文档编号：AN162

应用笔记

FS030

修订历史

版本	修订日期	修改概要
V1.00	2024-12-17	初版

目 录

内容目录

第1章 FS030 应用注意

1.1 配置字DEBUG功能

调试时：CFG_DEBUGEN 需配置为“DEBUG 自动识别”。

生产正式产品时：CFG_DEBUGEN 需禁止，否则加密编程无效，并且可能会因调试管脚输入悬空而出现芯片休眠功耗异常、抗干扰性能变差等隐患。

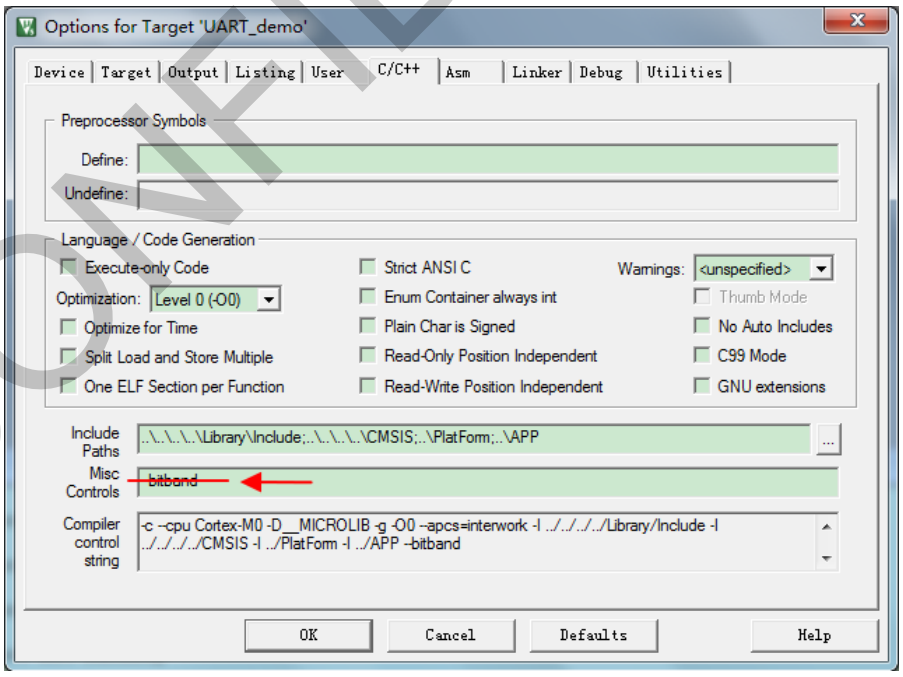
1.2 开发环境

IDE	推荐版本	插件包
Keil4	V4.72 及以上版本	Keil 4 芯片支持包
Keil5	V5.2x 及以上版本	MDK v4 Legacy Support 和 Keil 4 芯片支持包

Keil5 使用说明：在 Keil5 下开发需要进行如下步骤：

1.安装“MDK v4 Legacy Support”（http://www2.keil.com/mdk5/legacy/），然后就可以在 keil5 下安装“Keil 4 芯片支持包”。

2.Keil5 限制用户对 Cortex-M0 进行 bitband 操作，用户需要去除原工程里对 C 编译器的 bitband 配置，如下图：



1.3 寄存器写保护

为避免程序的异常导致运行错误，芯片写保护寄存器用于阻止对被保护的寄存器误操作。

系统控制单元，IWDT，WWDT 等模块支持寄存器写保护，对被保护的寄存器进行写之前需要解除写保护状态（允许写），否则无法对被保护的寄存器写入。操作完成后，再使能写保护（禁止写）。

1.3.1 SCU写保护

系统控制寄存器 SCU 的访问操作会影响整个芯片的运行状态，芯片提供系统设置保护寄存器 SCU_PROT。

对 SCU_PROT 寄存器以字方式写入 0x55AA6996 会解除写保护， 对该寄存器写入其他任何值都会使能写保护。

SCU_PROT 保护的寄存器为 SCU_NMICON，SCU_PWRC，SCU_FAULTFLAG，SCU_FLASHWAIT，SCU_LVDCON，SCU_CCM，SCU_TIMEREN，SCU_TIMERDIS，SCU_SCLKEN0，SCU_SCLKEN1，SCU_PCLKEN，SCU_WAKEUPTIME，SCU_TBLREMAPEN。

