

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 - 778, 文章 - 0, 评论 - 325, 阅读 - 193万

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：6年
 粉丝：669
 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

[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\(12\)](#)
[CH579M学习开发\(8\)](#)
[ESP32学习开发\(20\)](#)
[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\(13\)](#)

5-HC32F460(华大单片机)-串口(基本使用)

<p>
 <iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnHC32F460"
 frameborder="0" scrolling="auto" width="100%" height="1500">
 </iframe>
 </p>

HC32F460(华大单片机)学习开发

开发板原理

图:<https://mnifdv.cn/resource/cnblogs/LearnHC32F460/HC32F460.PDF>

资料源码下载链接:<https://github.com/yangfengwu45/learnHC32F460.git>

- [1-硬件使用说明](#)
- [2-工程模板使用说明](#)
- [3-GPIO输出高低电平](#)
- [4-GPIO引脚电平检测](#)
- [5-串口\(基本使用\)](#)
-
-
-
-
-
-
-

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(12)
感想(6)
更多

阅读排行榜

1. ESP8266使用详解(AT,LUA, SDK)(173575)
2. 1-安装MQTT服务器(Windo ws),并连接测试(102377)
3. 用ESP8266+android,制作 自己的WIFI小车(ESP8266篇) (66199)
4. ESP8266刷AT固件与node mcu固件(65995)
5. 有人WIFI模块使用详解(390 18)
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)(36513)
7. 关于TCP和MQTT之间的转 换(34471)
8. C#中public与private与stat ic(34391)
9. android 之TCP客户端编程 (32506)
10. android客服端+eps8266 +单片机+路由器之远程控制系 统(31523)

说明

HC32F460基础例程源码下载链

接: <https://github.com/yangfengwu45/learnHC32F460.git>

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

HC32F460_SDK > 1.数据手册和用户手册

名称

- HC32F460系列数据手册Rev1.1.pdf
- HC32F460系列用户手册Rev1.1.pdf

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)

