Shanghai Fudan Microelectronics Group Company Limited

复旦微电子

FM3316/FM3313/FM3312
低功耗系列 MCU
应用笔记

UART 低功耗串口

AN0001 **V1.0**

目录

表格图片目录

1 说明

本文档为 FM3316/FM3313/FM3312 系列低功耗 MCU 的应用笔记 ,用于说明 UART 在低功耗情况下使用的原理和方法。FM3316/FM3313/FM3312 系列是复旦微电子公司开发的低功耗 MCU 芯片，请联系复旦微电子公司提供更多相关文档支持设计开发。

2 原理

2.1 背景

很多客户的模块是工作在休眠中的。如何在休眠唤醒间歇实现 UART 通信，本文档将简单讲一下原理与实现方法。

2.1.1 功能描述

- 系列芯片中 FM3316具有4路串口、FM3313具有3路串口、FM3312具有2路串口。其中UART2（PD2引脚，PD3引脚）为低功耗串口，并且在三个型号中是都有的，该串口的RX具有休眠状态下接收数据唤醒功能，即通信起始符的下降沿唤醒芯片，需要注意的是该功能必须使用芯片的PD2引脚作为RX（PD2引脚自带NWKUP异步唤醒功能）。

- 该系列芯片休眠打开低功耗串口后功耗增加很低，休眠时UART模块本身几乎没有什么功耗。但RX脚外部需要有上拉。RX引脚外部如果浮空，从硬件上看，RX脚的电平，将是浮动的。在某些情况下，会引发接收中断，影响功耗。

- 如果用UART0、UART1、UART3，也想实现相同功能。那这3路串口的RX引脚需和一个NWKUP引脚并联，利用NWKUP引脚的下降沿唤醒芯片，再通过UART接收。

