

文档编号: AN_073

上海东软载波微电子有限公司

应用笔记

HR8P506

修订历史

版本	修订日期	修改概要
V1.0	2016-03-29	初版
V1.1	2016-11-07	增加的内容： 1.多路 PWM 同步输出的实现； 2.GPIO 端口灌电流的限制要求。
V1.2	2017-3-9	1. 更新 2.2 T16N 脉宽调制章节内容及例程； 2. 新增 2.3 T16N 双边脉宽调制章节； 3. 补充说明 GPIO 端口输出电平位操作方式。
V1.3	2017-07-06	增加“位操作”、“写 1 清零寄存器”、“开发环境”三节
V1.4	2017-10-12	1. 增加未使用 GPIO 端口的处理方法； 2. 更新配置字 DEBUG 功能小节； 3. 更新 ADC 模块应用注意事项关于 VDD 检测控制位 VDD5_FLAG_EN 的使用描述。
V1.5	2018-01-17	1. 删除复位部分描述 2. 删除模块例程 3. 增加 I2C 高速从机编程操作章节
V1.6	2018-05-08	1. 增加“低功耗系统设计注意事项与推荐结构” 2. 增加 ADC 模块关于转换建立过程描述及注意事项 3. PLL 时钟源选择 LRC 或 LOSC 的注意事项
V1.7	2018-10-23	1.增加外设时钟开关描述 2.增加 PWM 中断描述
V1.8	2018-12-25	1.增加 LQFP48 封装型号芯片的 ADC 使用注意事项 2.增加进休眠模式前清所有中断挂起标志位的注意事项 3.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
V1.9	2019-04-25	变更 Logo。
V1.10	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
V1.11	2020-03-16	1.删除“HR8P506FHLQ ADC 特殊注意事项” 2.增强对配置字 debug 功能和调试端口的描述 3.简化 ADC 模块应用注意事项章节的描述 4.简化 EUART 在 UART 应用时端口操作章节的描述
V1.12	2020-9-21	1.禁止使用位域进行“写 1 清零”

地 址：中国上海市龙漕路 299 号天华信息科技园 2A 楼 5 层
邮 编：200235

E-mail: support@essemi.com

电 话：+86-21-60910333

传 真：+86-21-60914991

网 址：<http://www.essemi.com/>

版权所有©

上海东软载波微电子有限公司

本资料内容为上海东软载波微电子有限公司在现有数据资料基础上慎重且力求准确无误编制而成，本资料中所记载的实例以正确的使用方法和标准操作为前提，使用方在应用该等实例时请充分考虑外部诸条件，上海东软载波微电子有限公司不担保或确认该等实例在使用方的适用性、适当性或完整性，上海东软载波微电子有限公司亦不对使用方因使用本资料所有内容而可能或已经带来的风险或后果承担任何法律责任。基于使本资料的内容更加完善等原因，上海东软载波微电子有限公司保留未经预告的修改权。使用方如需获得最新的产品信息，请随时用上述联系方式与上海东软载波微电子有限公司联系。

目 录

内容目录

第 1 章	HR8P506 应用注意	5
1.1	配置字 DEBUG 功能	5
1.2	开发环境	5
1.3	寄存器写保护	5
1.3.1	SCU 写保护	6
1.3.2	GPIO 写保护	6
1.3.3	RTC 写保护	6
1.3.4	WDT 写保护	6
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	7
1.5	写 1 清零寄存器	7
1.6	标志位查询超时机制	8
1.7	串行总线操作	8
1.8	I2C 高速从机编程操作	8
1.9	IAP 操作程序与防误擦	9
1.10	多路 PWM 同步输出	9
1.11	PWM 中断	9
1.12	GPIO 端口输出电平位操作	9
1.13	GPIO 端口灌电流的限制要求	9
1.14	未使用和未封装的 GPIO 端口处理	9
1.15	ADC 模块应用注意事项	10
1.16	EUART 在 UART 应用时端口操作	10
1.17	低功耗系统程序设计注意事项与推荐结构	10
1.18	PLL 时钟源选择 LRC 或 LOSC 的注意事项	11
1.19	外设时钟	11

第1章 HR8P506 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 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。