库函数提供 SCU_RegUnLock 宏解除写保护，SCU_RegLock 宏使能写保护。

1.3.2 IWDT写保护

对 IWDT_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

IWDT_LOCK 保护的寄存器为 IWDT_LOAD，IWDT_CON，IWDT_INTCLR

库函数提供 IWDT_RegUnLock 宏解除写保护，IWDT_RegLock 宏使能写保护。

1.3.3 WWDT写保护

对 WWDT_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护，对该寄存器写入其他任何值都会使能写保护。

WWDT_LOCK 保护的寄存器为 WWDT_LOAD，WWDT_CON，WWDT_INTCLR

库函数提供 WWDT_RegUnLock 宏解除写保护，WWDT_RegLock 宏使能写保护。

1.4 位操作

Cortex-M0 本身不支持位带操作(bitband)，本芯片为了方便用户操作，为用户扩展了位带功能。

1.4.1 位带扩展原理

SRAM 位带扩展功能，对 SRAM 的每个 bit，都赋予了一个扩展地址，通过该扩展地址，可直接访问其对应的 SRAM 数据位，从而极大的方便了对 SRAM 单元的位读写操作。对于 SRAM 的某个 bit，记它所在字节地址为 A，位序号为 N（0≤N≤7），SRAM 位带扩展映射区的基地址为 0x2200_0000，则该 bit 的位带扩展地址为：

AliasAddress_A_N = 0x2200_0000 + (A – 0x2000_0000) x 32 + N x 4

外设寄存器位带扩展功能，对外设寄存器的每个 bit，都赋予了一个扩展地址，通过该扩展地址，可直接访问其对应的寄存器位，从而极大的方便了对外设寄存器的位读写操作。对于外设寄存器的某个 bit，记它所在字节地址为 A，位序号为 N（0≤N≤7），外设寄存器位带扩展映射区的基地址为 0x4200_0000，则该 bit 的位带扩展地址为：

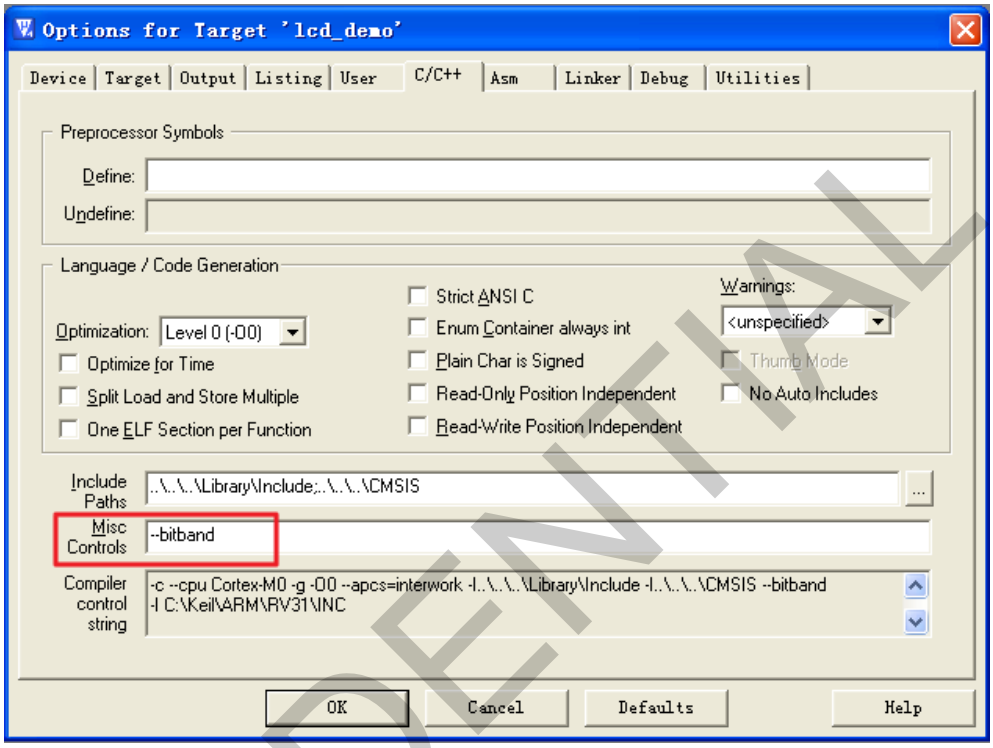
AliasAddress_A_N = 0x4200_0000 + (A – 0x4000_0000) x 32 + N x 4