2.2 原理

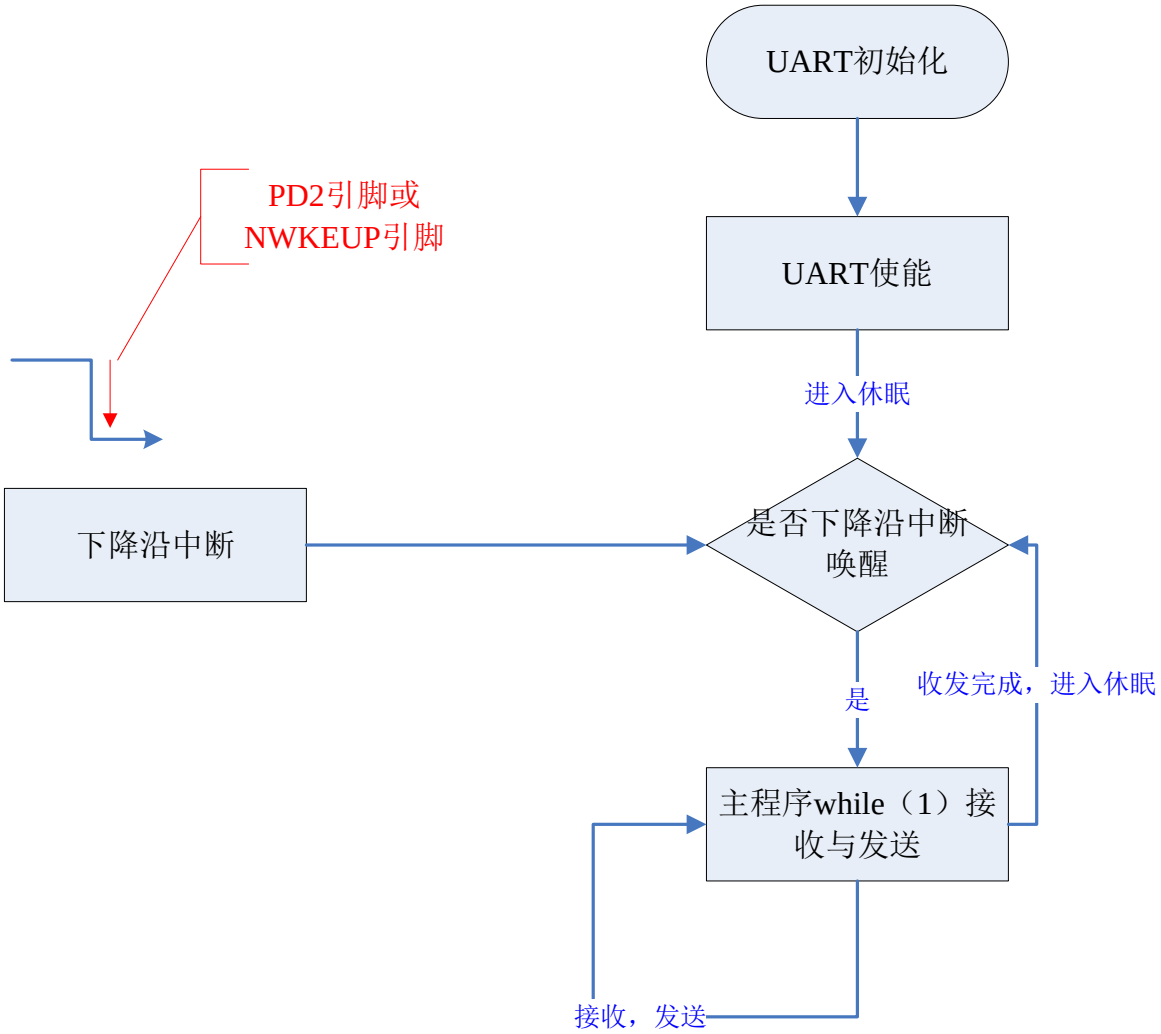

图 1：休眠状态 UART 通信原理框图

3 实现方法

- FM3316 没有在休眠中，接收一个字节然后唤醒 CPU 的功能。 所以只能在接收信号的下降沿来时，唤醒 CPU，使 CPU 工作在 MCLK 模式，去接收与发送字节。工作在 MCLK 模式的，功耗相对高些，大约需要 1.5mA。但因为低功耗应用，肯定不会频繁通信，所以将通信的这个时间，平均到总的工作时间，几乎可以忽略不计。所以一瞬间的高功耗，也就可以忽略不计了。休眠时 UART 模块可以不用关闭使能，UART 模块本身几乎没有什么功耗，但 RX 脚外部需要有上拉。防止输入信号浮空。
- 为何用 PD2 或 NWKUP 引脚作为唤醒源，而不用普通的 IO 口中断作为唤醒源。这是因为 NWKUP 下降沿中断自带 100ns 的滤波，可以有效地滤除毛刺。且 NWKUP 中断这是个异步唤醒的中断，唤醒时间比较短，可以更及时地捕捉到信号。这种休眠唤醒接收波特率，最高大约可达到 38400bps。

