

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 - 688, 文章 - 0, 评论 - 310, 阅读 - 171万

### 导航

博客园

首页

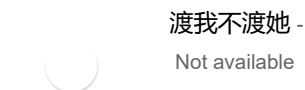
新随笔

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7个月

粉丝：598

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(1)  
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)

## 2-3-HC32F460(华大)+BC260Y(NB-IOT)基本控制篇(自建物联网平台)-基础外设例程-串口(基本使用)

```
<p><iframe name="ifd"
src="https://mnifdv.cn/resource/cnblogs/ZLIOTA_BC260Y/my.html" frameborder="0"
scrolling="auto" width="100%" height="1500"></iframe></p>
```

### 基本控制篇(自建MQTT服务器)[方案购买链接](#)

技术支持:  [加入QQ群](#) [技术论坛](#) [论武天地技术论坛](#)

- [开源必看教程:数据处理思想和程序架构](#)
- [基础开源教程:学习Android](#)
- [基础开源教程:微信小程序开发入门篇](#)
- [基础开源教程:MySQL数据库应用教程](#)
- [基础开源教程:硬件基本知识和典型应用](#)

以上为基础公开教程,基础公开教程全部开源,请自行学习！

### 基本控制篇篇章：[HC32F460\(华大\)+BC260Y\(NB-IOT\)基本控制篇\(自建物联网平台\)](#)

- [1-硬件使用说明](#)
- [2-1-基础外设例程-工程模板使用说明](#)
- [2-2-基础外设例程-GPIO输出高低电平](#)
- [2-3-基础外设例程-GPIO引脚电平检测](#)
- 
- 
- 
- 
- 
- 
- 
-

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(42)  
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: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
2. Re:03-STM32+Air724UG  
远程升级篇OTA(阿里云物联  
网平台)-STM32+Air724UG  
使用阿里云物联网平台OTA  
远程更新STM32程序  
@xxJian 和设备名称没有关  
系,一个产品下的设备都是使  
用一个固件...  
--杨奉武

#### 阅读排行榜

1. ESP8266使用详解(AT,LUA,  
SDK)(171890)

## 说明

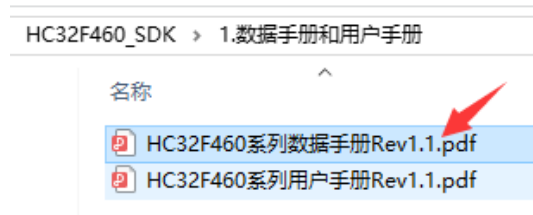
基础外设例程是提供给已经开发过M0或M1或M3或M4等ARM内核单片机的开发人员!

例程精简扼要, 力求让开发人员快速使用华大单片机做项目!

这节说一下串口.

## 先提示一个事情

### 1.打开数据手册



2. 1-安装MQTT服务器(Windows),并连接测试(95913)
3. ESP8266刷AT固件与node mcu固件(63547)
4. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(62048)
5. 有人WIFI模块使用详解(37998)
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)(35239)
7. 关于TCP和MQTT之间的转换(31893)
8. android 之TCP客户端编程(31134)
9. android客服端+eps8266+单片机+路由器之远程控制系统(31089)
10. C#中public与private与static(30660)

