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/

导航

博客园
首页
新随笔
联系
订阅 XML
管理

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)

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

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
-
-
-
-
-
-
-
-

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

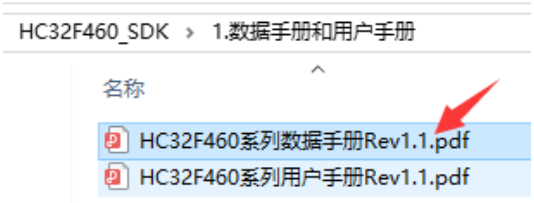
基础外设例程是提供给已经开发过M0或M1或M3或M4等ARM内核单片机的开发人员!

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



推荐排行榜

2.第33页(我以自己使用的HC32F460JETA为例)

主要说一下后面的 Func_Grp

LQFP100	VFBGA100	LQFP64	QFN60	LQFP/QFN48	Pin Name	Analog	EIRQ/WKUP	TRACE/JTAG/SWD	Func0	Func1	Func2	Func3	Func4	Func5	Func6	Func7	Func8	Func9	Func10	Func11	Func12	Func13	Func14	Func15	Func16~31	Func32~63
									GPO	ather	TIM4	TIM6	TIMA	TIMA	EMB,TIMA	USART/SPI/I2SPI	KEY	SDIO	USBFS/I2S	-	-	-	EVNTPT	EVENTOUT	-	Communication Funcs
1	B2	-	-	-	PE2		EIRQ2	TRACECLK	GPO				TIMA_3_PWM_5			USART3_CK								EVENTOUT		Func_Grp2
2	A1	-	-	-	PE3		EIRQ3	TRACEDATA0	GPO				TIMA_3_PWM_6			USART4_CK								EVENTOUT		Func_Grp2
3	B1	-	-	-	PE4		EIRQ4	TRACEDATA1	GPO				TIMA_3_PWM_7											EVENTOUT		Func_Grp2
4	C2	-	-	-	PE5		EIRQ5	TRACEDATA2	GPO				TIMA_3_PWM_8											EVENTOUT		Func_Grp2
5	D2	-	-	-	PE6		EIRQ6	TRACEDATA3	GPO															EVENTOUT		Func_Grp2
6	E2	1	1	1	PH2		EIRQ2		GPO	FCMREF	TIM4_2_CLK		TIMA_4_PWM_7		EMB_IN4			SDIO2_D4	I2S1_EXCK					EVENTOUT		Func_Grp2
7	C1	2	2	2	PC13		EIRQ13		GPO	RTC_OUT			TIMA_4_PWM_8					SDIO2_CK	I2S3_MCK				EVNTP313			Func_Grp2
8	D1	3	3	3	PC14	XTAL32_OUT	EIRQ14		GPO				TIMA_4_PWM_5										EVNTP314			
9	E1	4	4	4	PC15	XTAL32_IN	EIRQ15		GPO				TIMA_4_PWM_6										EVNTP315			
10	F2	-	-	-	VSS																					
11	G2	-	-	-	VCC																					
12	F1	5	5	5	PH0	XTAL_IN	EIRQ0		GPO					TIMA_5_PWM_3												
13	G1	6	6	6	PH1	XTAL_OUT	EIRQ1		GPO					TIMA_5_PWM_4												
14	H2	7	7	7	NRST																					
15	H1	8	8	-	PC0	ADC12_IN10/CMP3_INP3	EIRQ0		GPO				TIMA_2_PWM_5					SDIO2_D5					EVNTP300	EVENTOUT		Func_Grp1
16	J2	9	9	-	PC1	ADC12_IN11	EIRQ1		GPO				TIMA_2_PWM_6					SDIO2_D6					EVNTP301	EVENTOUT		Func_Grp1
17	J1	10	10	-	PC2	ADC1_IN12	EIRQ2		GPO				TIMA_2_PWM_7		EMB_IN3			SDIO2_D7					EVNTP302	EVENTOUT		Func_Grp1
18	K2	11	-	-	PC3	ADC1_IN13/CMP1_IN2	EIRQ3		GPO				TIMA_2_PWM_8					SDIO1_WP					EVNTP303	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	ADC1_IN0/CMP1_INP1	EIRQ0/WKUP0_0		GPO		TIM4_2_OUH		TIMA_2_PWM/TIMA_2_CLKA		TIMA_2_TRIG	SPI1_SSI		SDIO2_D4					EVNTP100	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_Grp 1	USART1_TX	USART1_RX	USART1_RTS	USART1_CTS	USART2_TX	USART2_RX	USART2_RTS	USART2_CTS	SPI1_MOSI	SPI1_MISO	SPI1_SS0	SPI1_SCK	SPI2_MOSI	SPI2_MISO	SPI2_SS0	SPI2_SCK
Func_Grp 2	USART3_TX	USART3_RX	USART3_RTS	USART3_CTS	USART4_TX	USART4_RX	USART4_RTS	USART4_CTS	SPI3_MOSI	SPI3_MISO	SPI3_SS0	SPI3_SCK	SPI4_MOSI	SPI4_MISO	SPI4_SS0	SPI4_SCK