1.4.2 位带使用方法

1. 直接对位带扩展地址进行读写操作：按照上面的方法计算得到所要操作 bit 的位带扩展地

址，然后直接对其地址进行读写操作。

2. 将外设寄存器或者变量使用 C 语言定义成位域，如果位域变量为 1bit 宽度，并且对 C 编译器进行了如下设置，那么编译出来的代码将自动对其进行位带操作。如果用户没有按照下面的方式对 C 编译器进行“--bitband”设置，那么所有的位域写操作都将使用对原地址进行“读-修改-写”的方式实现。



1.5 写 1 清零寄存器

有很多中断标志寄存器都是用“写 1 清零”的方式来操作。对于“写 1 清零”的寄存器，不可使用“读-修改-写”的方式来进行“写 1 清零”，否则会引起标志误清，进而产生漏中断的后果。

例如对 T16N0 的中断标志寄存器 MAT0IF 进行“写 1 清零”，应该按照如下操作：

```
T16N0->IF.Word = (uint32_t)0x01;
```

禁止进行位操作，如下操作均为错误的写法：

```
T16N0->IF.MAT0IF = 1;
T16N0->IF.Word |= (uint32_t)0x01;
```

因为位操作最终都是按照“读-修改-写”的方式来操作的，这时如果 MAT1IF 也为 1，那么 MAT1IF 就会被误请，从而造成漏中断的后果。

1.6 标志位查询超时机制

在 MCU 程序开发中经常会对标志寄存器进行查询，如下面的例子：等待 xxIF 为 1 后再进行后面的操作。

```
while(xxIF == 0);
```

健壮的系统要避免这种没有时间限制的等待，可以参考下面的例子改善。

```
for(i=0; i<n; i++)
```

```
{
  if(xxIF == 1)
    break;
}
if(i==n)
  return error;
```

提供的标准库并不具体超时机制，用户需要根据实际系统需设计超时机制。以上程序中的“n”也需要根据用户系统时钟和应用场景来确定。

1.7 串行总线操作

串行总线 I2C 发送数据时，需等待 I2C_STA 寄存器的 TBEF0~3 标志置 1，即发送缓冲器全空后才能发送停止位，否则会导致最后装载的数据不能正常发出。

SPI 总线发送数据时，需等待 SPI_STA 寄存器的 IDLE 标志置 1，即发送缓冲器全空后才能关闭发送使能。

UART 总线发送数据时，需等待 UART_STA 寄存器的 TXBUSY 标志清 0，即发送缓冲器全空后才能关闭发送使能。

UART 的 RBIF 和 TBIF 两个标志位为只读，无法直接清除。其中 RBIF 在读取接收缓存后可自动清除；TBIF 在发送缓冲中有数据时可自动清除。因此在使能 RBIE 中断时，在中断服务函数中读取接收缓存 UART_RBR 后可自动清除 RBIF。在使能 TBIE 中断时，在中断服务函数中向发送缓冲器写入下一个想要发送的数据，可自动清除 TBIF；若要停止发送数据，则需在中断服务函数中关闭 TBIE 中断，以避免芯片不停的进入发送缓冲空中断。