## 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)

# 2.第33页(我以自己使用的HC32F460JETA为例)

## 主要说一下后面的 Func\_Grp

LCF_P180	VREG_A30	LCF_H4	CFM_B	LDSP_Cpin_8	Pin Name	Analog	ERRQWK_UP	TRACE/ITAG_PWD	Func0	Func1	Func2	Func3	Func4	Func5	Func6	Func7	Func8	Func9	Func10	Func11	Func12	Func13	Func14	Func15	Func16	Func17-48	
									GPO	other	TIM4		TIMA	TIMA	TIMA	EMBLTIMA	USART2/PWD_SPI	KEY	SDIO	USBFSDS	-	-	-	EVENTPT	EVENTOUT	-	Common action Fuses
1	B2	-	-	-	PE2		ERRQ2	TRACCLK	GPO				TIMA_L_PWM5			USART3_CK								EVENTOUT		Func_Grp2	
2	A1	-	-	-	PE3		ERRQ3	TRACDATA0	GPO				TIMA_L_PWM6			USART4_CK								EVENTOUT		Func_Grp2	
3	B1	-	-	-	PE4		ERRQ4	TRACDATA1	GPO				TIMA_L_PWM7											EVENTOUT		Func_Grp2	
4	C2	-	-	-	PE5		ERRQ5	TRACDATA2	GPO				TIMA_L_PWM8											EVENTOUT		Func_Grp2	
5	D2	-	-	-	PE6		ERRQ6	TRACDATA3	GPO				TIMA_L_PWM9											EVENTOUT		Func_Grp2	
6	E2	1	1	1	PE2		ERRQ2		GPO	FCMBEF	TIM2_CK1K		TIMA_L_PWM7		EMBL_N4			SDIO2_D4	I2S3_CKCK					EVENTOUT		Func_Grp2	
7	C1	2	2	2	PC13		ERRQ13		GPO	RTC_OUT			TIMA_L_PWM8					SDIO2_CK	I2S3_MCK			EVENTP13				Func_Grp2	
8	D1	3	3	3	PC14	XTAL32_0_RT	ERRQ14		GPO				TIMA_L_PWM9									EVENTP14					
9	E1	4	4	4	PC15	XTAL32_1_N	ERRQ15		GPO				TIMA_L_PWM6									EVENTP15					
10	F2	-	-	-	VSS																						
11	G2	-	-	-	VCC																						
12	F1	5	5	5	PB0	XTAL_3N	ERRQ0		GPO				TIMA_L_PWM9														
13	G1	6	6	6	PB1	XTAL_OUT	ERRQ1		GPO				TIMA_L_PWM8														
14	H2	7	7	7	SMBT																						
15	H1	8	8	-	PC0	ADC12_N0MCP1_NP3	ERRQ0		GPO				TIMA_L_PWM9					SDIO2_D5				EVENTP00	EVENTOUT			Func_Grp1	
16	J2	9	9	-	PC1	ADC12_N01	ERRQ1		GPO				TIMA_L_PWM6					SDIO2_D6				EVENTP01	EVENTOUT			Func_Grp1	
17	J1	10	10	-	PC2	ADC12_N02	ERRQ2		GPO				TIMA_L_PWM7		EMBL_N3			SDIO2_D7				EVENTP02	EVENTOUT			Func_Grp1	
18	K2	11	-	-	PC3	ADC12_N03MCP1_NM2	ERRQ3		GPO				TIMA_L_PWM8					SDIO2_NP				EVENTP03	EVENTOUT			Func_Grp1	
19	-	-	-	-	VCC																						
20	J1	12	11	8	AVSS																						
-	K1	-	-	-	VREFL																						
21	L1	-	-	-	VREFH																						
22	M1	13	12	9	AVCC																						
23	L2	14	13	10	PA0	ADC12_N04MCP1_NP0	ERRQ04	ERRQ04_L0P_0	GPO			TIM2_C0KH			TIMA_L_PWM10TIMA2_CK1_CK4			TIMA2_TRG	SPI1_SS1			SDIO2_D4			EVENTP00	EVENTOUT	Func_Grp1

# 3.然后找到38页

	Func32	Func33	Func34	Func35	Func36	Func37	Func38	Func39	Func40	Func41	Func42	Func43	Func44	Func45	Func46	Func47
Func_Grp1	USART1_TX	USART1_RX	USART1_RTS	USART1_CTS	USART2_TX	USART2_RX	USART2_RTS	USART2_CTS	SPI1_MO	SPI1_MISO	SPI1_SS0	SPI1_SC	SPI2_MO	SPI2_MISO	SPI2_SS0	SPI2_SC
Func_Grp2	USART3_TX	USART3_RX	USART3_RTS	USART3_CTS	USART4_TX	USART4_RX	USART4_RTS	USART4_CTS	SPI3_MO	SPI3_MISO	SPI3_SS0	SPI3_SC	SPI4_MO	SPI4_MISO	SPI4_SS0	SPI4_SC