	Func48	Func49	Func50	Func51	Func52	Func53	Func54	Func55	Func56	Func57	Func58	Func59	Func60	Func61	Func62	Func63
Func_Grp 1	I2C1_SDA	I2C1_SCL	I2C2_SDA	I2C2_SCL	I2S1_SD	I2S1_SDIN	I2S1_WS	I2S1_CK	I2S2_SD	I2S2_SDIN	I2S2_WS	I2S2_CK				
Func_Grp 2	I2C3_SDA	I2C3_SCL	CAN_TxD	CAN_RxD	I2S3_SD	I2S3_SDIN	I2S3_WS	I2S3_CK	I2S4_SD	I2S4_SDIN	I2S4_WS	I2S4_CK				

表 2-2 Func32~63 表

4.列如:PC0

PC0后面写的是 Func_Grp1

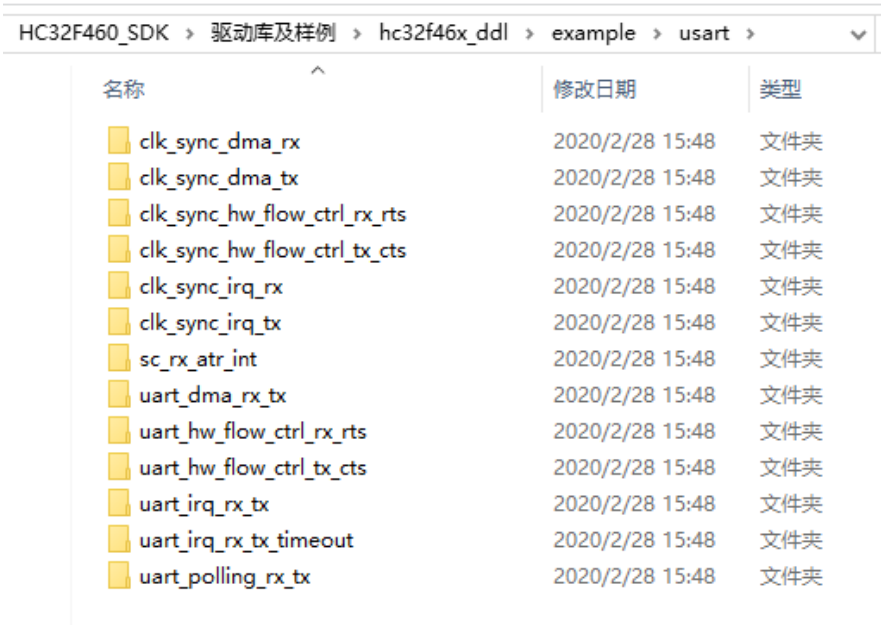
那么这个引脚可以作为这些功能使用

	Func32	Func33	Func34	Func35	Func36	Func37	Func38	Func39	Func40	Func41	Func42	Func43	Func44	Func45	Func46	Func47
Func_Grp 1	USART1_TX	USART1_RX	USART1_RTS	USART1_CTS	USART2_TX	USART2_RX	USART2_RTS	USART2_CTS	SPI1_MOSI	SPI1_MISO	SPI1_SS0	SPI1_SCK	SPI2_MOSI	SPI2_MISO	SPI2_SS0	SPI2_SCK