1. Re:用ESP8266+android,
制作自己的WIFI小车
(Android 软件)
百度网盘都失效了
--ghggaojian

--zsw1997

主要说一下后面的 Func_Grp

LOGP ID	VFRG ASB	LOGP ID	OFDA	LOGP ID	File Name	Analysis	ERRORW UP	TRACE/TCG S/WO	Func0	Func1	Func2	Func3	Func4	Func5	Func6	Func7	Func8	Func9	Func10	Func11	Func12	Func13	Func14	Func15	Func16	Func17	Func18	Func19
									GPO	other	TIMA	TIMA	TIMA	TIMA	EMBLTIMA	STARTSPVO SW	KEY	SDIO	USBFSDS	-	-	-	EVENTP	EVENTOUT	-			
1	02	1	-	-	FE2		ERRQ2	TRACEDATA 2	GPO				TIMA_PPMW 6			NSART_C_K								EVENTOUT				Func18-AS
2	A1	-	-	-	FE3		ERRQ3	TRACEDATA 3	GPO				TIMA_PPMW 6			NSART_C_K								EVENTOUT				Func19
3	B1	-	-	-	FE4		ERRQ4	TRACEDATA 4	GPO				TIMA_PPMW 7											EVENTOUT				Func20
4	C2	-	-	-	FE5		ERRQ5	TRACEDATA 5	GPO				TIMA_PPMW 8											EVENTOUT				Func21
5	D0	-	-	-	FE6		ERRQ6	TRACEDATA 6	GPO															EVENTOUT				Func22
6	E2	5	1	1	FE2		ERRQ2		GPO	PCMBFF	TIMA_C_K		TIMA_PPMW 7		EMBL_NG		NRQD_H	NSA_C_K							EVENTOUT			Func23
7	F1	2	2	2	FE3		ERRQ3		GPO	NSC_DET			TIMA_PPMW 8				NRQD_C_K	NSA_MCK					EVENTP13					
8	D1	3	3	3	FE4	XTAL32_2	ERRQ4		GPO				TIMA_PPMW 8											EVENTP14				
9	E1	4	4	4	FE5	XTAL32_3	ERRQ5		GPO				TIMA_PPMW 9											EVENTP15				
10	F2	-	-	-	FE6																							
11	G2	-	-	-	FE7																							
12	F1	5	5	5	FE6	XTAL_N	ERRQ6		GPO				TIMA_PPMW 9															
13	G1	6	6	6	FE7	XTAL_OG 7	ERRQ7		GPO				TIMA_PPMW 9															
14	H2	7	7	7	FE8																							
15	I1	8	8	8	FE9	AXC12_R03	ERRQ8		GPO				TIMA_PPMW 6					NRQD_H					EVENTP16	EVENTOUT				Func24
16	J2	9	9	9	FE10	AXC12_R03	ERRQ9		GPO				TIMA_PPMW 7					NRQD_H					EVENTP17	EVENTOUT				Func25
17	K1	10	10	10	FE11	AXC1_R02	ERRQ10		GPO				TIMA_PPMW 7		EMBL_NG								EVENTP18	EVENTOUT				Func26
18	K2	11	-	-	FE12	AXC1_R03	ERRQ11		GPO				TIMA_PPMW 8					NRQD_H					EVENTP19	EVENTOUT				Func27
19	-	-	-	-	FE13																							
20	L1	12	11	8	FE14																							
21	L1	-	-	-	FE15																							
22	M1	13	12	9	FE16																							
23	L2	14	13	10	FE17	AXC1_R03	ERRQ12		GPO			TIMA_C_K	TIMA_PPMW 10		TIMA_TRO	NSP1_S3		NRQD_H					EVENTP20	EVENTOUT				Func28

	Func32	Func33	Func34	Func35	Func36	Func37	Func38	Func39	Func40	Func41	Func42	Func43	Func44	Func45	Func46	Func47
Func_Grp 1	USART1_RX	USART1_RX	USART1_R_TS	USART1_C_TS	USART2_TX	USART2_RX	USART2_TS	USART2_C_TS	SP1_M0	SP1_M5	SP1_S0	SP1_SC_SI	SP2_M0	SP2_M5	SP2_S0	SP2_SC_K
Func_Grp 2	USART1_TX	USART1_RX	USART1_R_TS	USART1_C_TS	USART4_TX	USART4_RX	USART4_TS	USART4_C_TS	SP1_M0_SI	SP1_M5_O	SP1_S0_K	SP1_SC_SI	SP4_M0	SP4_M5_O	SP4_S0_K	SP4_SC_K

表 2-2 Func32~63 表

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_Grp	USART1_	USART1_	USART1_R	USART1_C	USART2_	USART2_	USART2_R	USART2_C	SP1_M0	SP1_MMS	SP1_S80	SP1_SC	SP2_M0	SP2_MS	SP2_S80	SP2_SC
1	TX	RX	TS	TS	TX	RX	TS	TS	SI	O		K	SI	O		K