	Func48	Func49	Func50	Func51	Func52	Func53	Func54	Func55	Func56	Func57	Func58	Func59	Func60	Func61	Func62	Func63
Func_Grp1	I2C1_SDA	I2C1_SCL	I2C2_SDA	I2C2_SCL	I2S1_SD	I2S1_SDIN	I2S1_WS	I2S1_CK	I2S2_SD	I2S2_SDI_N	I2S2_SS0	I2S2_CK				
Func_Grp2	I2C3_SDA	I2C3_SCL	CAN_TxD	CAN_RxD	I2S3_SD	I2S3_SDIN	I2S3_WS	I2S3_CK	I2S4_SD	I2S4_SDI_N	I2S4_WS	I2S4_CK				

表 2-2 Func32~63 表

# 4.列如:PC0

## PC0后面写的是 Func\_Grp1

## 那么这个引脚可以作为这些功能使用

	Func32	Func33	Func34	Func35	Func36	Func37	Func38	Func39	Func40	Func41	Func42	Func43	Func44	Func45	Func46	Func47
Func_Grp1	USART1_TX	USART1_RX	USART1_RTS	USART1_CTS	USART2_TX	USART2_RX	USART2_RTS	USART2_CTS	SPI1_MO	SPI1_MISO	SPI1_SS0	SPI1_SC	SPI2_MO	SPI2_MISO	SPI2_SS0	SPI2_SC

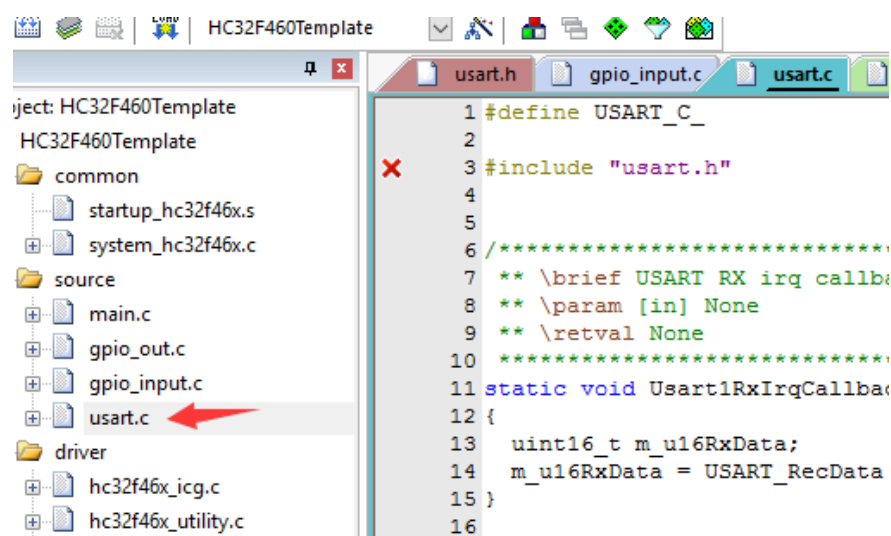
	Func48	Func49	Func50	Func51	Func52	Func53	Func54	Func55	Func56	Func57	Func58	Func59
Func_Grp1	I2C1_SDA	I2C1_SCL	I2C2_SDA	I2C2_SCL	I2S1_SD	I2S1_SDIN	I2S1_WS	I2S1_CK	I2S2_SD	I2S2_SDI_N	I2S2_WS	I2S2_CK

5.单片机的USART ,SPI, I2C, I2S, 不是固定的引脚,自己根据表格去指定

6.华大给了好几个串口的例子

HC32F460_SDK > 驱动库及样例 > hc32f46x_ddl > example > usart >			
名称	修改日期	类型	
clk_sync_dma_rx	2020/2/28 15:48	文件夹	
clk_sync_dma_tx	2020/2/28 15:48	文件夹	
clk_sync_hw_flow_ctrl_rx_rts	2020/2/28 15:48	文件夹	
clk_sync_hw_flow_ctrl_tx_cts	2020/2/28 15:48	文件夹	
clk_sync_irq_rx	2020/2/28 15:48	文件夹	
clk_sync_irq_tx	2020/2/28 15:48	文件夹	
sc_rx_atr_int	2020/2/28 15:48	文件夹	
uart_dma_rx_tx	2020/2/28 15:48	文件夹	
uart_hw_flow_ctrl_rx_rts	2020/2/28 15:48	文件夹	
uart_hw_flow_ctrl_tx_cts	2020/2/28 15:48	文件夹	
uart_irq_rx_tx	2020/2/28 15:48	文件夹	
uart_irq_rx_tx_timeout	2020/2/28 15:48	文件夹	
uart_polling_rx_tx	2020/2/28 15:48	文件夹	

