# G030F6P6软件流程图

**西安冠通数源电子股份有限公司**

**2021年10月18日**

**文档修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修改人** | **修改页码、原因及主要内容** | **日期** |
| V1.0 | 贾磊 | 创建文档。 | 2021/10/08 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 概述

本文档用于描述G030F6P6软件的书写流程。

# Cubemx配置生成部分

# 程序中设计的全局变量结构体

## SysFlag

typedef union

{

uint16\_t bytes;

struct

{

uint16\_t flashSave\_Flag:1;//flash可写标志位，只有485通讯或者一线通通信满足条件才会设置此bit为1。 485标定电压时的数据，485写ID时数据，一线通写ID数据时。

有些代码中直接调用Flash\_WritePara进行了flash读写：

一线通发下载指令时；深度休眠之前需要写一次flash；电流校准后；flash初始化时。

uint16\_t socInit\_Flag:1; //Soc初始化标志位 第一次初始化soc后将此bit置1

uint16\_t DET\_Flag:1; //获取对应的CHG寄存器中的值赋到这个变量中 0-充电器未连接，1-充电器已连接

uint16\_t LoadFlag:1; //负载是否连接0-未连接，1-已连接，读中颖获取对应的CHG寄存器中的值赋到这个变量中

uint16\_t StopSocAdddFlag:1; //停止soc积分标志

uint16\_t StopSocCutFlag:1; //停止soc削减标志

uint16\_t CHGingFlag:1; //正在充电标志位，本程序未用到

uint16\_t DSGingFlag:1; //正在放电标志位，本程序未用到

uint16\_t RTCWeak:1; //rtc-wakeup，rtc中断中会将此bit置1

uint16\_t CHGWeak:1; //充电wakeup，会在中断中置1

uint16\_t DSGWeak:1; //放电wekeup，会在中断中置1

uint16\_t SIFWeak:1; //一线通唤醒，会在中断中置1

uint16\_t VoltChargeBigFlag:1;//电压change大，1s内变化超过xmV，此程序没用到

uint16\_t SysWeakFlag:1; //在此程序中该变量一直为1，有些程序中有用，此程序没用到

uint16\_t SampOverFlag:1; //采样完成标志，此程序中未用到

uint16\_t Rs485Staty:1; //485状态status，此程序未用到

}bits;

}SysFlag;

## Syspara

typedef struct

{

uint16\_t alarmByte; //报警字节-本程序未使用

uint8\_t SOC; //soc自己程序计算的

uint8\_t SOH; //soh

uint8\_t setSoc; //手动设置的soc值-本程序未使用

uint8\_t batStat; //电池状态-充电、放电、静置

uint8\_t chgMos; //充电mos的状态

uint8\_t dsgMos; //放电mos的状态

uint8\_t cellTemp[6]; //电芯温度，500ms循环中会更新一次

uint8\_t tempMax; //最大值温度，500ms循环中会更新一次

uint8\_t tempMin; //最小值温度，500ms循环中会更新一次

uint8\_t tempAvg; //平均值温度，500ms循环中会更新一次

uint8\_t balTemp; //均衡温度，500ms循环中会更新一次

uint8\_t mosTemp; //mos温度，500ms循环中会更新一次

uint8\_t envTemp; //环境温度，本程序未用到

uint16\_t cellVolt[25]; //电芯电压，标定后的，500ms循环中会更新一次

uint16\_t batVolt[25]; //电芯电压，标定前的，500ms循环中会更新一次

int16\_t voltMax; //电压最大值，500ms循环中会更新一次

int8\_t voltMaxNum; //电压最大的电芯编号，500ms循环中会更新一次

int16\_t voltMin; //电压最小值，500ms循环中会更新一次

int16\_t voltAvg; //电压平均值，500ms循环中会更新一次

int16\_t voltDiff; //压差，500ms循环中会更新一次

uint16\_t Period\_Integral; //soc积分时间，100ms循环周期里面会更新。

uint16\_t IntegralPower; //放电soc系数，sys初始化时赋值；检测电池充放电状态时赋值；soc校准时赋值；mos控制时赋值

uint16\_t IntegralPower\_C; //充电soc系数，sys初始化时赋值；检测电池充放电状态时赋值；soc校准时赋值；mos控制时赋值

uint16\_t IntegralPowerjisuan; //soc系数，可赋值充电系数、放电系数、静置系数

uint32\_t totalSumVolt; //单体累加总压

uint32\_t FullChargeCap; /\*充满容量\*/

uint32\_t RemainCap; /\*积分剩余容量\*/

// uint32\_t AbsCap; /\*绝对容量\*/

uint32\_t TotalDsgCap; //总放电容量

int32\_t current; //总电流，放电为负充电为正

int8\_t CurrCailFalg; //电流校准标志，在一线通或485通信时会被配置。

int8\_t DeepSleepCMD; //没用

int8\_t SleepCMD; //没用

uint8\_t SocChangFlag; //没用到

}SysPara;

## FlashPara

typedef struct

{

uint32\_t appmask;//=0xABCD时会进入下载模式（一线通时可以），默认设置的0x1234

uint32\_t yuliu;

uint8\_t id[16];//可以通过485改，也可以通过一线通改

uint8\_t chgMode;//充电模式，只有初始化的设置了一下-程序后面没用到

uint8\_t workMode;//工作模式，后面没用

uint16\_t CycleCount;//循环次数，放电容量达到充满容量+1

int16\_t currentRate; //标定电流，100ms循环里面若CurrCailFalg=1，重新获取标定电流

uint16\_t FistUseData; //只有初始化时设置了，后面没用到

uint32\_t cyc\_CHG;//充电循环次数

uint32\_t shortCnt;

int8\_t VoltRate[32]; //标定电压值

uint32\_t BKP3\_Data; //本程序未用到

uint16\_t P\_UV\_Count;//过放次数

uint16\_t checkh; //校验值

}FlashPara;

## ringBuffer\_t 存放485数据的结构体

typedef struct{

uint16\_t headPosition;//缓冲区头部位置

uint16\_t tailPosition;// 缓冲区尾部位置

uint8\_t ringBuf[BUFFER\_MAX];//缓冲区数据数组

uint8\_t dummy; //读485buf数据成功置0，失败置1

uint8\_t rcvFlag; //串口接收完数据就会置1，表示又485数据需要处理 -在中断中置1

uint8\_t NodataTime; //有485通信时置0，没485通讯时会累加最大到99

}ringBuffer\_t;