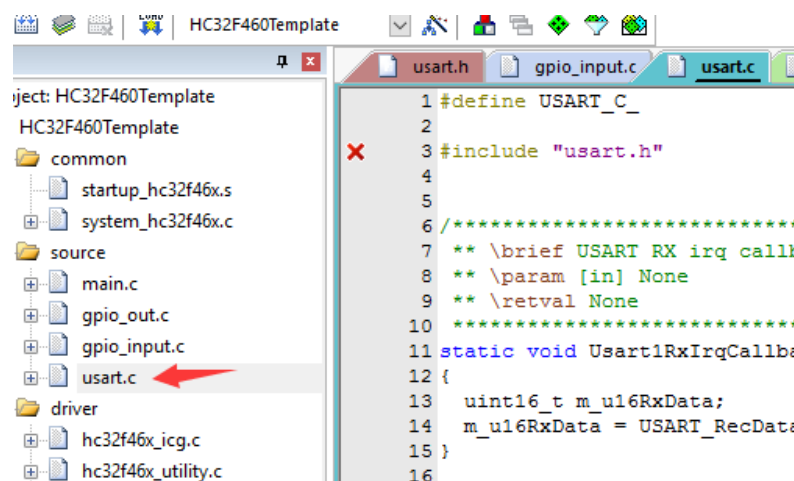
	Func48	Func49	Func50	Func51	Func52	Func53	Func54	Func55	Func56	Func57	Func58	Func59
Func_Grp	I2C1_SDA	I2C1_SCL	I2C2_SDA	I2C2_SCL	I2S1_SD	I2S1_SDIN	I2S1_WS	I2S1_CK	I2S2_SD	I2S2_SDI	I2S2_WS	I2S2_CK
1										N		

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_ul6RxData;
    m_ul6RxData = 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()
    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_FcglPeriphClockCmd(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_Uart1_Rx, Disable);
PORT_SetFunc(USART1_TX_PORT, USART1_TX_PIN, Func_Uart1_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; //设置中断向量
stcIrqRegiCfg.pfnCallback = &Uart1RxIrqCallback; //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); //设置中断
NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn); NVIC_EnableIRQ(stcIrqRegiCfg.enIR

```



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);

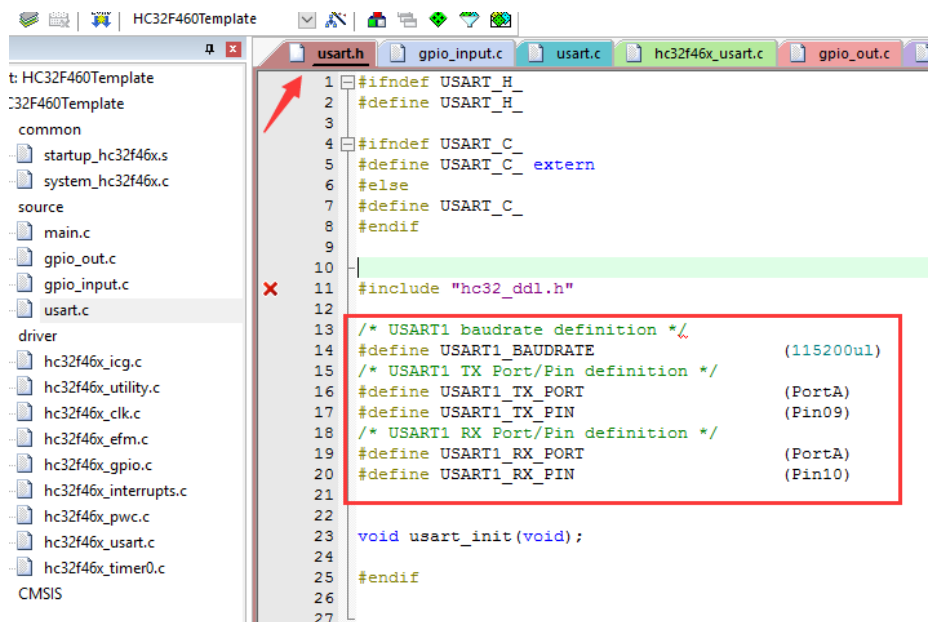
#endif

```



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



```
1 #ifndef USART_H_
2 #define USART_H_
3
4 #ifndef USART_C_
5 #define USART_C_ extern
6 #else
7 #define USART_C_
8 #endif
9
10
11 #include "hc32_ddl.h"
12
13 /* USART1 baudrate definition */
14 #define USART1_BAUDRATE (115200ul)
15 /* USART1 TX Port/Pin definition */
16 #define USART1_TX_PORT (PortA)
17 #define USART1_TX_PIN (Pin09)
18 /* USART1 RX Port/Pin definition */
19 #define USART1_RX_PORT (PortA)
20 #define USART1_RX_PIN (Pin10)
21
22 void usart_init(void);
23
24
25 #endif
26
27
```