## 基本使用



1.串口基本的初始化

usart.c



```
#define USART_C_

#include "usart.h"

/*****
** \brief USART RX irq callback function.//串口接收中断函数
** \param [in] None
** \retval None
*****/
static void Usart1RxIrqCallback(void)
{
    uint16_t m_u16RxData;
    m_u16RxData = USART_RecData(M4_USART1); //获取串口接收的数据
}

/*****
** \brief USART RX error irq callback function.(串口接收错误中断处理函数)
** \param [in] None
** \retval None
*****/
static void Usart1ErrIrqCallback(void)
{
    if (Set == USART_GetStatus(M4_USART1, UsartFrameErr)) { USART_ClearStatus(1)
    else {}

    if (Set == USART_GetStatus(M4_USART1, UsartParityErr)) {USART_ClearStatus
    else {}

    if (Set == USART_GetStatus(M4_USART1, UsartOverrunErr)) {USART_ClearStatu
    else {}
}

/*****
** \brief 串口初始化
** \param [in] None
** \retval None
*****/
void usart_init(void)
{
    en_result_t enRet = Ok;
    stc_irq_regi_conf_t stcIrqRegiCfg;

    /*配置串口使用的时钟和基本通信配置*/
    const stc_usart_uart_init_t stcInitCfg = {
        UsartIntClkCkNoOutput, //使用内部时钟源,不需要在其时钟输出IO上输出通信的时钟信
        UsartClkDiv_1,         //时钟不分频
        UsartDataBits8,         //一个字节数据用8位数据位表示
        UsartDataLsbFirst,      //先传输低位
        UsartOneStopBit,        //停止位1位
        UsartParityNone,        //无奇偶校验
        UsartSamleBit8,         //每次传输8位(1字节),也可以传输 UsartSamleBit16(16
        UsartStartBitFallEdge,
        UsartRtsEnable,         //使能RTS (串口开始传输前让RTS输出一个高脉冲信号)
    };

    /* Enable peripheral clock *//*打开时钟*/
    PWC_Fcg1PeriphClockCmd(PWC_FCG1_PERIPH_USART1 | PWC_FCG1_PERIPH_USART2 |
    PWC_FCG1_PERIPH_USART3 | PWC_FCG1_PERIPH_USART4, Enable);
```

```

/* Initialize USART IO */ /*配置相应的IO作为串口的TX,RX引脚*/
PORT_SetFunc(USART1_RX_PORT, USART1_RX_PIN, Func_Usart1_Rx, Disable);
PORT_SetFunc(USART1_TX_PORT, USART1_TX_PIN, Func_Usart1_Tx, Disable);

/* Initialize UART */ /*初始化串口配置*/
enRet = USART_UART_Init(M4_USART1, &stcInitCfg);
if (enRet != Ok)while (1);
/* Set baudrate */ /*设置串口波特率*/
enRet = USART_SetBaudrate(M4_USART1, USART1_BAUDRATE);
if (enRet != Ok)while (1);

/* Set USART RX IRQ */ /*设置串口接收中断*/
stcIrqRegiCfg.enIRQn = Int000_IRQn; //设置中断优先级
stcIrqRegiCfg.pfnCallback = &Usart1RxIrqCallback; //设置中断回调函数
stcIrqRegiCfg.enIntSrc = INT_USART1_RI; //中断名称
enIrqRegistration(&stcIrqRegiCfg);
NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);

/* Set USART RX error IRQ */ /*设置串口接收错误中断*/
stcIrqRegiCfg.enIRQn = Int001_IRQn;
stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
stcIrqRegiCfg.enIntSrc = INT_USART1_EI;
enIrqRegistration(&stcIrqRegiCfg);
NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);

/*Enable RX && RX interrupt function && UsartTx*/
USART_FuncCmd(M4_USART1, UsartRx, Enable); //使能接收
USART_FuncCmd(M4_USART1, UsartRxInt, Enable); //使能接收中断
USART_FuncCmd(M4_USART1, UsartTx, Enable); //使能发送
}