1.8 I2C高速从机编程操作

I2C 支持 7 位从机地址匹配，由 I2C 主机控制发送或接收数据。当主机向从机发送数据时，从机通常判断 RBIF 标志，如果接收缓冲器不空，即接收到主机数据，则读接收缓冲器的数据；当主机读取从机数据时，从机可以判断 TIDLEIF 标志，如果发送空闲，则依次写入需要发送的数据。

当 I2C 做从机需要高速传输时，用户需要注意以下几点：

1. 使能时钟线自动下拉功能。在通常情况下，从动器处于释放时钟线的状态，时钟线 SCL 完全由主控器控制。但当从动器出现异常情况，短时间内无法继续进行数据传输时，从动器可以在时钟线 SCL 为低电平时输出 0（不可以在高电平时输出 0，否则会破坏数据传输过程），强行使 SCL 保持低电平，使主控器进入通讯等待状态，直到从动器释放时钟线；
2. 为实现 I2C 时钟线的下拉等待请求功能，还需 I2C_CON 寄存器中配置 SCLOD，将通讯端口 SCL 选择为开漏输出模式，通过上拉电阻提供高电平（复用的 IO 口也需要设置为开漏输出，上拉模式），使从动器可对时钟线下拉控制，使主控器等待；
3. 为避免从机自动下拉时间太长，超出主机的最大等待时间，程序需尽快将数据写入 TWB 寄存器；
4. 为了使代码的效率更高，可以使用直接操作寄存器的方法来控制 I2C 的传输。

1.9 SPI主机操作注意项

当 SPI 做主机时，在 SPI 的“DFS<1:0> = 10，上升沿接收（先），下降沿发送（后）”模式下，必须先将 SPI_CON 寄存器中除 REN 和 EN 的其他配置设置好后，再使能 REN 和 EN（即需

要两条代码完成），否则 SPI 通讯异常，甚至会多出 8 个时钟周期。

1. 10 IAP操作程序

芯片内置 IAP 自编程固化模块， 由硬件电路实现。 IAP 操作既可以放在 SRAM 中执行，也可以调用自编程固化模块，推荐用户调用自编程固化模块，以减少 SRAM 中的 IAP 操作代码量。

在进行 FLASH 编程时，无论是否编写相同的数据，在 FLASH 编程前均必须先进行擦除。禁止通过对一个 word 的多次写入实现按 byte 或按 bit 修改。

1. 11 PWM输出

PWM 占空比配置为 0%时，会产生一个计数周期的相应脉冲。

1. 12 多路PWM同步输出

若要实现多路 PWM 的同步输出，可以通过 SCU_TIMEREN 和 SCU_TIMERDIS 控制寄存器，选择同时启动或关停多个 T16N 定时器来实现。

1. 13 GPIO上电复位后状态

上电复位后通用 GPIO 端口均为高阻状态，端口内部弱上拉和弱下拉均为禁止，但当 PB12 端口作为 GPIO 时，芯片在 VDD 上电过程中，PB12 端口的内部弱上拉均会自动使能，PB12 端口电平跟随 VDD 上升，直到 VDD 上升到芯片开始工作的电压后，该 IO 端口的内部弱上拉恢复为默认的禁止状态。

1. 14 GPIO端口输出电平位操作

GPIO 端口输出电平位操作寄存器 GPIO_PADATABSR、GPIO_PBDATABSR、GPIO_PADATABCR、GPIO_PBDATABCR、GPIO_PADATABRR、GPIO_PBDATABRR 不能进行与或操作，只能按 word 写入。

GPIO 端口输出电平操作时建议用上述寄存器而不是端口寄存器（GPIO_PADATA 和 GPIO_PBDATA），以避免读-修改-写情况的发生。

1. 15 未使用和未封装的GPIO端口处理

系统中未使用和未封装出来的 GPIO 端口建议设置为输出固定电平并悬空，若设置为输入则不可悬空，须加上拉或下拉电阻接到电源或地。

1. 16 ADC模块应用注意事项

ADC 控制寄存器 0（ADC_CON0）中的 A/D 转换状态位（TRIG）和 A/D 转换使能位（EN）必须按字方式进行改写（如：ADC->CON0.Word |= ADC_EN_MASK; ADC->CON0.Word |= ADC_TRIG_MASK;)；否则会影响 A/D 分辨率选择位（BIT_SEL）。