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     stc_irq_regi_conf_t stcIrqRegiCfg;
43
44     /* Enable peripheral clock */ /*打开时钟*/
45     PWC_FcglPeriphClockCmd(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);
76
77 }
```

4,配置串口中断接收回调函数

注意:中断向量有144个: Int000_IRQn, Int001_IRQn, Int002_IRQn Int142_IRQn, Int143_IRQn;数字越小,越优先执行

所谓中断向量实际上就是中断地址,Int000_IRQn, Int001_IRQn Int143_IRQn 都是代表不同的中断地址.

程序执行中断的时候是从Int000_IRQn, Int001_IRQn Int143_IRQn 依次执行.

设置中断向量其实就是把这个回调函数放到相应的中断地址上执行,所以每个中断必须设置不同的中断向量.

下面是把串口接收中断函数放到了 Int000_IRQn, 就是说运行 Int000_IRQn地址就是执行串口接收中断函数

-

下面还有个中断优先级,中断优先级共15个

NVIC_SetPriority(stcIrqRegiCfg.enIRQn,
DDL_IRQ_PRIORITY_DEFAULT);//设置中断优先级

中断优先级控制着中断嵌套优先的顺序,

假设串口1接收中断向量是Int000_IRQn, 中断优先级是 2

假设串口2接收中断向量是Int001_IRQn, 中断优先级也是 2

假设同时来中断,因为他们的中断优先级是一样的,那么就会对比中断向量, 所以就会先执行串口1,再执行串口2;

--

假设串口1接收中断向量是Int000_IRQn, 中断优先级是 2

假设串口2接收中断向量是Int001_IRQn, 中断优先级是 1

假设同时来中断,串口2的中断优先级大于串口1, 所以就会先执行串口2,再执行串口1;

假设串口1在执行中断,串口2来了中断,由于串口2的中断优先级大于串口1,所以也会去执行串口2,再执行串口1;

```
70
71 /* Set USART RX IRQ *//*设置串口接收中断*/
72 stcIrqRegiCfg.enIRQn = INT_USART1_RI_IRQ; //设置中断向量
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); //设置中断优先级
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 = INT_USART1_EI_IRQ; //设置中断向量
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); //设置中断优先级
86 NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
87 NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
88
89 /*Enable RX && RX interrupt 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 }
```

```
9
10
11 #include "hc32_ddl.h"
12
13 /* USART1 baudrate definition */
14 #define USART1_BAUDRATE (115200ul)
15 /* USART1 TX Port/Pin definition */
16 #define USART1_TX_PORT (PortA)
17 #define USART1_TX_PIN (Pin09)
18 /* USART1 RX Port/Pin definition */
19 #define USART1_RX_PORT (PortA)
20 #define USART1_RX_PIN (Pin10)
21 /*串口1数据中断向量*/
22 #define INT_USART1_RI_IRQ (Int000_IRQn)
23 /*串口1数据接收错误中断向量*/
24 #define INT_USART1_EI_IRQ (Int001_IRQn)
25
26
27 void usart_init(void);
28
29 #endif
30
```