```



## usart.h



```

#ifndef USART_H_
#define USART_H_

#ifndef USART_C_
#define USART_C_ extern
#else
#define USART_C_
#endif

#include "hc32_dd1.h"

/* USART1 baudrate definition */
#define USART1_BAUDRATE (115200ul)
/* USART1 TX Port/Pin definition */
#define USART1_TX_PORT (PortA)
#define USART1_TX_PIN (Pin09)
/* USART1 RX Port/Pin definition */

```

```

#define USART1_RX_PORT (PortA)
#define USART1_RX_PIN (Pin10)

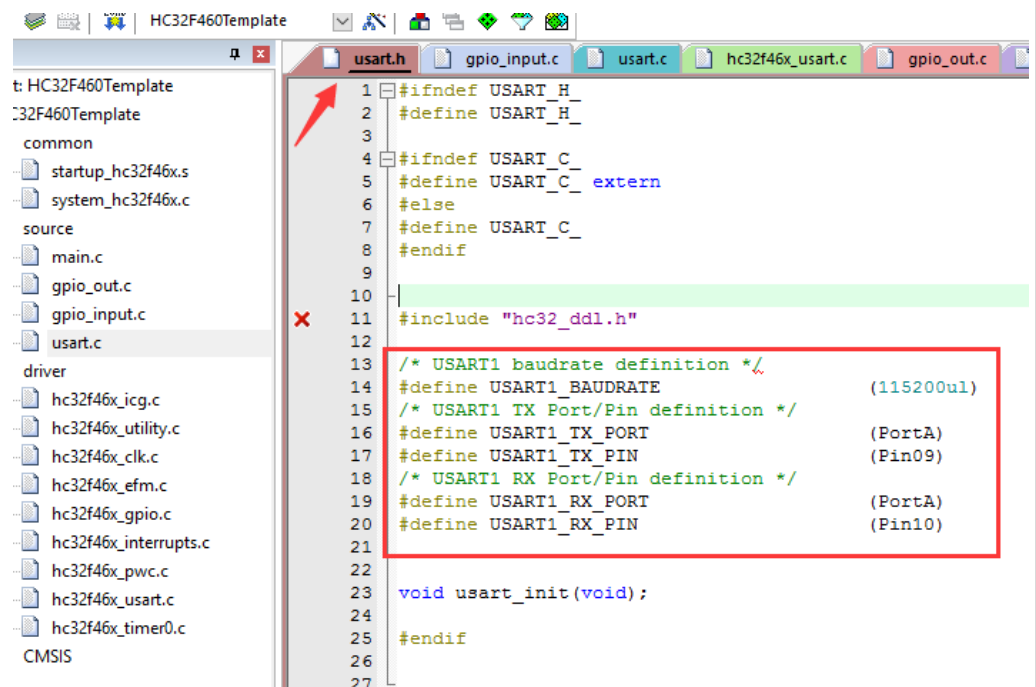
void usart_init(void);

#endif