当选择内部参考电压 VREF 2.048V 作为 ADC 正向参考电压时，需先设置 ADC_VREFCON 寄存器的 VREF_EN 位使能内部参考，并设置 IREF_EN 和 ADC_CON0 寄存器的 EN 位使能 ADC，然后等待 300us 以后，再设置 CHOP_EN 位使能参考电压斩波器，否则内部参考电压可能不稳定，然后延时 1ms 以上，ADC 工作建立完成（否则有可能导致 ADC 转换异常），再启动 ADC 转换（TRIG=1），可得到正确的转换结果。

因每次 IREF_EN，VREF_EN，CHOP_EN，A/D 转换使能位 EN 重新使能后，均需要执行上

述 ADC 工作建立过程，所以应用中，在芯片正常运行时不建议关闭上述 4 个使能控制信号，保持为 1，只在进入深睡眠模式前，关闭 ADC。

为了保证 ADC 转换结果的稳定可靠、避免噪声干扰，建议在模拟输入通道接外部电容（100nF 或 10nF）进行滤波。

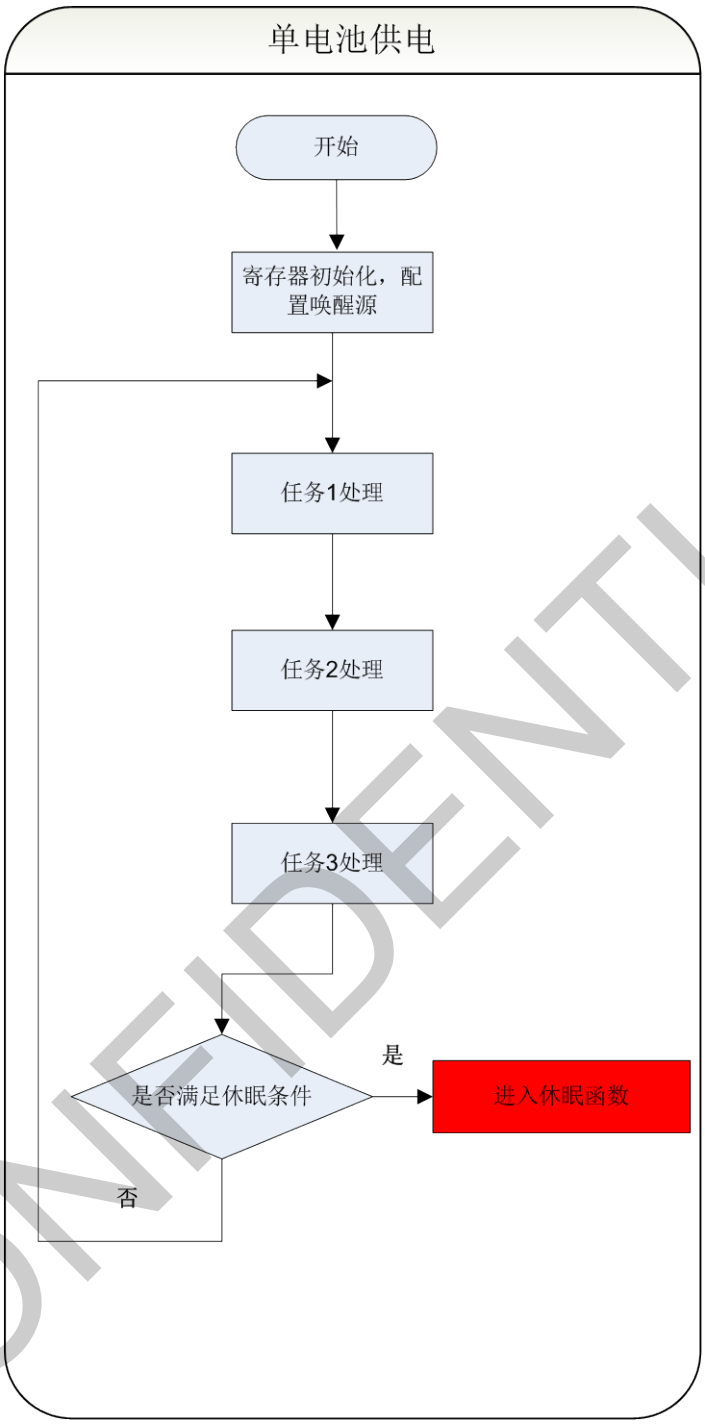
1. 17 低功耗系统程序设计注意事项与推荐结构

在进行低功耗系统程序设计时需要注意以下几点：

