文档编号：AN_095

上海东软载波微电子有限公司

应用笔记

ES8P508x/ES8H508x/ES8H507x/ES8H69x

修订历史

版本	修订日期	修改概要
V1.0	2017-12-27	初版
V1.1	2018-05-08	1. 增加“对未使用 GPIO 的处理”说明 2. 增加“低功耗系统程序设计注意事项与推荐结构”
V1.2	2019-01-02	1. 增加 IWDT 默认复位周期的说明 2. 低功耗程序框架调整："__WFI()"前后增加"__NOP()"；清挂起标志提到休眠语句前
V1.3	2019-4-25	变更 Logo。
V1.4	2020-01-07	1.增加未封装 IO 的处理意见 2.增加 IAR 和 Keil5 开发环境说明 3.增加“标志位查询超时机制”章节 4.增加“IAP 防误擦”说明
V1.5	2020-05-20	1.增加 debug 配置字必须在正式产品生产时禁止的说明 2. 增加 LQFP32 封装的 ADC 负向参考电压选择不支持“内部地参考电压”的说明
V1.6	2020-9-21	1.去除使用位域进行“写 1 清零”
V1.07	2021-10-11	1.修正 GPIO 端口输出电平位操作 2.添加 IWDT 使用 LRC 时钟的注意事项 3.添加“单电池供电系统低功耗设计”流程图
V1.08	2021-1-10	1.增加 SPI 主机编程操作注意项 2.补充低功耗程序设计说明

地　址：中国上海市徐汇区古美路 1515 号凤凰园 12 号楼 3 楼
邮　编：200235
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目 录

内容目录

第 1 章 ES8P508x 应用注意 4
1.1 配置字 DEBUG 功能 4
1.2 开发环境 4
1.3 寄存器写保护 5
1.3.1 SCU 写保护 5
1.3.2 IAP 写保护 5
1.3.3 RTC 写保护 5
1.3.4 IWDT 写保护 5
1.3.5 WWDT 写保护 5
1.3.6 CRC 写保护 5
1.4 位操作 6
1.4.1 位带扩展原理 6
1.4.2 位带使用方法 6
1.5 写 1 清零寄存器 7
1.6 标志位查询超时机制 7
1.7 IWDT 使用 LRC 时钟的注意事项 8
1.8 串行总线操作 8
1.9 I2C 高速从机编程操作 8
1.10 SPI 主机操作注意项 9
1.11 IAP 操作程序 9
1.12 多路 PWM 同步输出 9
1.13 硬件独立看门狗（IWDT） 9
1.14 GPIO 端口输出电平位操作 9
1.15 GPIO 端口灌电流的限制要求 9
1.16 未使用和未封装的 GPIO 端口处理 9
1.17 ADC 模块应用注意事项 10
1.18 PLL 系统时钟选择注意事项 10
1.19 低功耗系统程序设计注意事项与推荐结构 10
1.19.1 单电池供电系统低功耗设计 10
1.19.2 电池和市电同时供电系统低功耗设计 11
1.20 IAP 防误擦 12