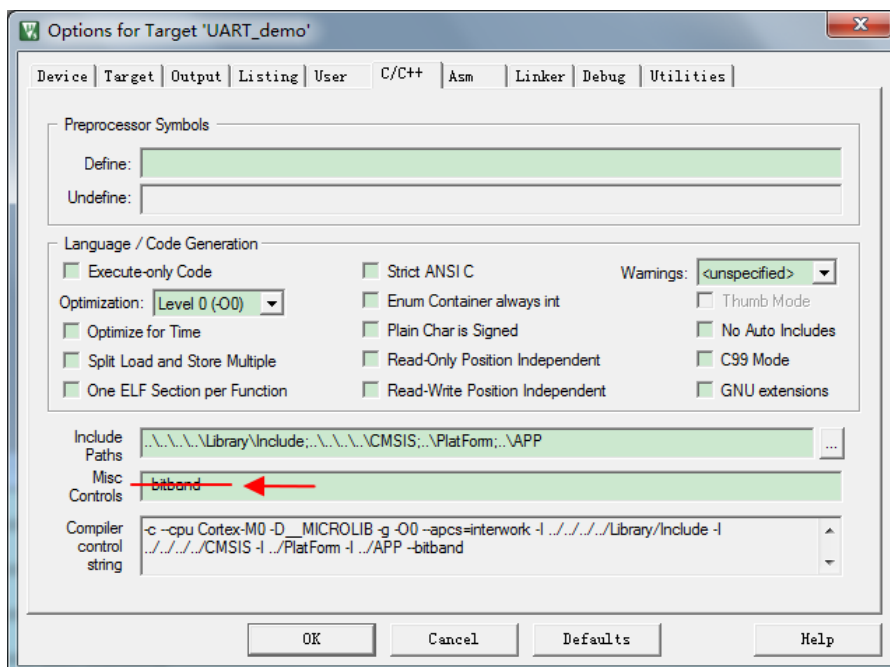
1.2 开发环境

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
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，GPIO，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 NMICON, PWRC, LVDxCON, CLKEN_PERIx, CLKEN_SYSx, TBLREMAPEN, 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 GPIO写保护

对 GPIO_PROT 寄存器以字方式写入 0x78879669 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。

GPIO_PROT 保护的 GPIO 寄存器为 PA_FUNCx, PB_FUNCx, PA_WSEL, PB_WSEL, PA_WEN, PB_WEN, PA_DS, PB_DS 和 PWMOC。

库函数提供 GPIO_RegUnlock 宏解除写保护，GPIO_RegLock 宏使能写保护。

1.3.3 RTC写保护

对 RTCWP 寄存器以字方式写入 0x55AAAA55 会解除写保护，对 RTC 模块其它寄存器执行一次写操作后自动恢复到写保护状态。

若要再写 RTC 模块其它寄存器，仍需重新对 RTCWP 寄存器写 0x55AAAA55 解除写保护，再执行写操作。

可通过读 RTCWP 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。

RTCWP 保护的 RTC 寄存器为 RTCCON, RTCWA, RTCDA, RTCHMS 和 RTCYMDW。

1.3.4 WDT写保护

对 WDOGLOCK 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。

WDOGLOCK 保护的寄存器为 WDOGLOAD, WDOGCON, WDOGINTCLR

库函数提供 WDT_RegUnlock 宏解除写保护，WDT_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（ $0 \leq N \leq 7$ ），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：

$$\text{AliasAddress_A_N} = 0x2200_0000 + (A - 0x2000_0000) \times 32 + N \times 4$$