## AlarmSt

typedef union

{

uint16\_t bytes;

struct

{

uint16\_t O\_V:1; //0 单体过压

uint16\_t U\_V\_2700:1; // 1 低压<2700

uint16\_t O\_C\_D:1;// 2 放电过流

uint16\_t S\_C\_D:1;// 3 短路

uint16\_t U\_T\_D:1;// 4 放电低温

uint16\_t O\_T\_M:1;// 5 mos高温

uint16\_t O\_C\_C:1;// 6 充电过流

uint16\_t U\_V\_1500:1;// 7 低压<1500

uint16\_t O\_T\_C:1;// 8 充电高温

uint16\_t O\_T\_D:1;// 9 放电高温

uint16\_t U\_T\_C:1;// 10 充电低温

uint16\_t C\_M\_E:1;// 11 充电mos故障

uint16\_t D\_M\_E:1; 放电mos故障

uint16\_t C\_P\_Count:1; 充电过压保护

uint16\_t RESREVE:2; 预留

}bits;

}AlarmSt;

## ALARM

typedef struct

{

uint8\_t alarmFlag;

uint8\_t Times;

uint8\_t ReTimes;

//回调函数

//void ( \*Callback )( void );

} ALARM;

# 主流程

## 使能中断(bootloader里面关闭了中断，主程序中要重新开总中断)

## 开启总线时钟

## 设置不同外设的时钟源(RTC/UART/ADC).

## bsp\_init

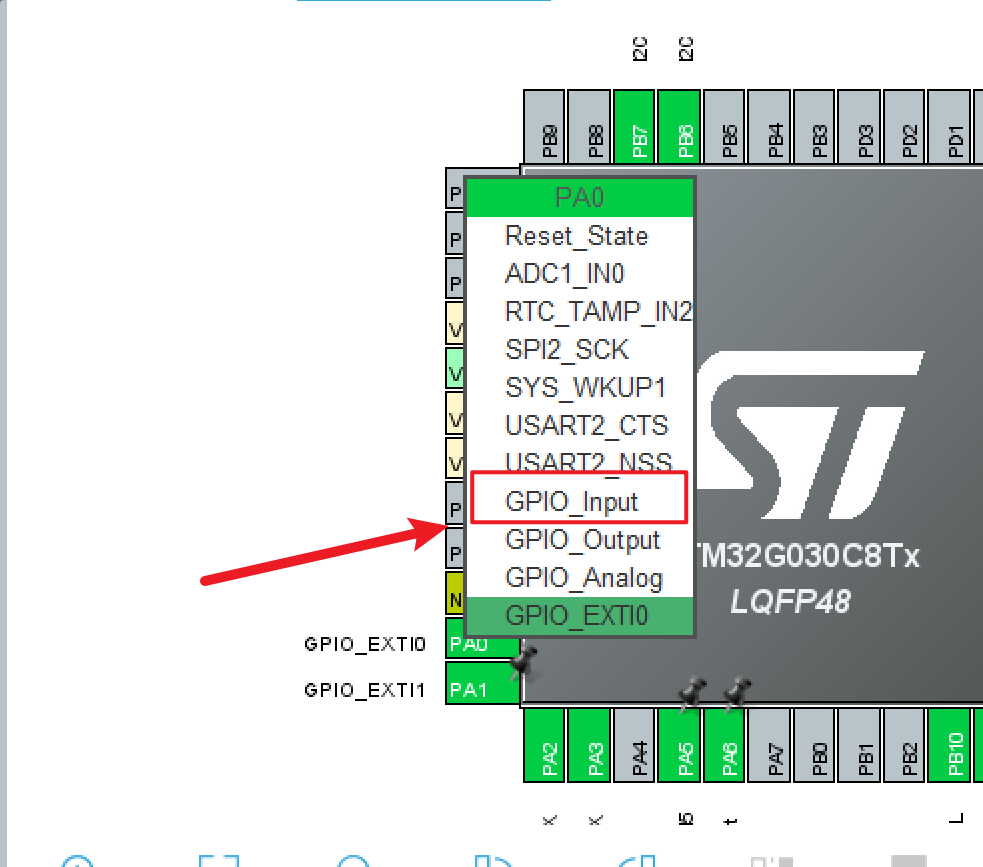
### GPIO初始化

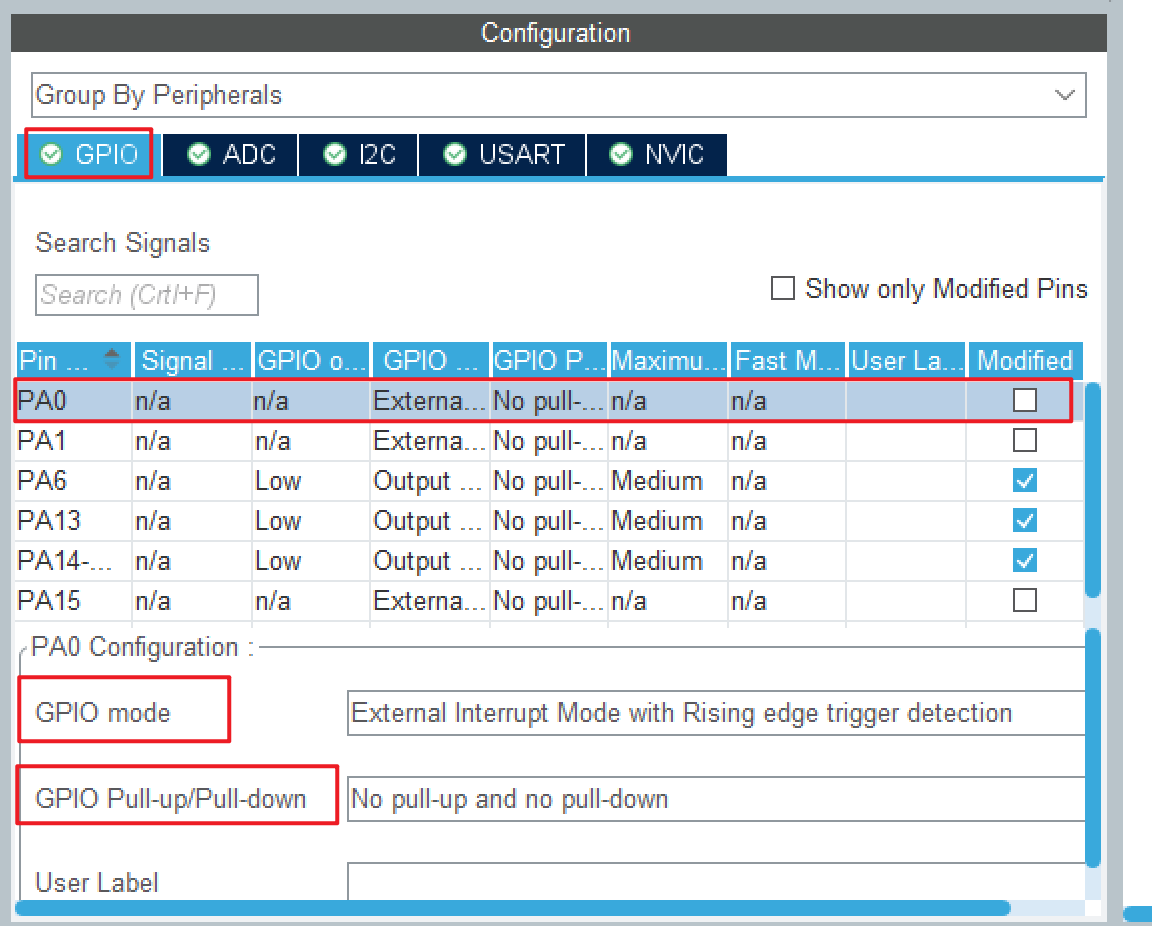
对于需要输出高低电平的引脚输出选择：output+pp +无上下拉+速度

对于需要获取电平信号的引脚设置：input+无上下拉+速度

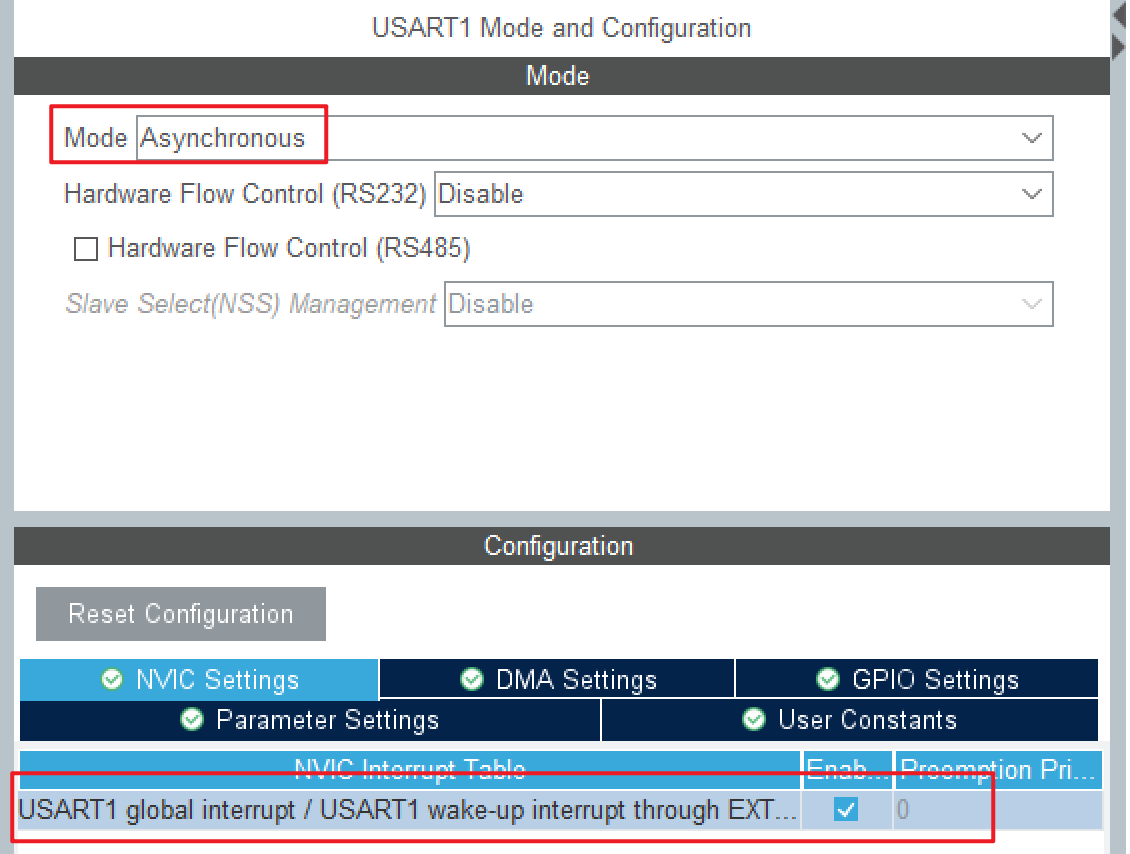
对于一线通的tx引脚配置：output+od+无上下拉+速度

备注：io口可以用作不同的功能，需要在使用不同的功能之前重新初始化io口配置



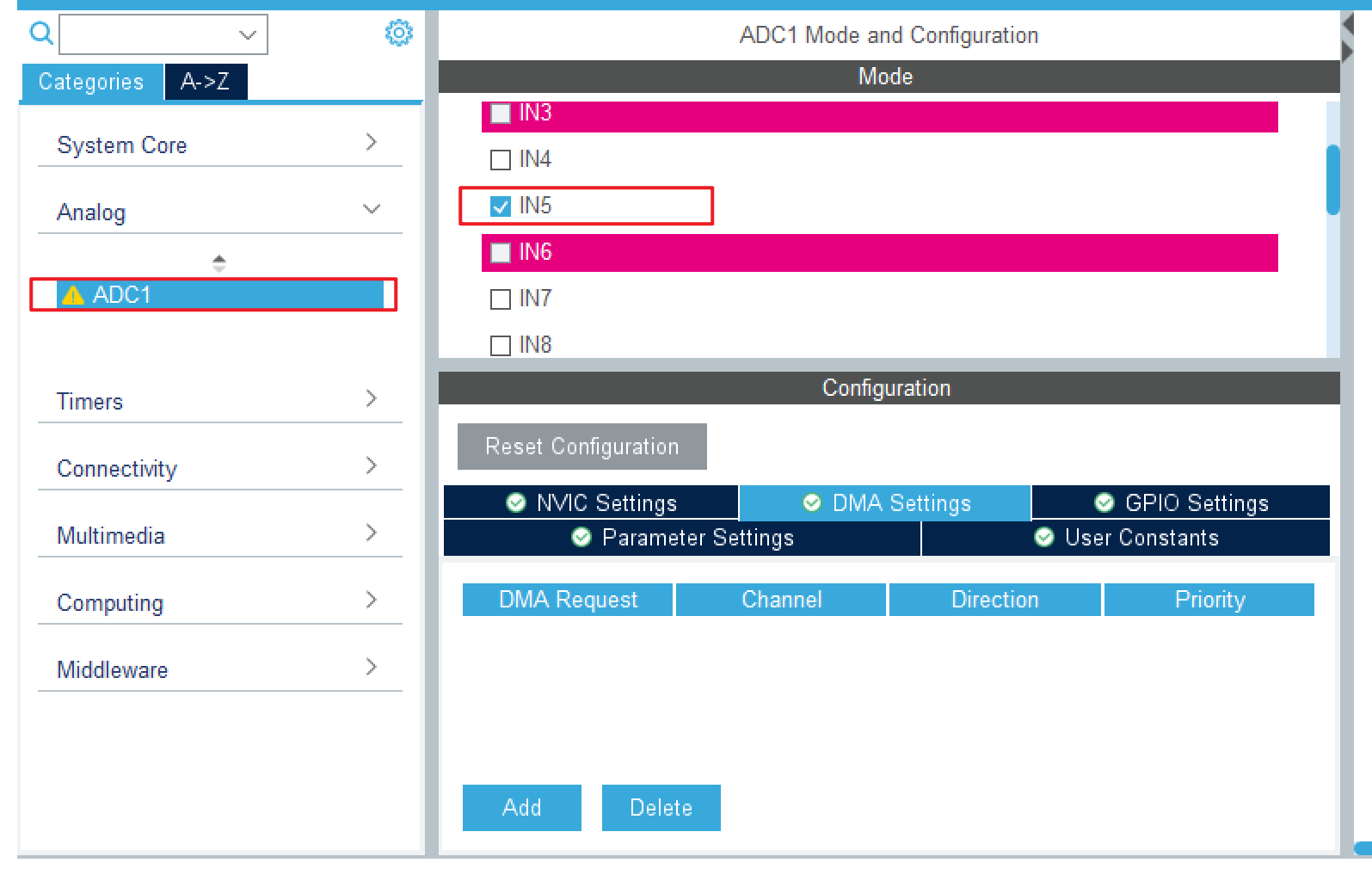


### Uart初始化



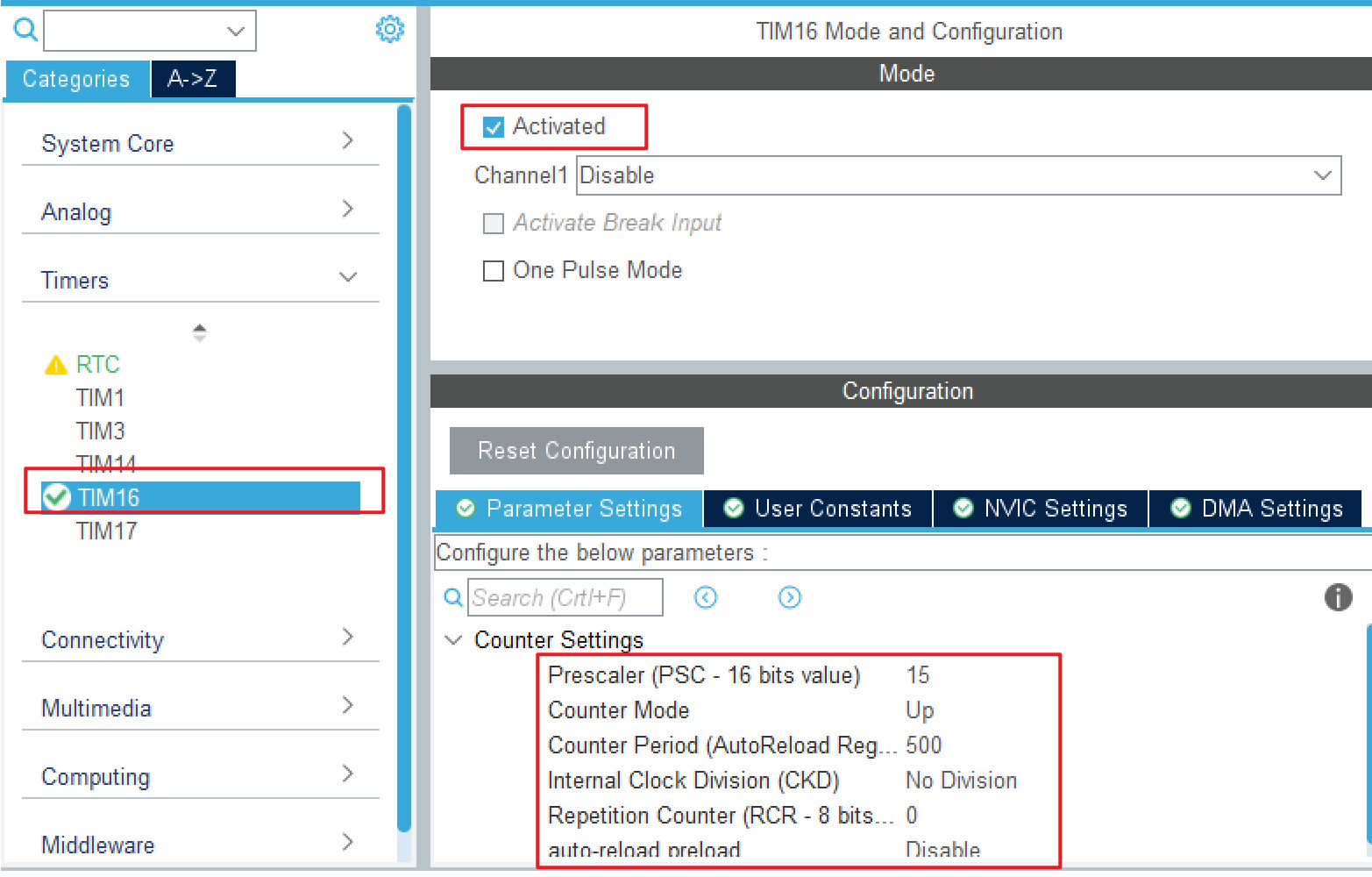
### Adc初始化

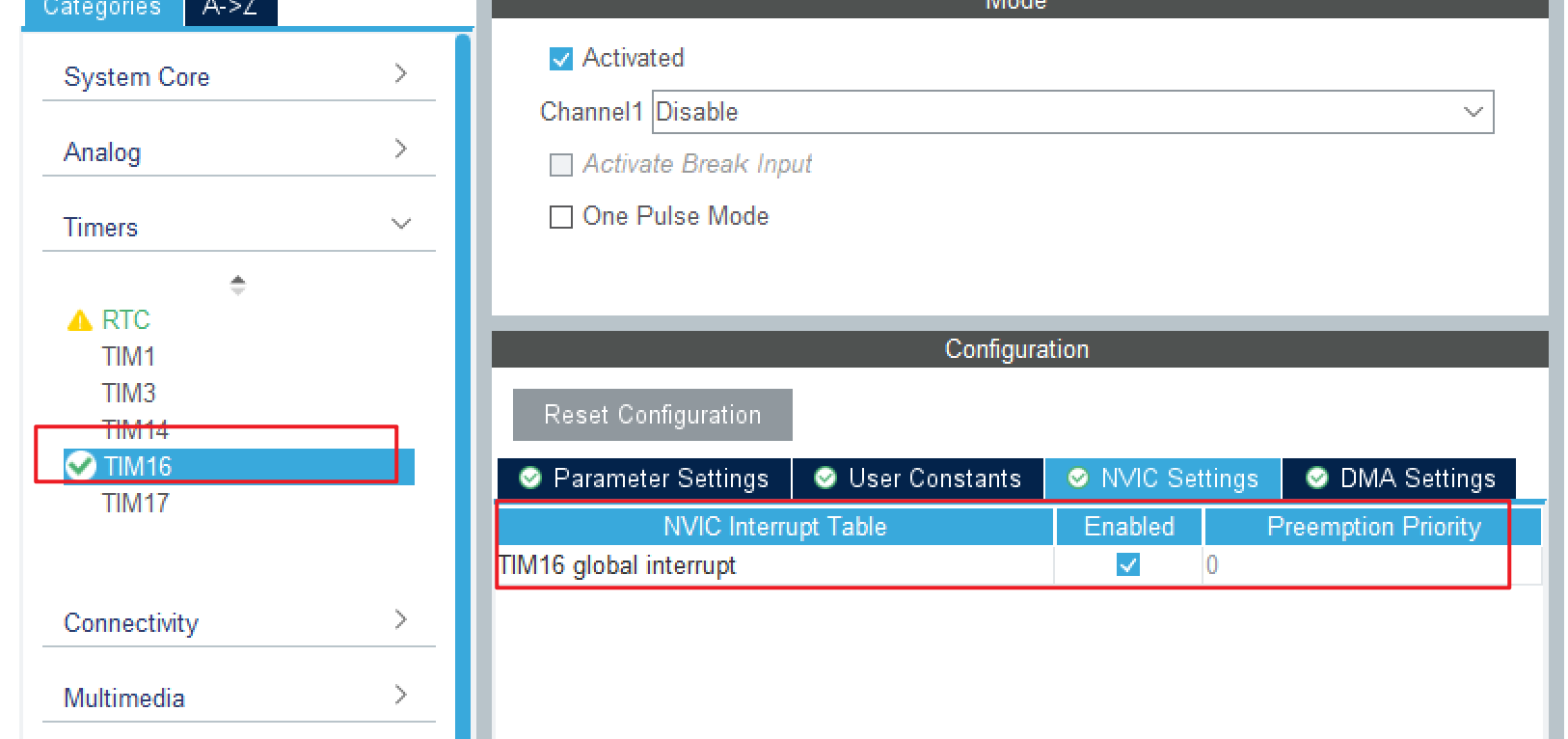