第1章 ES8P508x应用注意

1.1 配置字DEBUG功能

调试时：CFG_DEBUG 要使能。

生产正式产品时：CFG_DEBUG 必须禁止，否则加密无效、休眠异常、抗干扰性能变差。

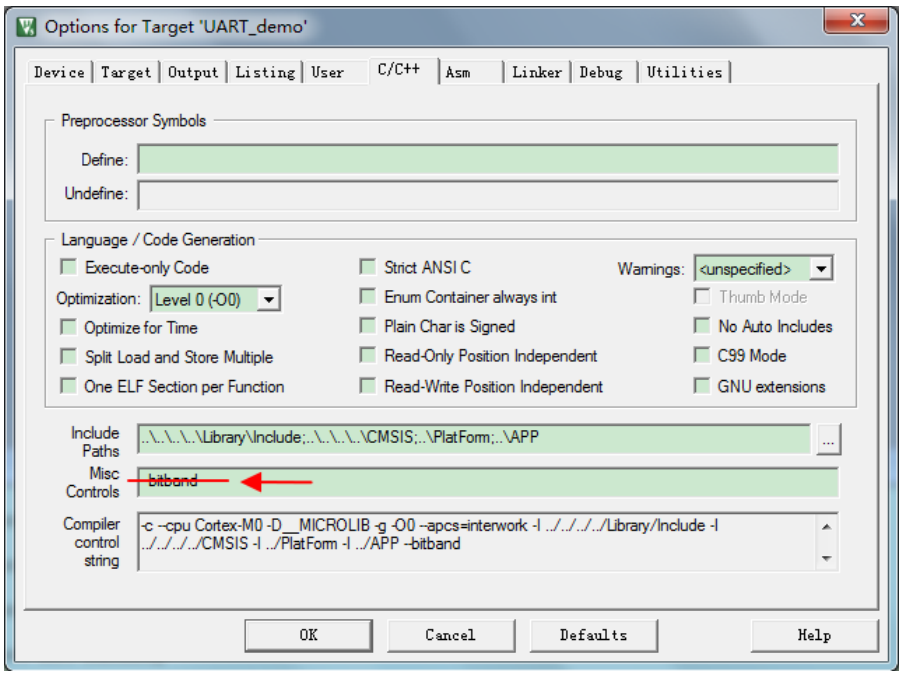
1.2 开发环境

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
IAR	IAR for ARM 8.11.1 及以上版本	IAR 插件
iDesigner	使用 essemi 官网最新版本	-

Keil5 使用说明：8P/8H 产品在 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， RTC，WDT 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），否则无法对写保护寄存器写入。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_WAKEUPTIME， SCU_MRSTN_SOFT， SCU_DBGHALT， SCU_FLASHWAIT，SCU_SOFTCFG， SCU_LVDCON， SCU_CCM， SCU_PLLLKCON， SCU_SCLKEN0，SCU_SCLKEN1， SCU_PCLKEN0， SCU_PCLKEN1， SCU_PRSTEN0， SCU_PRSTEN1，SCU_TIMEREN，SCU_TIMERDIS，SCU_TBLREMAPEN，SCU_TBLOFF。

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

1.3.2 IAP写保护

对 IAP_UL 寄存器以字方式写入 0x4941_5055 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。

IAP_UL 保护的寄存器为 IAP_CON，IAP_ADDR，IAP_DATA，IAP_TRIG。

库函数提供 FlashIAP_RegUnLock 宏解除写保护，FlashIAP_RegLock 宏使能写保护。

1.3.3 RTC写保护

对 RTC_WP 寄存器以字方式写入 0x55AAAA55 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。

可通过读 RTC_WP 寄存器确认 RTC 模块是否处于写保护状态，读出值为 0x55AAAA55，表示当前处于解除写保护状态；读出值为 0x00000000 表示 RTC 模块处于使能写保护状态。

RTC_WP 保护的 RTC 寄存器为 RTC_CON，RTC_CAL，RTC_WA，RTC_DA，RTC_HMS，RTC_YMDW，RTC_IE，RTC_IF。

1.3.4 IWDT写保护

对 IWDOG_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。

IWDOG_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。

1.3.5 WWDT写保护

对 WWDOG_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。

WWDOG_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。

1.3.6 CRC 写保护

对 CRC_UL 寄存器以字方式写入 0x4352_4355 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。