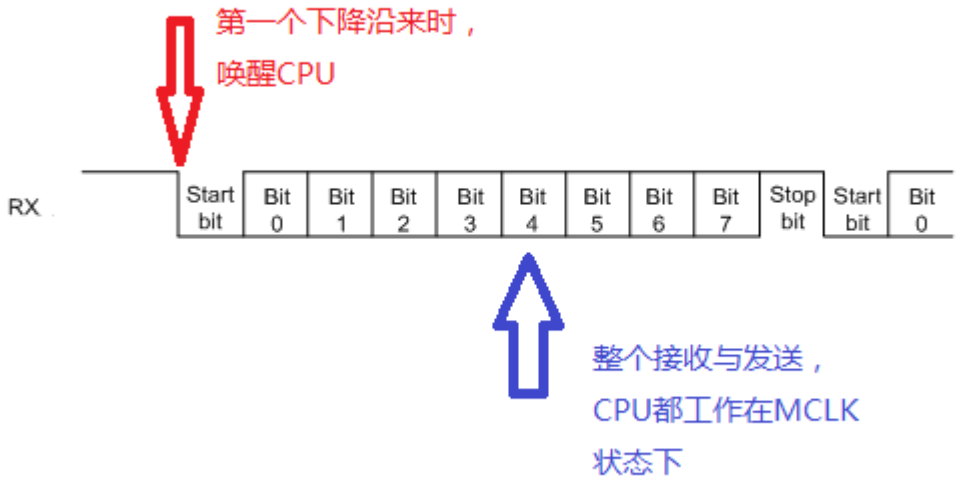

图 2：边沿触发原理图

3.1 芯片结构介绍

3.1.1 功能描述

判定接收发送一个字节的完成标志只能是 UARTIF 寄存器中的 RXIFx 与 TXIFx。其他都不行。

3.1.1.1 UART 中断标志寄存器

表格 1：UART 中断标志寄存器

名称	UART 中断标志寄存器 UARTIF							
地址	01:02A1H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	rxif3	txif3	rxif2	txif2	rxif1	txif1	rxif0	txif0
位权限	R/W0/Dy-0	R/W0/Dy-1	R/W0/Dy-0	R/W0/Dy-1	R/W0/Dy-0	R/W0/Dy-1	R/W0/Dy-0	R/W0/Dy-1

Bit	助记符	功能描述
7	RXIF3	UART3 接收中断标志位 1 = 接收完成标志，数据被写入 RCREG 后硬件自动置 1； 0 = 等待接收或正在接收（CPU 执行任何对 RCREG 的操作将会引起硬件清 0）。
6	TXIF3	UART3 发送中断标志位 1 = 发送缓冲器 TXREG 中数据被送入移位寄存器 TSR 中，硬件自动置 1； 0 = CPU 向发送缓冲器 TXREG 写入数据后，硬件自动清 0。也可软件清 0
5	RXIF2	UART2 接收中断标志位 1 = 接收完成标志，数据被写入 RCREG 后硬件自动置 1； 0 = 等待接收或正在接收（CPU 执行任何对 RCREG 的操作将会引起硬件清 0）。
4	TXIF2	UART2 发送中断标志位

Bit	助记符	功能描述
		1 = 发送缓冲器 TXREG 中数据被送入移位寄存器 TSR 中，硬件自动置 1； 0 = CPU 向发送缓冲器 TXREG 写入数据后，硬件自动清 0。也可软件清 0
3	RXIF1	UART1 接收中断标志位 1 = 接收完成标志，数据被写入 RCREG 后硬件自动置 1； 0 = 等待接收或正在接收（CPU 执行任何对 RCREG 的操作将会引起硬件清 0）。
2	TXIF1	UART1 发送中断标志位 1 = 发送缓冲器 TXREG 中数据被送入移位寄存器 TSR 中，硬件自动置 1； 0 = CPU 向发送缓冲器 TXREG 写入数据后，硬件自动清 0。也可软件清 0
1	RXIF0	UART0 接收中断标志位 1 = 接收完成标志，数据被写入 RCREG 后硬件自动置 1； 0 = 等待接收或正在接收（CPU 执行任何对 RCREG 的操作将会引起硬件清 0）。
0	TXIF0	UART0 发送中断标志位 1 = 发送缓冲器 TXREG 中数据被送入移位寄存器 TSR 中，硬件自动置 1； 0 = CPU 向发送缓冲器 TXREG 写入数据后，硬件自动清 0。也可软件清 0

3.1.1.2 UART 发送状态控制寄存器

表格 2：UART 发送状态控制寄存器

名称	发送状态控制寄存器 TXSTA0							
地址	01:02A8H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	–	STOPSEL0	TXIS0	TXEN0	IREN0	–	TSRF0	TX9D0
位权限		R/W-0	R/W-0	R/W-0	R/W-0		R-1	R/W-0

名称	发送状态控制寄存器 TXSTA1							
地址	01:02A9H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	–	STOPSEL1	TXIS1	TXEN1	IREN1	–	TSRF1	TX9D1
位权限		R/W-0	R/W-0	R/W-0	R/W-0		R-1	R/W-0