```

## 2.在头文件里面设置串口1波特率,还有使用哪个引脚作为串口1的发送和接收引脚

我配置了波特率为115200; PA9作为串口1的发送数据引脚; PA10作为串口1的串口接收引脚



## 3.串口基本的配置

```

34 /*****
35 ** \brief 串口初始化
36 ** \param [in] None
37 ** \retval None
38 *****/
39 void usart_init(void)
40 {
41     en_result_t enRet = Ok;
42     stcIrqRegi_conf_t stcIrqRegiCfg;
43
44     /* Enable peripheral clock */ /*打开时钟*/
45     PWC_Fcg1PeriphClockCmd(PWC_FCG1_PERIPH_USART1 | PWC_FCG1_PERIPH_USART2 | \
46     PWC_FCG1_PERIPH_USART3 | PWC_FCG1_PERIPH_USART4, Enable);
47
48     /* Initialize USART IO */ /*配置相应的Io作为串口的TX,RX引脚*/
49     PORT_SetFunc(USART1_RX_PORT, USART1_RX_PIN, Func_Usart1_Rx, Disable);
50     PORT_SetFunc(USART1_TX_PORT, USART1_TX_PIN, Func_Usart1_Tx, Disable);
51
52     /*配置串口使用的时钟和基本通信配置*/
53     const stc_usart_uart_init_t stcInitCfg = {
54         UsartIntClkCkNoOutput, //使用内部时钟源,不需要在其时钟输出Io上输出通信的时钟信号
55         UsartClkDiv_1, //时钟不分频
56         UsartDataBits8, //一个字节数据用8位数据位表示
57         UsartDataLsbFirst, //先传输低位
58         UsartOneStopBit, //停止位1位
59         UsartParityNone, //无奇偶校验
60         UsartSamleBit8, //每次传输8位(1字节),也可以传输 UsartSamleBit16(16位,2字节)
61         UsartStartBitFallEdge,
62         UsartRtsEnable, //使能RTS (串口开始传输前让RTS输出一个高脉冲信号)
63     };
64     /* Initialize UART */ /*初始化串口配置*/
65     enRet = USART_UART_Init(M4_USART1, &stcInitCfg);
66     if (enRet != Ok) while (1);
67     /* Set baudrate */ /*设置串口波特率*/
68     enRet = USART_SetBaudrate(M4_USART1, USART1_BAUDRATE);
69     if (enRet != Ok) while (1);
70
71     /* Set USART RX IRQ */ /*设置串口接收中断*/
72     stcIrqRegiCfg.enIRQn = Int000_IRQn; //设置中断优先级
73     stcIrqRegiCfg.pfnCallback = &Usart1RxIrqCallback; //设置中断回调函数
74     stcIrqRegiCfg.enIntSrc = INT_USART1_RI; //中断名称
75     enIrqRegistration(&stcIrqRegiCfg);

```

## 4,配置串口中断接收回调函数

注意:中断优先级有145个: Int000\_IRQn, Int001\_IRQn, Int002\_IRQn  
.... Int142\_IRQn, Int143\_IRQn;数字越小,优先级别越高

```

67 /* Set baudrate */ /*设置串口波特率*/
68 enRet = USART_SetBaudrate(M4_USART1, USART1_BAUDRATE);
69 if (enRet != Ok) while (1);
70
71 /* Set USART RX IRQ */ /*设置串口接收中断*/
72 stcIrqRegiCfg.enIRQn = Int000_IRQn; //设置中断优先级
73 stcIrqRegiCfg.pfnCallback = &Usart1RxIrqCallback; //设置中断回调函数
74 stcIrqRegiCfg.enIntSrc = INT_USART1_RI; //中断名称(串口1接收中断)
75 enIrqRegistration(&stcIrqRegiCfg);
76 NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
77 NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
78 NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
79
80 /* Set USART RX error IRQ */ /*设置串口接收错误中断*/
81 stcIrqRegiCfg.enIRQn = Int001_IRQn; //中断优先级
82 stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;

```



```

uart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32
4
5
6 /*****
7  ** \brief USART RX irq callback function.//串口接收中断函数
8  ** \param [in] None
9  ** \retval None
10 *****/
11 static void Usart1RxIrqCallback(void)
12 {
13     uint16_t m_u16RxData;
14     m_u16RxData = USART_RecData(M4_USART1); //获取串口接收的数据
15 }
16

```

## 4.配置串口接收错误中断回调函数

这个应该是固定处理形式,然后应该可以去掉.

```

uart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f46x_interrupts
76 NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
77 NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
78 NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
79
80 /* Set USART RX error IRQ */ /*设置串口接收错误中断*/
81 stcIrqRegiCfg.enIRQn = Int001_IRQn; //中断优先级
82 stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
83 stcIrqRegiCfg.enIntSrc = INT_USART1_EI; //中断名称(串口1接收错误中断)
84 enIrqRegistration(&stcIrqRegiCfg);
85 NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
86 NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
87 NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
88

```

```

uart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f46x_interrupts.h  hc32f46x.h  hc32f46x_usart.h  gpio_out
16
17 /*****
18  ** \brief USART RX error irq callback function.(串口接收错误中断处理函数)
19  ** \param [in] None
20  ** \retval None
21  *****/
22 static void Usart1ErrIrqCallback(void)
23 {
24     if (Set == USART_GetStatus(M4_USART1, UsartFrameErr)) { USART_ClearStatus(M4_USART1, UsartFrameErr);}
25     else{}
26     if (Set == USART_GetStatus(M4_USART1, UsartParityErr)) {USART_ClearStatus(M4_USART1, UsartParityErr);}
27     else{}
28     if (Set == USART_GetStatus(M4_USART1, UsartOverrunErr)) {USART_ClearStatus(M4_USART1, UsartOverrunErr);}
29     else{}
30 }
31
32