CRC_UL 保护的寄存器为 CRC_CON，CRC_TRIG，CRC_ADDR，CRC_SIZE，CRC_DI，CRC_DO，CRC_STA

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：

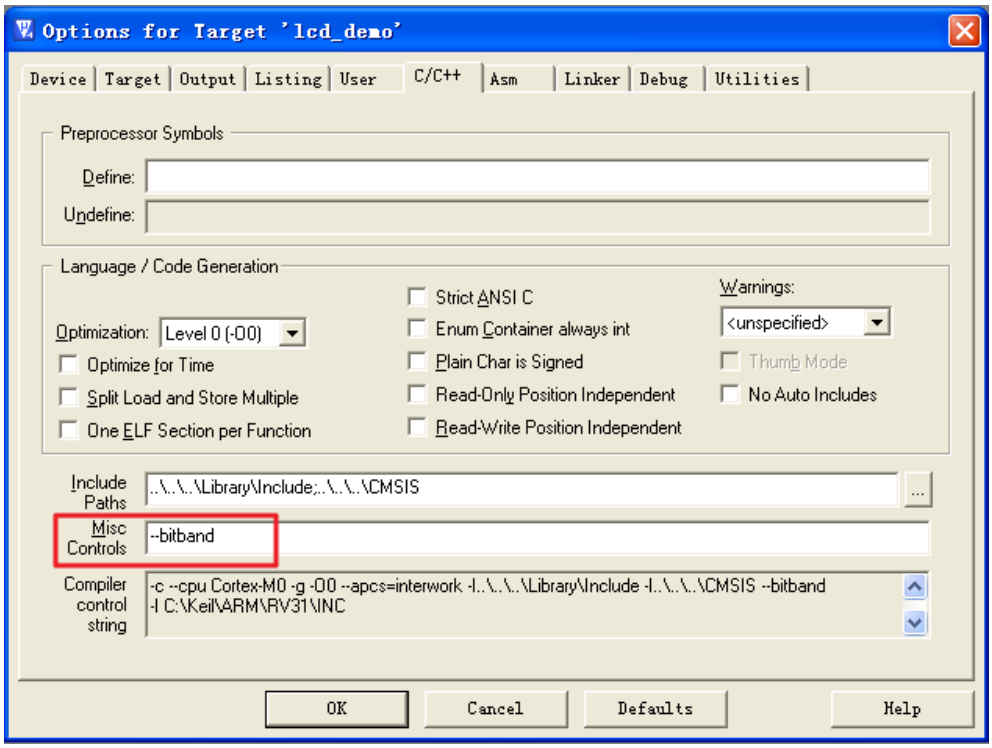
AliasAddress_A_N = 0x2200_0000 + (A – 0x2000_0000) x 32 + N x 4