```
usart.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.

```
usart.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
```

```
usart.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

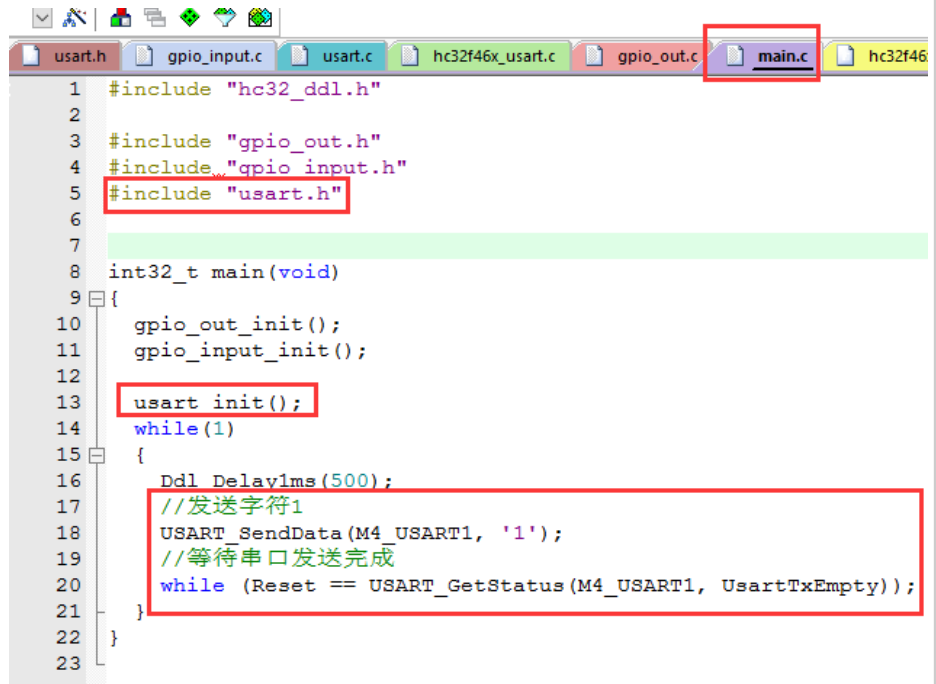
注意!要一句一句写!不要用 USART_FuncCmd(M4_USART1, UsartRx | XXXX | XXXX, Enable); 不可以这样子用

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

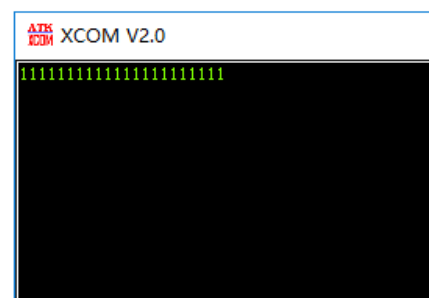
```
usart.h  gpio_input.c  usart.c  hc32f46x_usart.c  gpio_out.c  main.c  hc32f46x_inte
82  stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
83  stcIrqRegiCfg.enIntSrc = INT_USART1_EI; //中断名称(串口接收错误中断)
84  enIrqRegistration(&stcIrqRegiCfg);
85  NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
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



```
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_u16RxData;
14     m_u16RxData = 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 - 669

0

0

« 上一篇: [物联网开发:物联网卡,NB-IOT卡经销商](#)

» 下一篇: [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](#)

posted on 2021-06-01 09:09 杨奉武 阅读(931) 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】并行超算云面向博客园粉丝推出“免费算力限时申领”特别活动

【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】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



编辑推荐：

- 以终为始：如何让你的开发符合预期
- 五个维度打造研发管理体系
- 不会SQL也能做数据分析？浅谈语义解析领域的机会与挑战
- Spring IoC Container 原理解析
- 前端实现的浏览器端扫码功能

最新新闻：

- 董明珠：今年格力没有遇到停电问题 因为三分之一的电来自光伏系统（2021-10-20 22:28）
 - 联发科技天玑5G开放架构 打造差异化手机终端（2021-10-20 22:07）
 - 恒大汽车进行一系列人事任免 引战投尚未有实质性进展（2021-10-20 21:55）
 - 如何实现敏捷赋能？（2021-10-20 21:40）
 - 9月全球热门移动游戏下载量TOP10，《宝可梦大集结》强势登顶（2021-10-20 21:25）
- » 更多新闻...

Powered by:

博客园

Copyright © 2021 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。