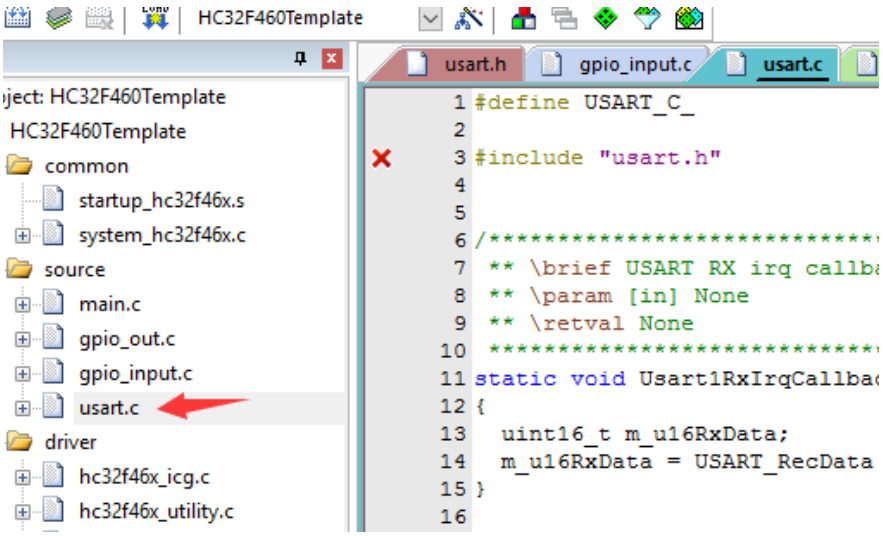
	Func48	Func49	Func50	Func51	Func52	Func53	Func54	Func55	Func56	Func57	Func58	Func59
Func_Grp 1	I2C1_SDA	I2C1_SCL	I2C2_SDA	I2C2_SCL	I2S1_SD	I2S1_SDIN	I2S1_WS	I2S1_CK	I2S2_SD	I2S2_SDIN	I2S2_WS	I2S2_CK

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



基本使用



1.串口基本的初始化

usart.c

```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(
    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 interupt 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_ddl.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 */
```

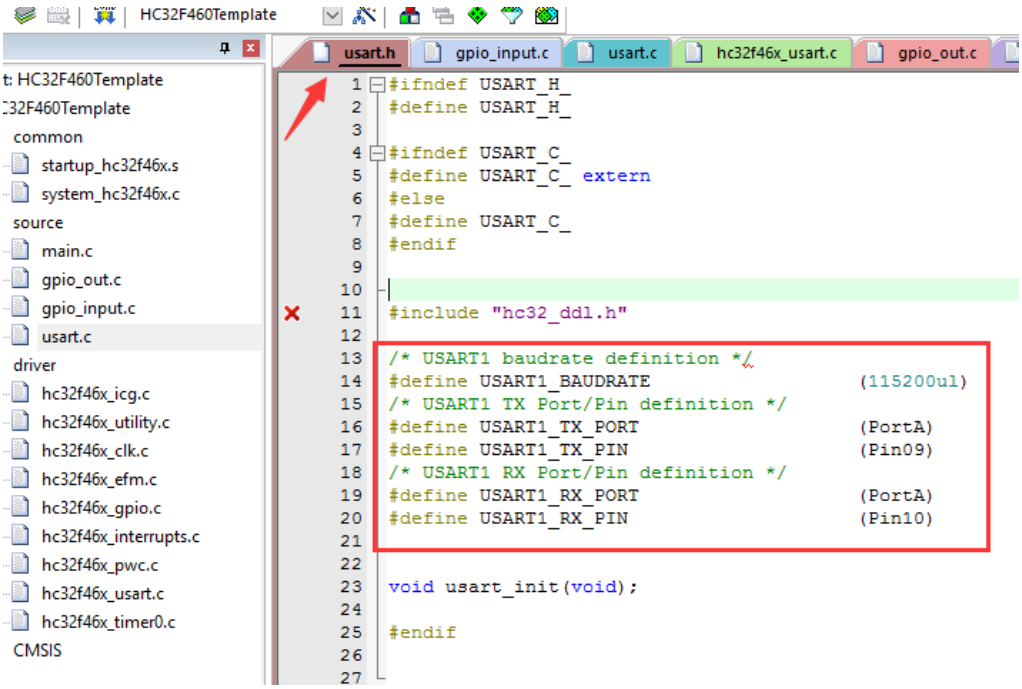
```
#define USART1_RX_PORT                  (PortA)
#define USART1_RX_PIN                   (Pin10)


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

usart.h | gpio_input.c | usart.c | hc32f46x_usart.c | gpio_out.c | main.c | hc32f46x_interrupts.h | hc32f46