名称	发送状态控制寄存器 TXSTA2							
地址	01:02AAH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	–	STOPSEL2	TXIS2	TXEN2	IREN2	–	TSRF2	TX9D2
位权限		R/W-0	R/W-0	R/W-0	R/W-0		R-1	R/W-0

名称	发送状态控制寄存器 TXSTA3							
地址	01:02ABH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	–	STOPSEL3	TXIS3	TXEN3	IREN3	–	TSRF3	TX9D3
位权限		R/W-0	R/W-0	R/W-0	R/W-0		R-1	R/W-0

Bit	助记符	功能描述
7		
6	STOPSELx	停止位选择位 1 = 停止位为 2 位； 0 = 停止位为 1 位。
5	TXISx	发送中断选择位 1 = 移位寄存器空产生中断；（使用这个） 0 = 发送缓冲器空产生中断。（不要使用）
4	TXENx	发送模块使能位 1 = 使能发送模块； 0 = 禁止发送模块，发送模块被复位。
3	IRENx	发送红外调制使能位 1 = 使能发送红外调制； 0 = 禁止发送红外调制。
2		
1	TSRFx	发送移位寄存器 TSR 空标志位 1 = TSR 空；（不要使用） 0 = TSR 满。（不要使用）
0	TX9Dx	发送数据的第 9 位，此 bit 应在启动发送前写入

3.1.1.3 发送缓冲区状态控制寄存器

表格 3：发送缓冲区状态控制寄存器

名称	发送缓存状态控制寄存器 TXFIFOSTA0							
地址	01:02BCH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			TX_INTSEL0		--	TXFF0
位权限	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0

名称	发送缓存状态控制寄存器 TXFIFOSTA1							
地址	01:02BDH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			TX_INTSEL1		--	TXFF1
位权限	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0

名称	发送缓存状态控制寄存器 TXFIFOSTA2							
地址	01:02BEH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			TX_INTSEL2		--	TXFF2
位权限	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0

名称	发送缓存状态控制寄存器 TXFIFOSTA3							
地址	01:02BFH							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			TX_INTSEL3		--	TXFF3
位权限	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0

Bit	助记符	功能描述
7		
6	RFUI	保留位
5		
4		
3	TX_INTSEL	发送中断选择位 11 = 不产生中断 10 = 发送缓冲器空产生中断 01 = 发送缓冲器空且移位寄存器空产生中断 00 = 根据 TXIS 寄存器的设定产生中断（使用这个）
2		
1		
0	TXFF	发送缓冲器状态位 1 = 发送缓冲器中有 1 笔数据； 0 = 发送缓冲器空；

接收缓冲区状态控制寄存器

表格 4：接收缓冲区状态控制寄存器

名称	接收缓存状态控制寄存器 RXFIFOSTA0							
地址	01:02C0H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			---	RX_INTSEL0	--	RXFF0
位权限	U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0

名称	接收缓存状态控制寄存器 RXFIFOSTA1							
地址	01:02C1H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			---	RX_INTSEL1	--	RXFF1
位权限	U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0

名称	接收缓存状态控制寄存器 RXFIFOSTA2							
地址	01:02C2H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

名称	接收缓存状态控制寄存器 RXFIFOSTA2							
地址	01:02C2H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			---	RX_INTSEL2	--	RXFF2
位权限	U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0

名称	接收缓存状态控制寄存器 RXFIFOSTA3							
地址	01:02C3H							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	---	RFUI			---	RX_INTSEL3	--	RXFF3
位权限	U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0

Bit	助记符	功能描述
7		
6	RFUI	保留位
5		
4		
3		
2	RX_INTSEL	接收中断选择位 1 = 缓冲器满产生中断；（不要使用） 0 = 单笔接收完成产生中断（使用这个）
1		
0	RXFF	接收缓冲器状态位 1 = 接收缓冲器中有 1 笔数据； 0 = 接收缓冲器空；