```

## 5.使能

注意!要一句一句写!不要用 USART\_FuncCmd(M4\_USART1, UsartRx | XXXX | XXXX, Enable); 不可以这样子用

注意!要一句一句写!不要用 USART\_FuncCmd(M4\_USART1, UsartRx | XXXX | XXXX, Enable); 不可以这样子用

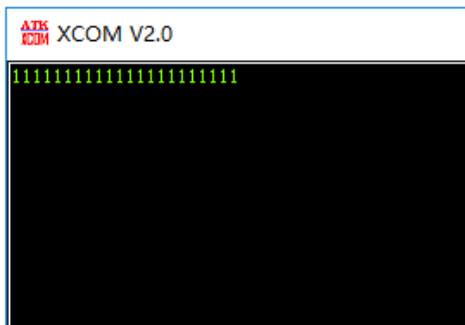
注意!要一句一句写!不要用 USART\_FuncCmd(M4\_USART1, UsartRx | XXXX | XXXX, Enable); 不可以这样子用

```
usart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f46x_inte
82  stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
83  stcIrqRegiCfg.enIntSrc = INT_USART1_EI;//中断名称(串口接收错误中断
84  enIrqRegistration(&stcIrqRegiCfg);
85  NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
86  NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
87  NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
88
89  /*Enable RX && RX interupt function && UsartTx*/
90  USART_FuncCmd(M4_USART1, UsartRx, Enable);//使能接收
91  USART_FuncCmd(M4_USART1, UsartRxInt, Enable);//使能接收中断
92  USART_FuncCmd(M4_USART1, UsartTx, Enable);//使能发送
93 }
94
```

## 6.测试每隔一段时间发送一个字符1

```
usart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f46
1  #include "hc32_ddl.h"
2
3  #include "gpio_out.h"
4  #include "gpio_input.h"
5  #include "usart.h"
6
7
8  int32_t main(void)
9  {
10     gpio_out_init();
11     gpio_input_init();
12
13     usart_init();
14     while(1)
15     {
16         Ddl_Delay1ms(500);
17         //发送字符1
18         USART_SendData(M4_USART1, '1');
19         //等待串口发送完成
20         while (Reset == USART_GetStatus(M4_USART1, UsartTxEmpty));
21     }
22 }
23
```

```
//发送字符1
USART_SendData(M4_USART1, '1');
//等待串口发送完成
while (Reset == USART_GetStatus(M4_USART1, UsartTxEmpty));
```



7.对于一般的用户接收数据呢用户可以按照自己的习惯去写就可以了

```
usart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f4
7  ** \brief USART RX irq callback function.//串口接收中断函数
8  ** \param [in] None
9  ** \retval None
10 *****
11 static void Usart1RxIrqCallback(void)
12 {
13     uint16_t m_ul6RxData;
14     m_ul6RxData = USART_RecData(M4_USART1); //获取串口接收的数据
15 }
16
```

## 增加空闲中断 (空闲中断需要用到定时器,在后面的章节介绍)

分类: [HC32F460\(华大\)](#) + [BC260Y\(NB-IOT\)](#) 物联网开发

好文要顶

关注我

收藏该文



杨奉武

关注 - 1

粉丝 - 598

0

0

« 上一篇: [物联网开发:物联网卡,NB-IOT卡经销商](#)

posted on 2021-06-01 09:09 杨奉武 阅读(3) 评论(0) 编辑 收藏

[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】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载!

【推荐】开说正事 | 这个六一，回到最初的美好，只需两个功能

【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

#### 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：

- 微信再次升级青少年模式：视频号专属内容池上线
  - 618开门红 华为FreeBuds 4无线耳机开卖：舒适与降噪兼得
  - CDPR利润大跌
  - 6月2日见！华为鸿蒙OS提前送惊喜：快看看你是否能升级了
  - 《黑神话：悟空》模型渲染师沉迷虚幻5 分享更多场景
- » 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, 加入群聊。