```c
/*******************************************************************************
** \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;

  /* 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);

  /*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号
    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位,2字节)
    UsartStartBitFallEdge,
    UsartRtsEnable,        //使能RTS (串口开始传输前让RTS输出一个高脉冲信号)
  };
  /* 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);
```

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

usart.h | gpio_input.c | usart.c | hc32f46x_usart.c | gpio_out.c | main.c | hc32f46x_interrupts.h

```c
  /* 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(串口1接收中断)
  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;//中断优先级
  stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
```

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
27   if (Set == USART_GetStatus(M4_USART1, UsartParityErr)) {USART_ClearStatus(M4_USART1, UsartParityErr);}
28   else{}
29
30   if (Set == USART_GetStatus(M4_USART1, UsartOverrunErr)) {USART_ClearStatus(M4_USART1, UsartOverrunErr);}
31   else{}
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

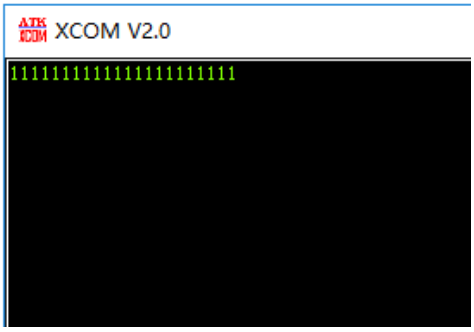
```
82  stcIrqRegiCfg.pfnCallback = &Usart1ErrIrqCallback;
83  stcIrqRegiCfg.enIntSrc = INT_USART1_EI;//中断名称(串口接收错误中断
84  enIrqRegistration(&stcIrqRegiCfg);
85  NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
86  NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
87  NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
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

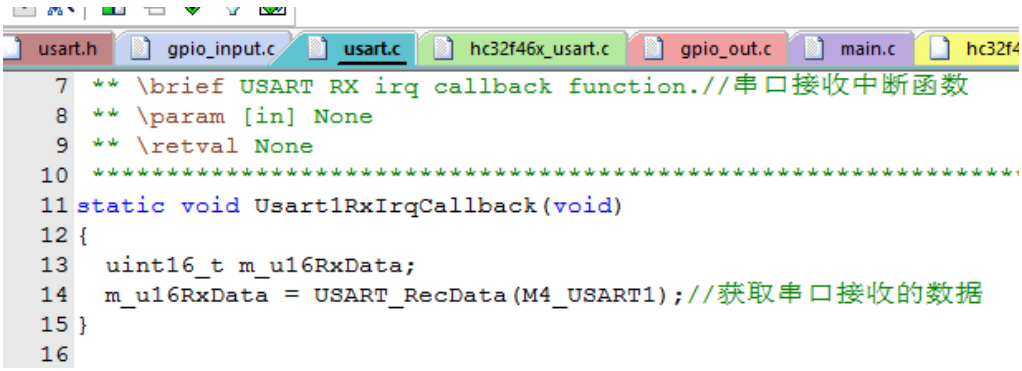
usart.h | gpio_input.c | usart.c | hc32f46x_usart.c | gpio_out.c | main.c | hc32f46

```
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



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

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

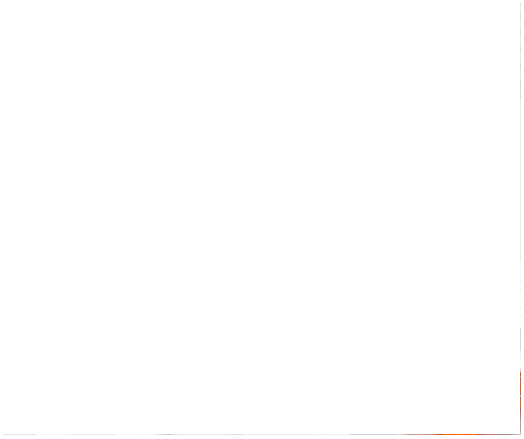
posted on 2021-06-01 09:09 杨奉武 阅读(3) 评论(0) 编辑 收藏

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]



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...

单片机,物联网,上位机,…

扫一扫二维码，加入群聊。