1. 芯片调试完毕，生产正式产品时，GFG_DEBUGEN 配置字需禁止。GFG_DEBUGEN 配置字禁止后，SWD 的二个端口状态由用户程序决定。如果用户将 SWD 端口设为输出口，要注意 SWD 端口外围电路是否会有漏电;如果用户将 SWD 端口设为输入口,要注意 SWD 端口不能悬空。
2. 系统唤醒时间控制寄存器 SCU_WAKEUPTIME.WAKEUPTIME 的值必须大于等于 0x120。
3. 建议悬空的 GPIO 输出固定电平，有上下拉的 GPIO 输出相应固定电平。输入功能的 IO 不可悬空。
4. 进入休眠前需清除所有中断挂起标志位，并在清除中断挂起标志位的指令和进休眠模式的指令之间，延时至少一个 NOP 指令周期。
5. 可靠的系统不应该在系统运行的过程中关闭看门狗，休眠时也不例外。建议休眠时将 WDT 做为唤醒源，唤醒后立即清狗，再让芯片进入深睡眠状态，这样的处理方式对系统平均功耗的增加可忽略不计。
6. 当用户使用的外部 MRSTN 复位功能唤醒深睡眠模式时，必须在芯片配置字中设置“PWRTEN”使能（上电 140ms 延时使能），并且 PB12 端口在唤醒期间也会出现短暂的内部弱上拉自动使能，维持时间约 300us。

