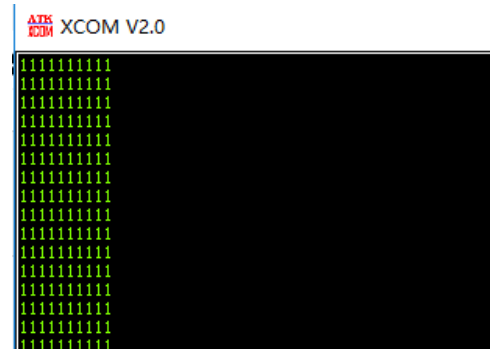
```

```

{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //设置优先级分组为2
    USART_Printf_Init(115200);
    SysTick_init();
    while(1)
    {
        if (SysTickCnt>=1000) {
            SysTickCnt=0;
            printf("1111111111\r\n");
        }
    }
}

```

## 下载测试会看到串口每隔1S打印一次数据



## 上面就是说完了, 我说一下我是如何写的上面的程序

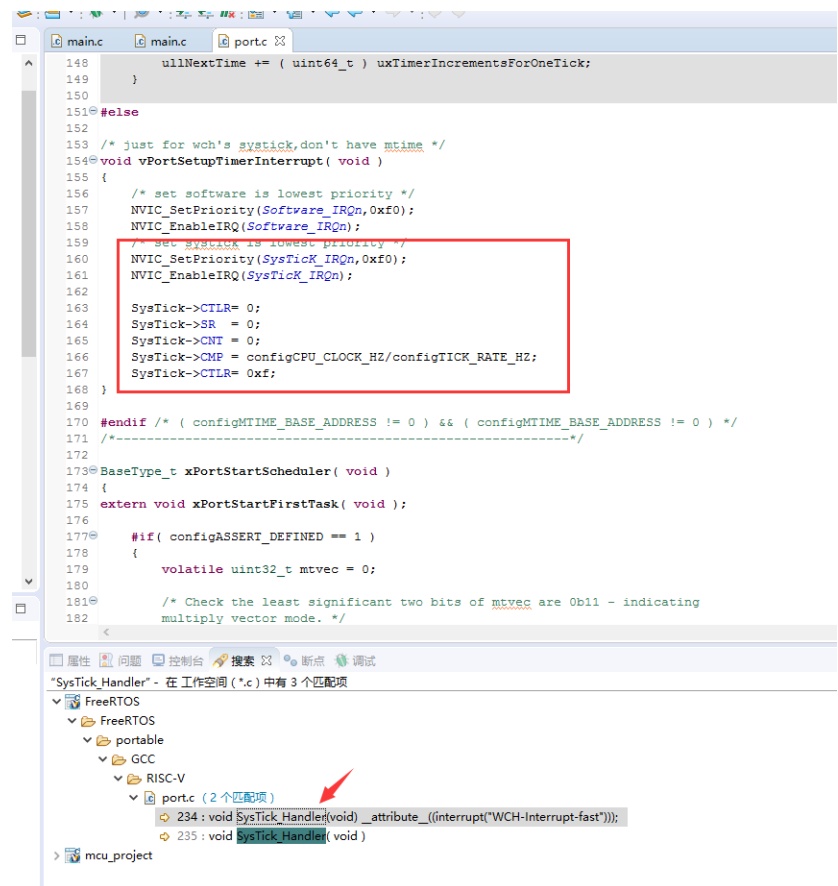
其实没有例程....

但是呢,官方提供了一个FreeRTOS(操作系统)的例子,

大家伙要知道,所有的操作系统都要依靠定时器来切换任务,一般都是使用系统滴答定时器

ADC	2
BKP	2
CAN	2
CRC	2
DAC	2
DMA	2
DVP	2
ETH	2
EXTI	2
FLASH	2
FPU	2
FreeRTOS	2
FSMC	~

## 我就全局搜索 SysTick\_Handler, 然后翻了翻就找到了



分类: CH32V307(WCH单片机)学习开发

好文要顶

关注我

收藏该文



杨奉武

关注 - 1

粉丝 - 779

0

0

« 上一篇: 移植升级底层包,把自己的用户程序增加上OTA功能

posted on 2022-05-18 23:49 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

提交评论 退出

[Ctrl+Enter快捷键提交]

【推荐】大学生技术公益开发训练营，让你的应用为公益发光发热

【推荐】阿里云数据库训练营，云数据库 MySQL 从入门到高阶

#### 编辑推荐：

- 闲置树莓派：种朵花然后做延时摄影吧
- 理解 ASP.NET Core - 发送 Http 请求(HttpClient)
- 探索 ABP 基础架构
- 万字长文深度剖析 RocketMQ 设计原理
- 戏说领域驱动设计（廿六）——再谈事务



#### 最新新闻：

- ACL 2022奖项公布！达摩院、华为等机构获奖，陈丹琦、杨笛一亦榜上有名
- 斗鱼发布2022年Q1财报：总营收17.96亿元，持续加大内容生态投入
- 用了NASA太空服技术的运动衣，知冷又知热
- Google 新园区正式开放，100%「纯电动」
- 如何编写数学上完美的软件

» 更多新闻...

#### 历史上的今天：

2021-05-18 04-STM32+Air724UG(4G模组)远程升级篇OTA(自建物联网平台)-STM3...

2017-05-18 STM32采集电阻触摸贴膜

Powered by:

博客园

Copyright © 2022 杨奉武

Powered by .NET 6 on Kubernetes



单片机,物联网,上位机,...

扫一扫二维码, 入群聊。