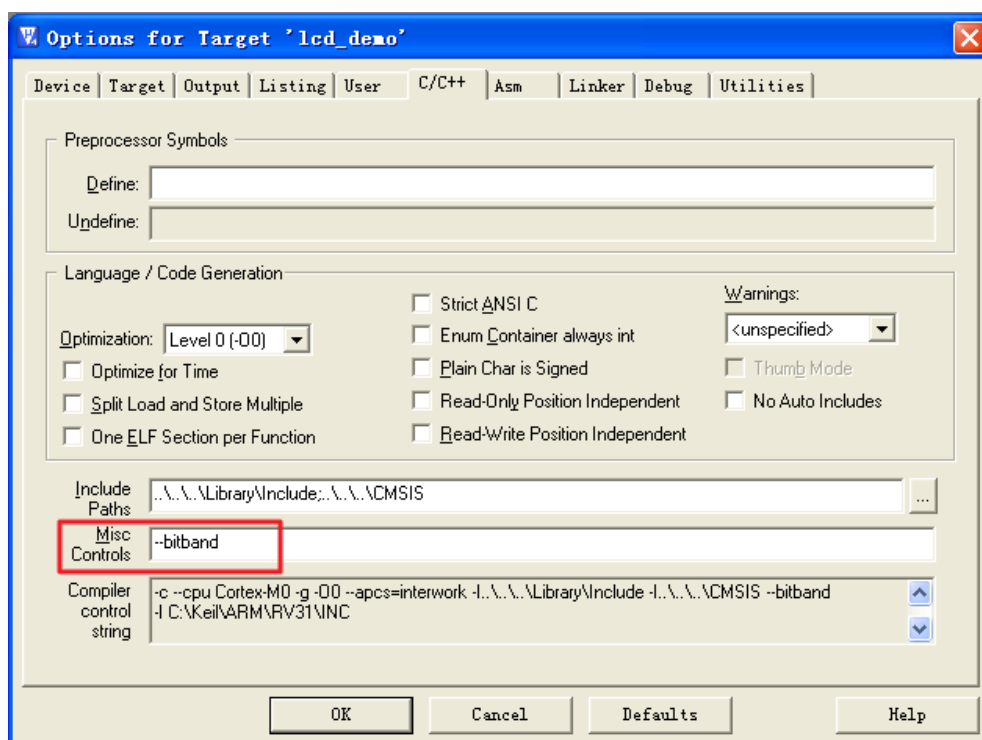
外设寄存器位带扩展功能，对外设寄存器的每个 bit，都赋予了一个扩展地址，通过该扩展地址，

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：

$$\text{AliasAddress_A_N} = 0x4200_0000 + (A - 0x4000_0000) \times 32 + N \times 4$$

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 就会被误清，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 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 / EUART 总线发送数据时，需等待 `UxTXIDLE / EUxTXIDLE` 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。

1.8 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` 完全由主控制器控制。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，使主控制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，使主控制器等待；

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 IAP操作程序与防误擦

HR8P506 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。

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_PCLKEN 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。

1.10 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。

1.11 PWM中断

在 T16Nx 的 PWM 独立模式下，通过 T16N_CNT1 匹配 T16N_MAT2/ T16N_MAT3，控制 T16NxOUT1 输出，但匹配中断标志 MAT2IF 和 MAT3IF 的产生，仍与 T16N_CNT0 计数寄存器有关，使得这两个匹配中断与 T16NxOUT1 的 PWM 波形对应不一致，所以在 PWM 独立模式下，不能使用中断标志 MAT2IF 和 MAT3IF。

1.12 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位操作寄存器而不是端口寄存器，以避免读-修改-写情况的发生。

1.13 GPIO端口灌电流的限制要求

1) 对于 PB13、PA0~PA5 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。

2) 对于 PB13、PA0~PA13 端口，当使用其中某几个端口作为输出驱动时，在满足上述条件 1) 的同时需满足总的灌电流不要超过 60mA，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。

1.14 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。

1.15 ADC模块应用注意事项

1) 屏蔽 ADC 的 VDD 检测, 即 ADC 通道选择寄存器 (ADC_CHS) 的 bit8 (VDD5_FLAG_EN) 设为 0。

2) 在 IREF_EN, VREF_EN, A/D 转换使能位 EN 使能后, ADC 需要先完成自身工作建立, 才能得到正确的转换结果。上述控制信号使能后, 延时 100us 以上, 启动第一次 ADC 转换 (TRIG=1), 转换结束后, 再延时 50us 以上, ADC 工作建立完成。

由于 ADC 建立过程中得到的转换结果与理论值偏差极大且不可预知, 所以在应用程序中需要丢弃 IREF_EN, VREF_EN, A/D 转换使能位 EN 使能后的第一次转换结果。

因每次 IREF_EN, VREF_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 3 个使能控制信号, 保持为 1, 只在进入深睡眠模式前, 关闭 ADC。