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”，否则会引起标志位误清，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;
```

8P/8H 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 IWDT使用LRC时钟的注意事项

当 IWDT 作为硬件看门狗使用，或作为软件看门狗使用时将时钟源配置为 LRC 时钟，两次喂狗的时间间隔应小于 IWDT 的计数溢出时间。例如：当 IWDT_LOAD 值为 0x0000_4000 时，IWDT 的最小溢出时间约为 0.32s（LRC 频率最大为 51.2KHz），则用户程序的喂狗间隔应小于 0.32s。

当 IWDT 使用 LRC 作为计数时钟源时，必须禁止 NVIC 寄存器的 IWDT 中断响应 IRQ，避免因干扰等因素误产生 IWDT 中断，而 IWDT 中断服务程序中没有清狗指令时，导致循环反复执行中断服务程序。在深度睡眠模式下，必须将将寄存器 SCU_WAKEUPTIME 的 CLKFLT_EN 和 FLASHPW_PD 位，均设置为 0。

1.8 串行总线操作

串行总线 I2C 发送数据时，需等待 I2CxTBEF0~3 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 SPIxIDLE 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 UxTXIDLE 标志置 1，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 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。

1.9 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。

1.10 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。

1.11 IAP操作程序

ES8P508x 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 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。

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，选择性同时启动或关停多个 T16N/T32N 定时器来实现。

1.13 硬件独立看门狗（IWDT）

IWDT 如果通过配置字使能为硬件看门狗，则计数时钟固定为 LRC 时钟，默认的计数周期典型值约 0.5 秒，用户需要保证程序的喂狗周期，并留有足够的裕量。用户可用通过修改 IWDT_LOAD 寄存器来调整计数周期。

当 IWDT 做为软件看门狗使用时，配置字必须禁止 IWDT，用户须通过寄存器对 IWDT 进行初始化。

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_PADATABCR，GPIO_PADATABRR， GPIO_PADIRBSR， GPIO_PADIRBCR， GPIO_PADIRBRR，GPIO_PBDATABSR， GPIO_PBDATABCR， GPIO_PBDATABRR， GPIO_PBDIRBSR，GPIO_PBDIRBCR，GPIO_PBDIRBRR 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 GPIO端口灌电流的限制要求