3.2 库函数介绍

3.2.1 UART 库函数

表格 5：UART 库函数

函数名	描述
UART_Init	UART 初始化
UART_RX_Interrupt_Enable	UART 接收终端允许
UART_RX_Enable	UART 接收模块打开
UART_TX_Enable	UART 发送模块打开

3.2.1.1 函数 UART_Init

表格 6：函数 UART_Init

函数名	UART_Init
函数原型	unsigned char UART_Init(unsigned char UARTx,UART_INIT_STRU *init)
功能描述	UART 初始化
输入参数 1	UARTx ：UART0 to UART3
输入参数 2	UART_INIT_STRU ：波特率，数据位，停止位，校验位，电平正反
输出参数 1	无
返回值	0 正确。 1 错误

3.2.1.2 函数 UART_RX_Interrupt_Enable

表格 7：函数 UART_RX_Interrupt_Enable

函数名	UART_RX_Interrupt_Enable
函数原型	void UART_RX_Interrupt_Enable(unsigned char UARTx)
功能描述	UART 接收中断允许
输入参数 1	UARTx ：UART0 to UART3
输出参数 1	无
返回值	无

3.2.1.3 函数 UART_RX_Enable

表格 8：函数 UART_RX_Enable

函数名	UART_RX_Enable
函数原型	void UART_RX_Enable(unsigned char UARTx)
功能描述	UART 接收模块打开
输入参数 1	UARTx ：UART0 to UART3
输出参数 1	无
返回值	无

3.2.1.4 函数 UART_TX_Enable

表格 9：函数 UART_TX_Enable

函数名	UART_TX_Enable
函数原型	void UART_TX_Enable(unsigned char UARTx)
功能描述	UART 接收模块打开
输入参数 1	UARTx ：UART0 to UART3
输出参数 1	无
返回值	无