Adc配置按照cubemx默认配置。



### Timer初始化

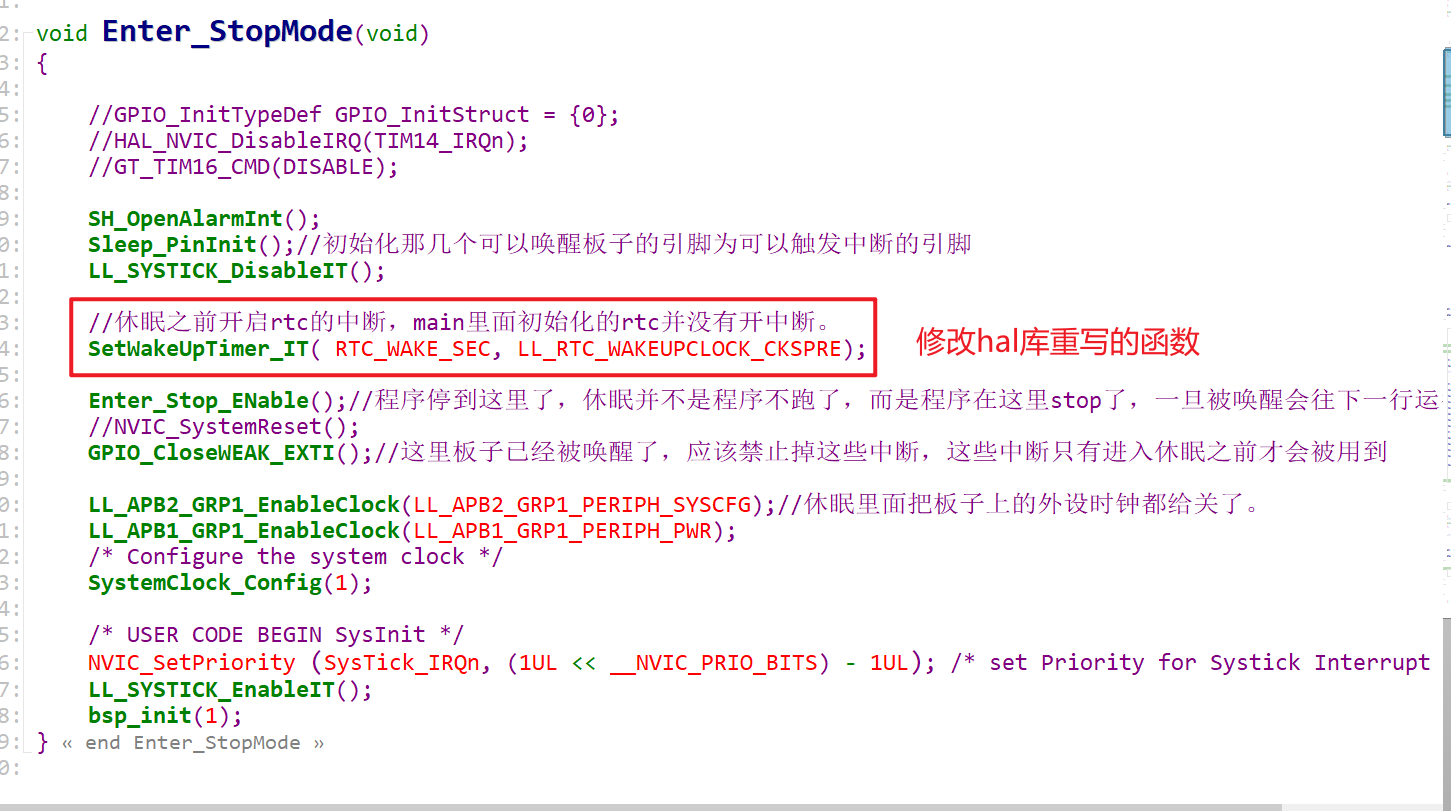
如图配置timer就初始化完成了，至于timer的中断可以在用的时候再去使能。

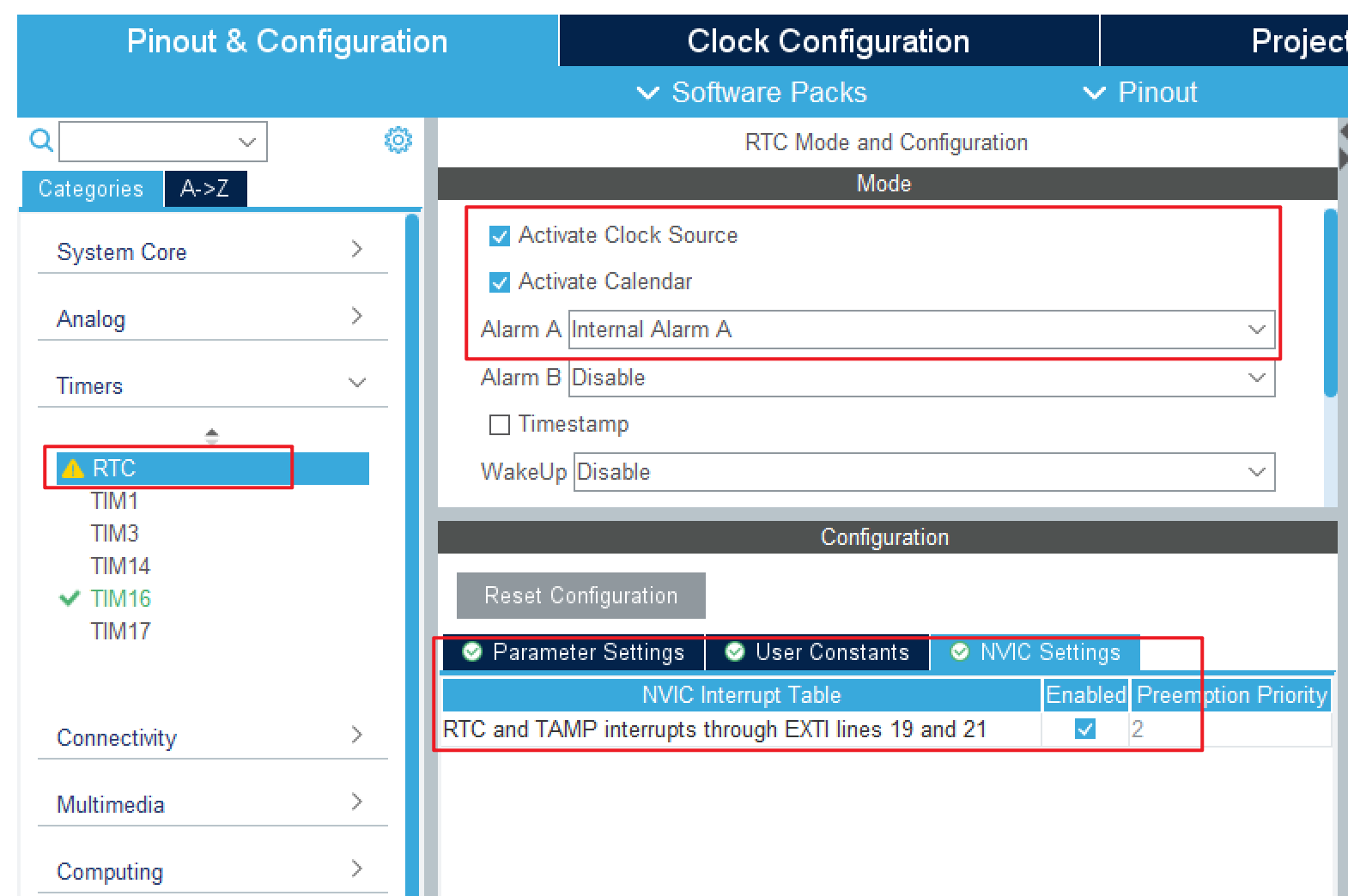




### RTC初始化

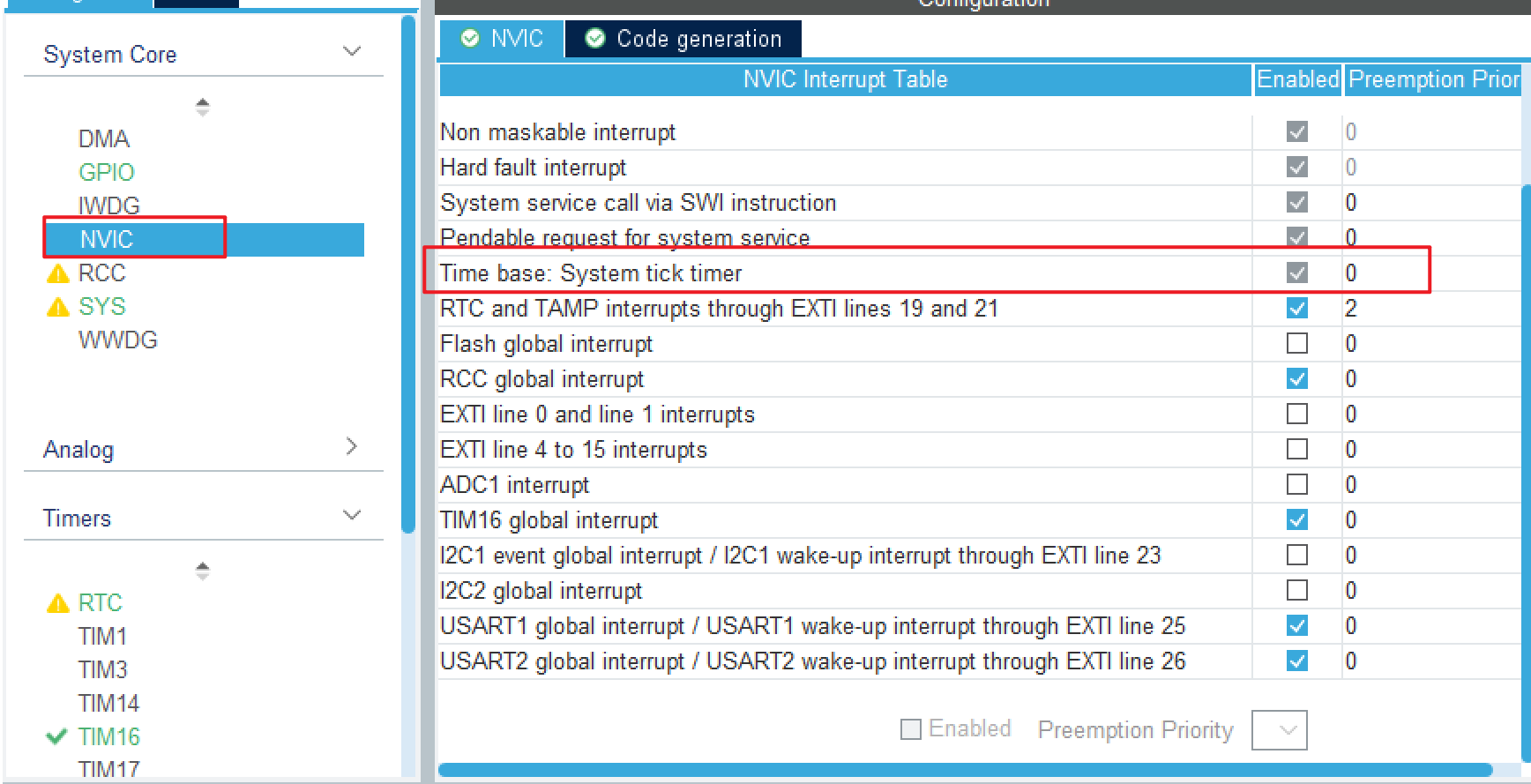
如下图中的RTC开启了中断的功能，但是程序代码初始化的时候并没有开启RTC的中断，程序中开rtc中断是在进入休眠之前才会开启的(可以通过RTC把板子唤醒)。





### Systick初始化

Cubemx里面默认就会开启systick的中断功能，但是我们还是需要在代码中通过LL库的LL\_SYSTICK\_EnableIT()调用去开启systick的中断。



### Sh367306初始化

涉及IIC通讯的代码驱动自己完成，如下：

void SH30x\_I2C\_Start(SIMIIC\_name\_t IICNum)

{

/\* 当SCL高电平时，SDA出现一个下跳沿表示I2C总线启动信号 \*/

I2C\_SDA\_1(IICNum);

Soft\_Delay(1);

I2C\_SCL\_1(IICNum);