对于 PA6~PA13 端口，当使用其中某几个端口作为输出驱动时，总的灌电流不要超过 40mA，否则可能会导致这些 IO 端口中的其它端口输出的低电平被抬高到 0.4V 以上。

1.16 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。

注：ES8P5086 LQFP32 封装的 PA9 端口在封装内部已经接地，用户程序必须将 PA9 配置成模拟口，并关闭其内部弱上下拉。

1. 17 ADC模块应用注意事项

1.用户不可使用 ADC 的 VDD 检测功能，即 ADC 通道选择寄存器（ADC_CHS）的 bit8（VDD5_FLAG_EN）需设为 0。

2.ES8P5086FJLK/ ES8P5088FLLK LQFP32 封装的 ADC 负向参考电压选择不支持“内部地参考电压”，用户只能使用“外部参考电压 AVREFN”。PA9 端口在封装内部已经接地，建议用户程序将 ADC 负参考电压设置为外部参考电压 AVREFN（即 PA9 配置成模拟口，并关闭其内部弱上下拉）。

1. 18 PLL系统时钟选择注意事项

1. CLKFLT 为系统时钟滤波器。当系统时钟为 PLL 输出 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。
2. SCU_SCLKEN0.PLL_MUX 寄存器，置 1 后，软件复位无法还原为 0，需要硬件复位。

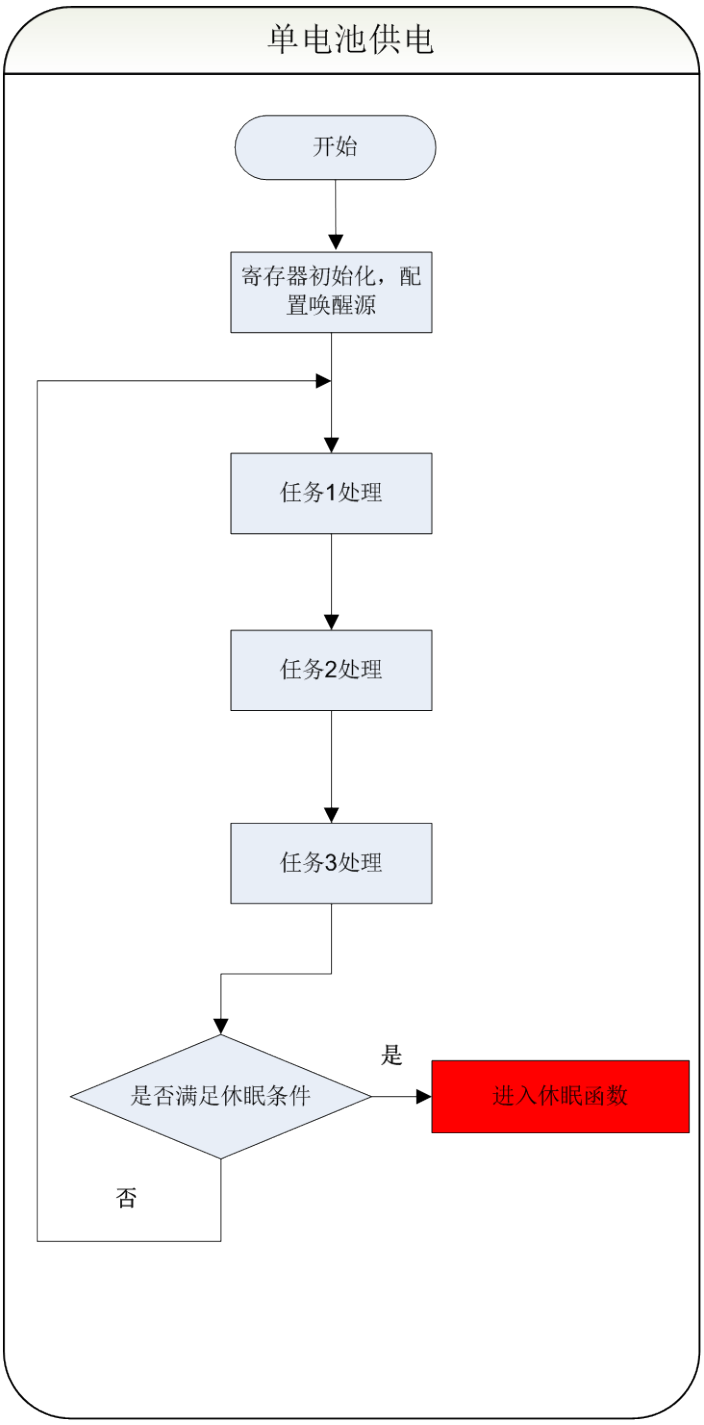
1. 19 低功耗系统程序设计注意事项与推荐结构

在进行低功耗系统程序设计时需要注意以下几点：

1. GFG_DEBUG 配置字必须禁止。
2. 建议悬空的 GPIO 固定输出低电平，有上下拉的 GPIO 输出相应固定电平。输入功能的 IO 不可悬空。
3. 系统唤醒时间控制寄存器 SCU_WAKEUPTIME.WAKEUPTIME 的值必须大于等于 0x3FF。
4. 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 WWDT 做为唤醒源，WWDT 唤醒对 IWDT 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。

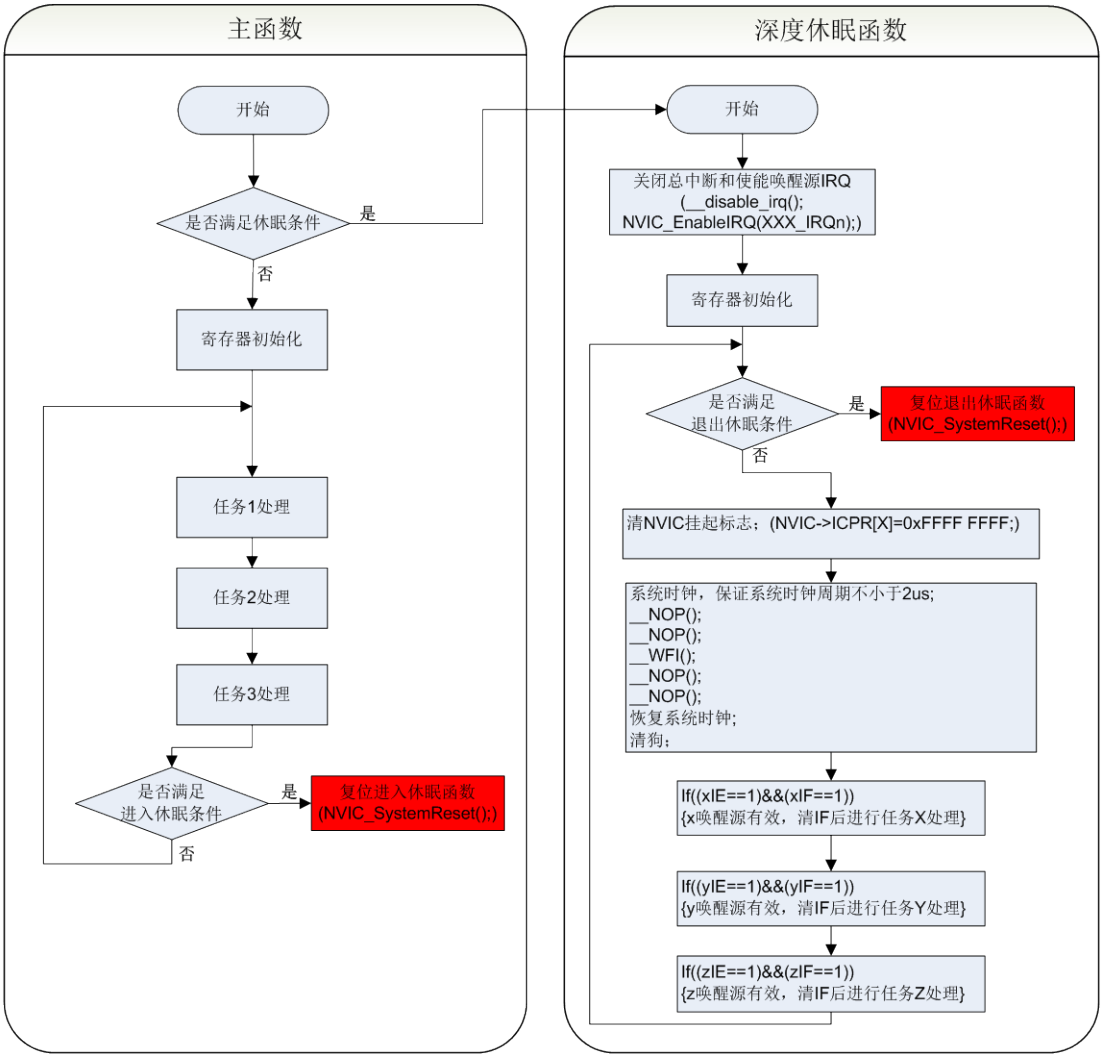
1. 19. 1 单电池供电系统低功耗设计

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. 19. 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. 20 IAP防误擦

在 EMC 干扰较强的系统，为防止 PC 跑飞到 IAP 自编程固化模块并执行，可以禁止 IAP 模块时钟。这样即使 PC 跑飞到 IAP 自编程固化模块并执行，也不会对 Flash 的内容产生任何影响。可以通过 SCU_PCLKEN0 的 IAP_EN 位来使能和禁止 IAP 模块的时钟。需要注意的是 IAP_EN 的写操作受到 SCU_PROT 的保护。