3.3 参考例程使用说明

例程请参考 UART 休眠唤醒示例程序。

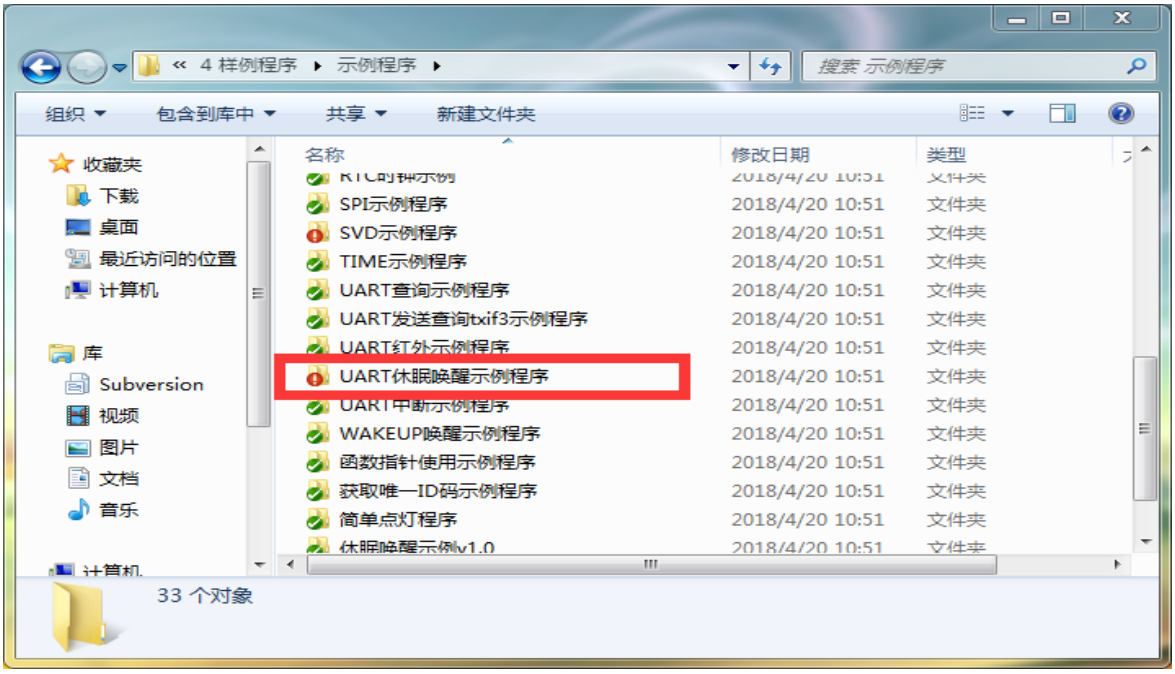

图 3：UART 休眠唤醒示例程序

3.3.1 初始化 UART 引脚

GPIO_Init(PORTD,PIN2,GPIO_MODE_ALTERNATE_FUNCTION, MODE_ALTERNATE_SELECT_0); //管脚 PD2 配置成 uart 2

GPIO_Init(PORTD,PIN3,GPIO_MODE_ALTERNATE_FUNCTION,MODE_ALTERNATE_SELECT_0);//管脚 PD3 配置成 uart 2

3.3.2 初始化 UART

```
//配置 UART2 寄存器
init.UART_BaudRate = 9600; //波特率 最高 38400bps (波特率高低 跟字节间间隔也有关系，自己试吧)
init.UART_DataLength = 8;//数据位长度 7，8 或 9 (7 位与 9 位数据无校验位)
init.UART_StopBits = 1;//停止位 1 或 2
init.UART_Parity = 2;//校验位 0 无校验 ，1 奇校验 ，2 偶校验
init.UART_RxTxDfl = 0;//电平取反(RS485 不同型号使用) 0 不取反 ，1 发送取反 ，2 接收取反 ，3 接收发送都取反
UART_Init(UART2,&init);
```

3.3.3 打开接收中断

```
UART_RX_Interrupt_Enable(UART2);
```

3.3.4 打开接收与发送模块

```
UART_RX_Enable(UART2);
UART_TX_Enable(UART2);
```

3.3.5 中断处理函数

```
//UART0/1/2/3
void Eint2_int (void) interrupt 10 using 1
{
unsigned char Temp_IF, Temp;
  AIF2 = 0;
  Temp_IF = UARTIE&UARTIF; //任何对寄存器的操作都会导致其清零，必须先缓存
  if(Temp_IF)
  {
        //UART0
        if( (Temp_IF&B0000_0010) ) //接收
        {
              Temp = RXREG0; //从 UART0 数据寄存器中取接收数据
              //用户中断代码
              if(RXSTA0&B0000_1110)//出错
              {
                    RXSTA0&=~B0000_1110;
              }
           TXREG0 = Temp;
        }
        if( (Temp_IF&B0000_0001) ) //发送
        {
              //用户中断代码
              UARTIF = B1111_1110;
        }
        //UART1
        if( (Temp_IF&B0000_1000) ) //接收
        {
              Temp = RXREG1; //从 UART1 数据寄存器中取接收数据