Soft\_Delay(4);

I2C\_SDA\_0(IICNum);

Soft\_Delay(4);

I2C\_SCL\_0(IICNum);

Soft\_Delay(1);

}

void SH30x\_I2C\_Stop(SIMIIC\_name\_t IICNum)

{

/\* 当SCL高电平时，SDA出现一个上跳沿表示I2C总线停止信号 \*/

I2C\_SDA\_0(IICNum);

Soft\_Delay(2);

I2C\_SCL\_1(IICNum);

Soft\_Delay(1);

I2C\_SDA\_1(IICNum);

Soft\_Delay(2);

}

void SH30x\_I2C\_SendByte(SIMIIC\_name\_t IICNum , uint8\_t \_ucByte)

{

uint8\_t i;

/\* 先发送字节的高位bit7 \*/

for (i = 0; i < 8; i++)

{

if (\_ucByte & 0x80)

{

I2C\_SDA\_1(IICNum);

}

else

{

I2C\_SDA\_0(IICNum);

}

Soft\_Delay(1);

I2C\_SCL\_1(IICNum);

\_ucByte <<= 1; /\* 左移一个bit \*/

Soft\_Delay(2);

I2C\_SCL\_0(IICNum);

Soft\_Delay(1);

if (i == 7)

{

I2C\_SDA\_1(IICNum); // 释放总线

}

}

}

uint8\_t SH30x\_I2C\_ReadByte(SIMIIC\_name\_t IICNum)

{

uint8\_t i;

uint8\_t value;

/\* 读到第1个bit为数据的bit7 \*/

value = 0;

for (i = 0; i < 8; i++)

{

Soft\_Delay(2);

I2C\_SCL\_1(IICNum);

Soft\_Delay(1);

value <<= 1;

if (I2C\_READ\_SDA(IICNum))

{

value++;

}

Soft\_Delay(1);

I2C\_SCL\_0(IICNum);

}

return value;

}

uint8\_t SH30x\_I2C\_WaitAck(SIMIIC\_name\_t IICNum)

{

uint8\_t re;

I2C\_SDA\_1(IICNum); /\* CPU释放SDA总线 \*/

Soft\_Delay(2);

I2C\_SCL\_1(IICNum); /\* CPU驱动SCL = 1, 此时器件会返回ACK应答 \*/

Soft\_Delay(1);

re = I2C\_READ\_SDA(IICNum); /\* CPU读取SDA口线状态 \*/

Soft\_Delay(1);

I2C\_SCL\_0(IICNum);

Soft\_Delay(2);

return re;

}

void SH30x\_I2C\_Ack(SIMIIC\_name\_t IICNum)

{

I2C\_SDA\_0(IICNum); /\* CPU驱动SDA = 0 \*/

Soft\_Delay(1);

I2C\_SCL\_1(IICNum); /\* CPU产生1个时钟 \*/

Soft\_Delay(2);

I2C\_SCL\_0(IICNum);

Soft\_Delay(2);

I2C\_SDA\_1(IICNum); /\* CPU释放SDA总线 \*/

}

void SH30x\_I2C\_NAck(SIMIIC\_name\_t IICNum)

{

I2C\_SDA\_1(IICNum); /\* CPU驱动SDA = 1 \*/

Soft\_Delay(1);

I2C\_SCL\_1(IICNum); /\* CPU产生1个时钟 \*/

Soft\_Delay(2);

I2C\_SCL\_0(IICNum);

Soft\_Delay(2);