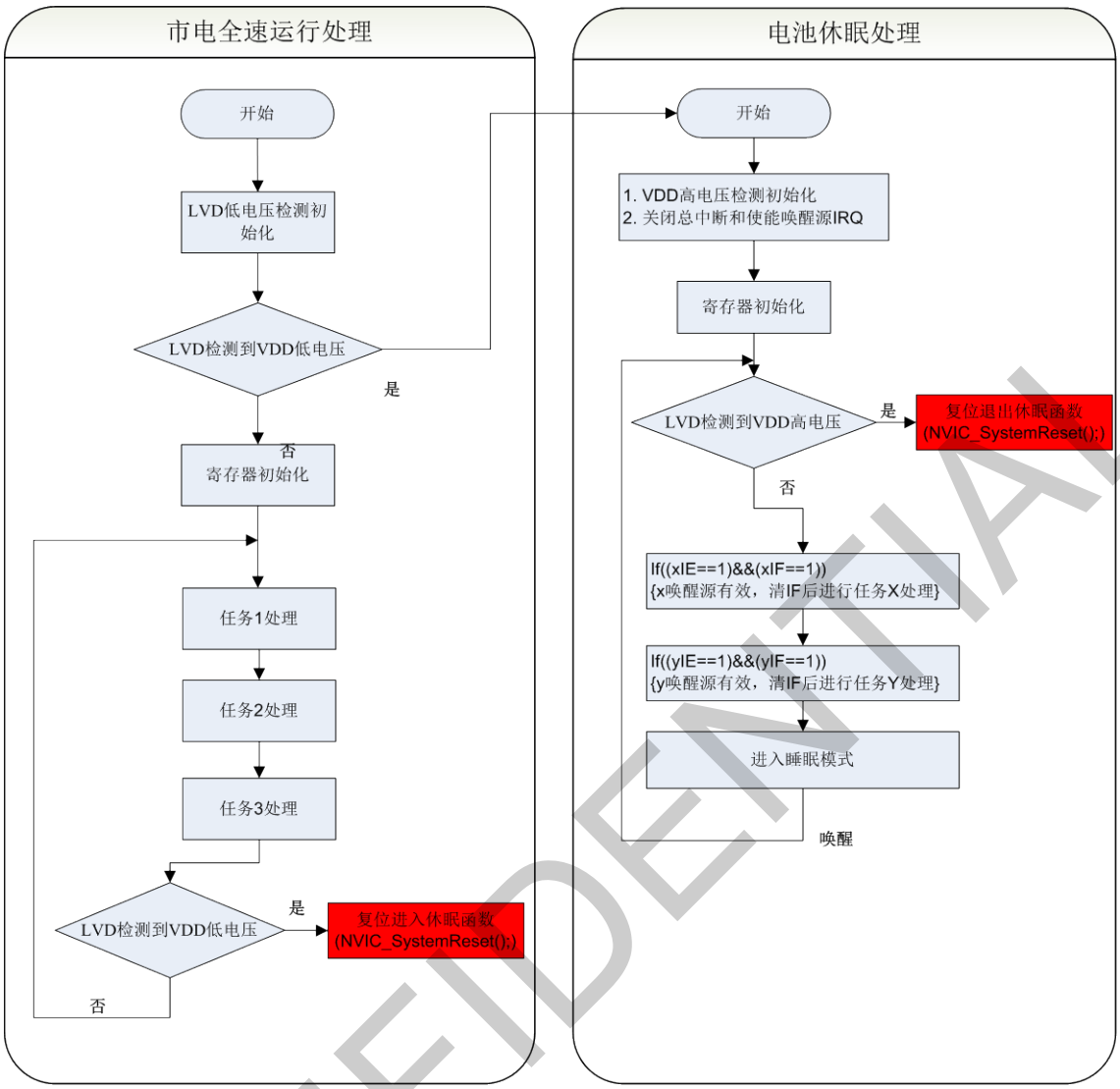
1. 17. 1 单电池供电系统低功耗设计

仅仅由电池供电的系统通常为休眠状态，并具备若干中断作为唤醒源，当中断产生并进入对应中断服务程序执行任务；中断服务程序执行完毕后，继续进入休眠状态。



1. 17. 2 电池和市电同时供电系统低功耗设计

1. 为使市电全速运行处理和电池休眠处理尽量减少相互影响，分别为市电全速运行处理和电池休眠处理各自独立设计初始化和循环体，并且用系统复位(NVIC_SystemReset();)来切换这两种工作状态。
2. 为使市电全速运行处理和电池休眠处理尽量减少相互影响，休眠状态下的中断服务采用查询方式进行。休眠函数初始化需要关闭总中断(__disable_irq();)，禁止在休眠函数中响应任何中断服务程序，并使能相应唤醒源 IRQ(NVIC_EnableIRQ(XXX_IRQn);)。



1. 18 外设时钟

系统程序中一定不能反复使能和关闭外设时钟，否则会有程序运行异常的风险。同理，在睡眠执行休眠指令前，也无需软件关闭外设时钟，休眠指令会关闭 Fosc 时钟，外设时钟也被自动停止。

1. 19 系统时钟

1. 19. 1 系统时钟滤波器注意事项

CLKFLT 为系统时钟滤波器。当系统时钟为 48MHz 时，需旁路 CLKFLT，否则可能会造成系统时钟有时失效；当系统时钟为其它时钟源时，则不建议旁路 CLKFLT，可进一步提升系统工作稳定性。

1. 19. 2 外部XTAL时钟注意事项

在使用外部时钟 XTAL 时，推荐提前将唤醒时间控制位 WAKEUPTIME<11:0>设置为 0xFFF，以增强振荡器工作稳定标志位 XTAL_RDY 的可靠性，然后再使能外部晶振。