              //用户中断代码
              if(RXSTA1&B0000_1110)//出错
              {
```

```
            RXSTA1&=~B0000_1110;
        }
        TXREG1 = Temp;
    }
    if( (Temp_IF&B0000_0100) ) //发送
    {
        //用户中断代码
        UARTIF = B1111_1011;
    }
    //UART2
    if( (Temp_IF&B0010_0000) ) //接收
    {
        Temp = RXREG2; //从 UART2 数据寄存器中取接收数据
        //用户中断代码
        if(RXSTA2&B0000_1110)//出错
        {
            RXSTA2&=~B0000_1110;
        }
        buf[num++]= Temp;
    }
    if( (Temp_IF&B0001_0000) ) //发送
    {
        //用户中断代码
        UARTIF = B1110_1111;
    }
    //UART3
    if( (Temp_IF&B1000_0000) ) //接收
    {
        Temp = RXREG3; //从 UART3 数据寄存器中取接收数据
        //用户中断代码
        if(RXSTA3&B0000_1110)//出错
        {
```

```
                RXSTA3&=~B0000_1110;
            }
            TXREG3 = Temp;
        }
        if( (Temp_IF&B0100_0000) ) //发送
        {
            //用户中断代码
            UARTIF = B1011_1111;
        }
        UARTIF = 0;
    }
}
//NWKUP 中断
void nmi_int (void) interrupt 7 using 2
{
  if(NMIFLAG & B0000_0010)//低频晶振停振中断
  {
        if(FDETIF & B0000_0001)
        {
            FDETIE = 0;
            FDETIF = 0;
        }
  }
  if(NMIFLAG & B0000_0001)
  {
        if(LPRUNERR & B0000_0001)
        {
//          SET_MCLK(XTLF);
            LPRUNERR = 0;
        }
        if(PMU_WKSRC & B0000_1111)
        {
```

```
            WakeSrc = PMU_WKSRC&B0000_0111;
            PMU_WKSRC = 0;//NWKUP 中断
        }
    }
    NMIFLAG = 0;
}
```