}

对于sh36730x的初始化：

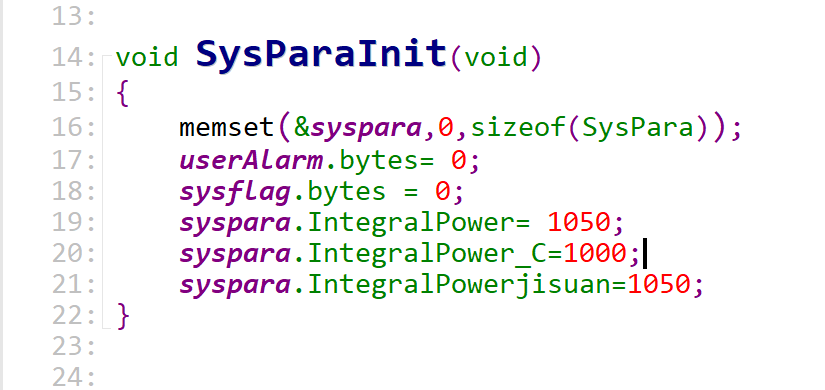


## 初始化flash

备注：进入休眠的时候别的外设都需要重新初始化，falsh不需要；flash只有重新上电才会重新初始化。

## 系统参数初始化

一些程序中用到的全局变量需要初始化一下:



## 循环次数初始化

## 死循环

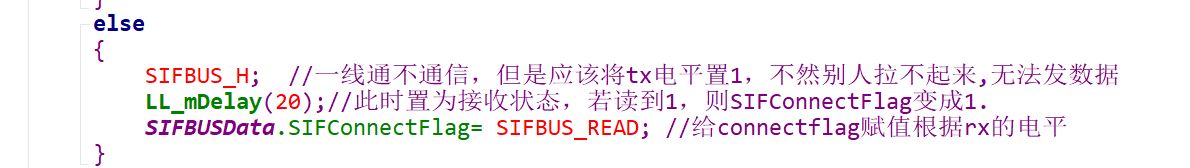
### 485流程处理

485中断中会把PC端发来的数据放进存储缓冲区中，并将RS485Buf. rcvFlag置1。主循环中检测这个rcvFlag然后根据协议规定处理收到的数据，

### 一线通流程处理

注意：用到的2根线用于调试模式时是作uart使用的，对应的是uart的收发模式；在接到车上时就会切换成车载模式，接到仪表上，此时为一线通模式，对应sif的收发模式。

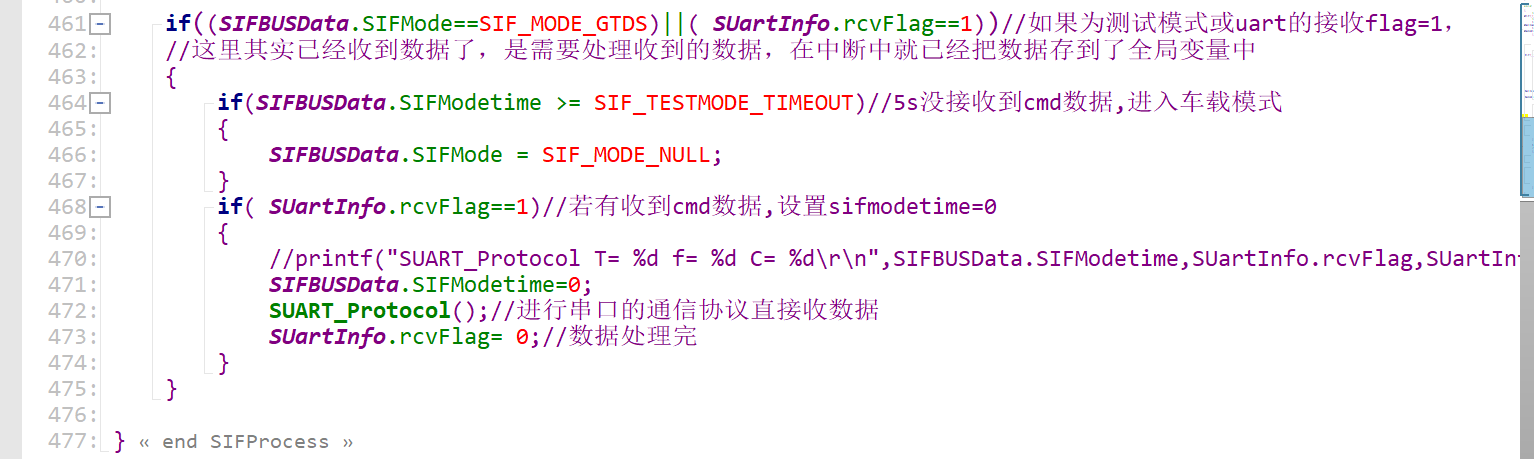
1. 刚开始串口2初始化的时候只初始化了rx引脚，注意USESIF\_FUNCTION=1.此时波特率是14400. 使能了串口2，但是只能接收，不能发送。
2. SIFProcess的函数中，第一次SIFConnectFlag=0执行的时候进入：



SIFBUSData.SIFConnectFlag= SIFBUS\_READ，如果外界给高电平，则读到高电平，如果外界给低电平读到低电平。（若读到1，则SIFConnectFlag变成1，下次进入这个循环的时候就到了）：

CFSIFProtocol();

1. 这个函数中发送数据之前会关闭串口2（disable uart2），此时是sifTxen，设置reload=500，使能time16.（这个time中断隔500us触发一次，中断函数SifTxDataPro中就去发送sif的数据）
2. SifTxDataPro()函数的default分支（没sif数据可发）中，会将串口2又使能，关闭time16，清除reload。
3. Uart2被使能后就能收到串口数据（中断中收），然后rcvFlag=1，会进入SUART\_Protocol 中的BootCmd\_Process ();



1. BootCmd\_Process 函数中发送数据的时候用的模拟uart模式的发送，还是会disable uart2，在tim16的中断中去发送，每隔（100 000/14400/2=34us）进一次中断去发送uart数据。
2. 自始至终，收数据的时候都是串口在收（uart2初始化波特率14400），就是发数据的时候是2种模式都是在TIM16不同的速率中去发的，uart模式波特率（），sif模式波特率（）。

跟它通信的仪表呢？它的接收数据2中波特率不一样呀？它怎么接收的呢？

### 100ms循环任务

### 500ms循环任务1

### 500ms循环任务2

### 是否需要写flash

flash可写标志位，只有485通讯或者一线通通信满足条件才会设置此bit为1，判断这个标志位是否需要flash写操作。