1.16 EUART在UART应用时端口操作

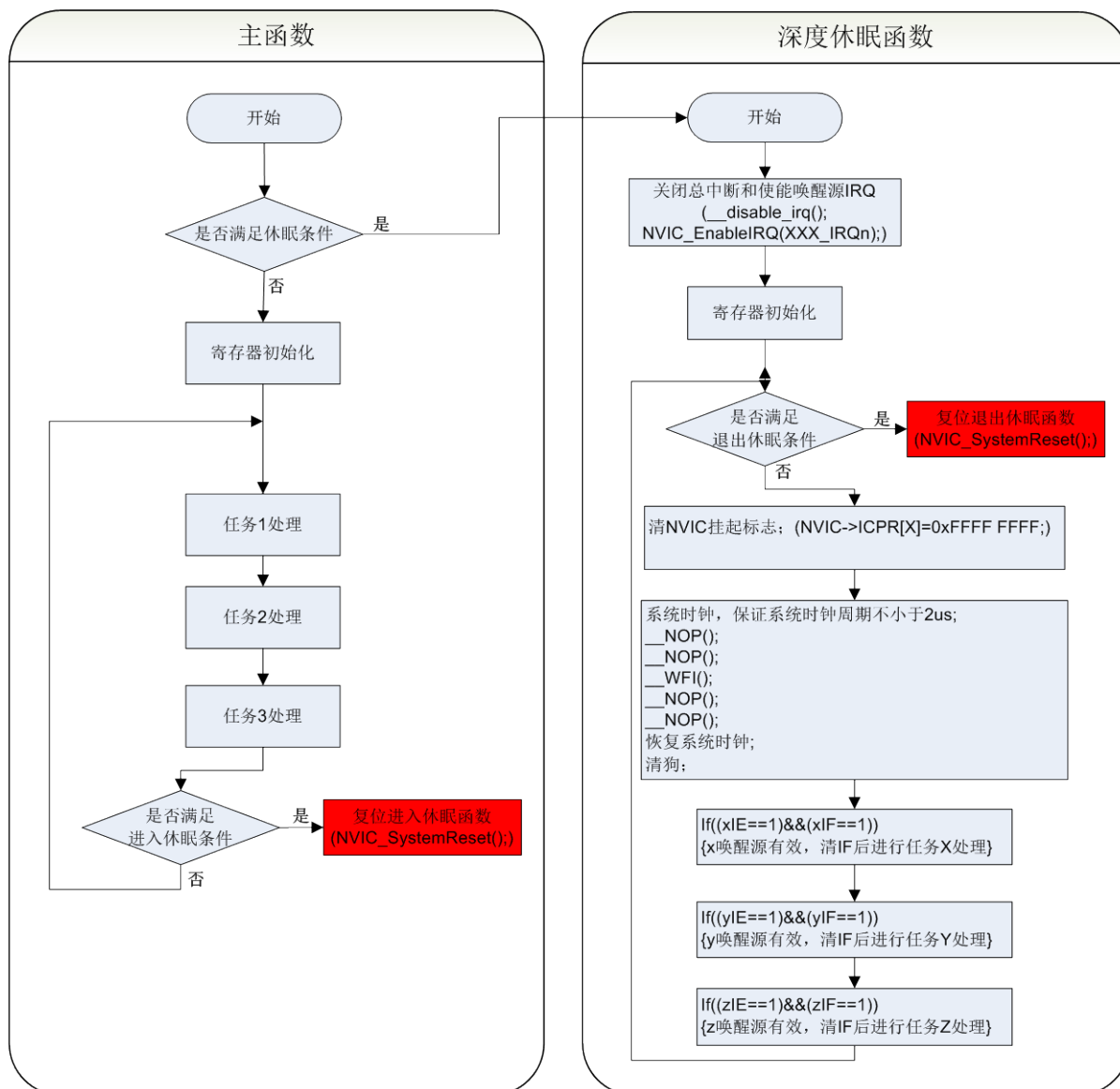
使用 PA12 或 PB4 作为 RX 时, 则禁止将 PB6 复用成 RX (FUN1);

使用 PB6 作为 RX 时, 则禁止将 PA12 复用成 RX (FUN2), 禁止将 PB4 复用成 RX (FUN1)。

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_DEBUG 配置字需禁止。GFG_DEBUG 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. 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. 在进入休眠函数和退出休眠函数时使用芯片软复位进行切换; (NVIC_SystemReset());
4. 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));
5. 进入休眠前需配置系统时钟周期不小于 2us, 唤醒后可将其恢复, 唤醒时间不可小于 40us(配置 WAKEUPTIME 寄存器); 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。
6. 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。



1. 18 PLL时钟源选择LRC或LOSC的注意事项

当 PLL 输入时钟源选择为内部 LRC 时钟或外部 32KHz 时钟时，如果 PLL_48M_SEL=1，则 PLL 输出时钟频率高于 48MHz，略超过芯片主系统时钟频率范围上限，所以建议此时设置 PLL_48M_SEL=0，或系统时钟后分频设置为 1:2 以上。

1. 19 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。