3.3.6 主程序处理函数

```
for( ; ; )
{
    EA = 1;//打开全局中断
    Clear_SYS_Wdt();
    Sleep(1,0,0);//深度休眠
    if( (WakeSrc ==B0000_0101)||(WakeSrc ==B0000_0110) )
    {
        WakeSrc=0;
        EA=1;
        //等 1 秒钟，等待接收完成  闪个灯方便 demo 观察
        GPIO_Write_Toggle_Data_Bit(PORTE  ,PIN0);
        Delay_x5ms(100);
        GPIO_Write_Toggle_Data_Bit(PORTE  ,PIN0);
        Delay_x5ms(100);
         //将接收到的字节发送回去
        for(i=0;i<num;i++)
        {
            TXREG2 = buf[i];
            while(!(UARTIF & txif2));
        }
        num=0;
    }
}
```

4 建议的实现步骤

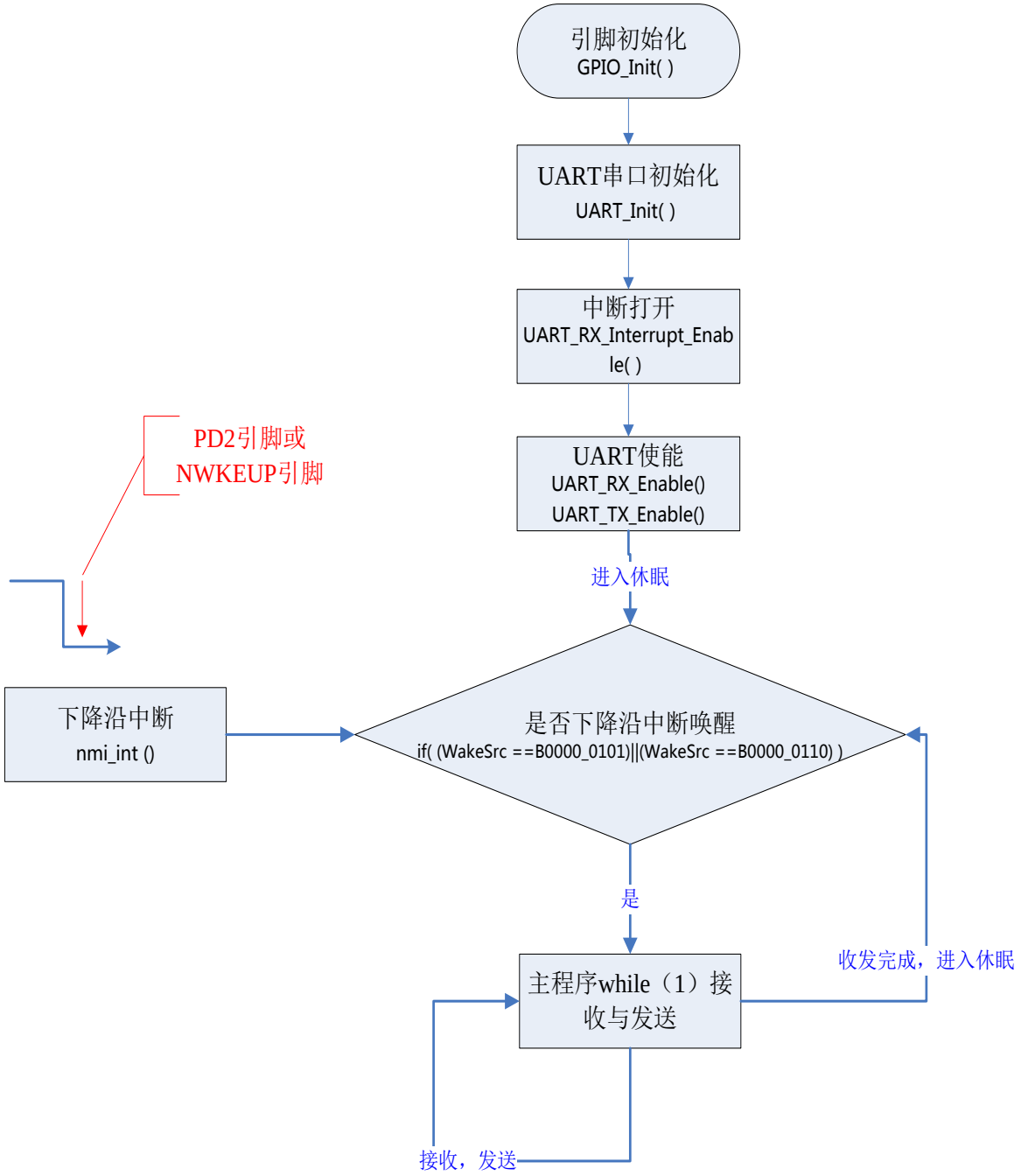

图 4：实现步骤

5 注意事项

5.1 软件设计

- 判定接收发送一个字节的完成标志只能是 UARTIF 寄存器中的 RXIFx 与 TXIFx，其他都不行。
- 寄存器的接收与发送的配置请尽量按照例程，有些寄存器配置不对会导致收发丢字节。
- 如果 RX 外部无上拉，休眠时，需要将 UART 口配置成普通 IO，否则 IO 口电平浮动将可能接收错误字节
- 只有 PD2，PD3 口，是自带低功耗唤醒功能。例程也是按照这个口做的。

版本信息

版本号	发布日期	更改说明
1.0	2018.05	首次发布

上海复旦微电子集团股份有限公司销售及服务网点

上海复旦微电子集团股份有限公司
地址：上海市国泰路 127 号 4 号楼
邮编：200433
电话：(86-021) 6565 5050
传真：(86-021) 6565 9115

上海复旦微电子（香港）股份有限公司
地址：香港九龙尖沙咀东嘉连威老道 98 号东海商业中心 5 楼 506 室
电话：(852) 2116 3288　2116 3338
传真：(852) 2116 0882

北京办事处
地址：北京市东城区东直门北小街青龙胡同 1 号歌华大厦 B 座 423 室
邮编：100007
电话：（86-10）8418 6608
传真：（86-10）8418 6211

深圳办事处
地址：深圳市华强北路 4002 号圣廷苑酒店世纪楼 1301 室
邮编：518028
电话：(86-0755) 8335 0911　8335 1011　8335 2011 8335 0611
传真：(86-0755) 8335 9011

台湾办事处
地址：台北市 114 内湖区内湖路一段 252 号 12 楼 1225 室
电话：(886-2) 7721 1889
传真：(886-2) 7722 3888

新加坡办事处
地址：237, Alexandra Road, #07-01, The Alexcier, Singapore 159929
电话：(65) 6472 3688
传真：(65) 6472 3669

北美办事处
地址：2490 W. Ray Road Suite#2 Chandler, AZ 85224 USA
电话：(480) 857-6500 ext 18

公司网址：http://www.fmsh.com/