文档编号：AN_129

上海东软载波微电子有限公司

应用笔记

ES8H296

修订历史

版本	修订日期	修改概要
V1.0	2020-4-9	初版发布

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/

目 录

内容目录

第1章 ES8H296 应用注意

1.1 配置字DEBUG功能

调试时，DBG0EB 或 DBG1EB 位要使能，并且 WDT 关闭、BOR 关闭。

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。

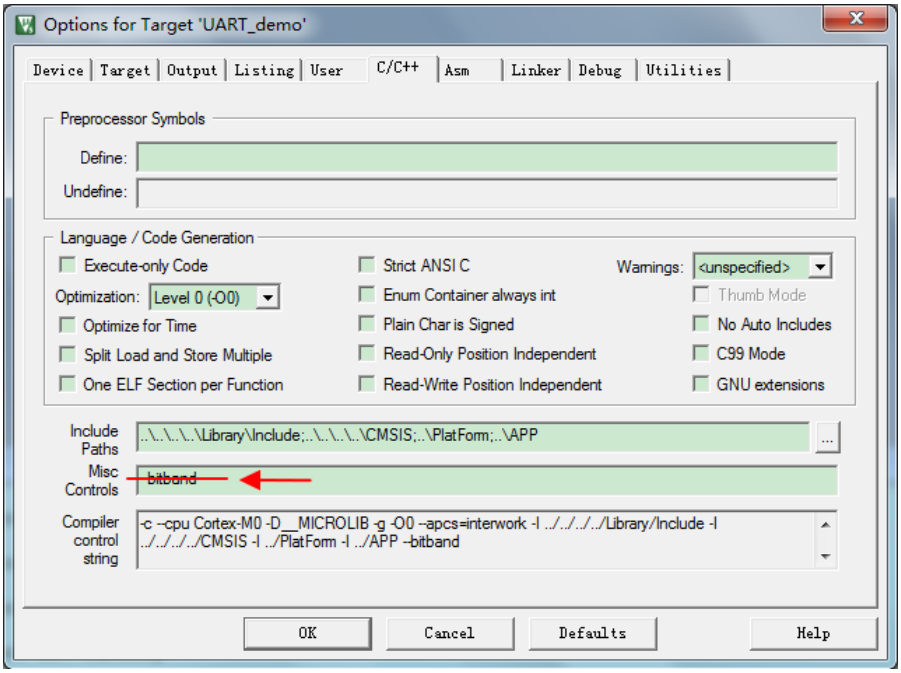
1.2 开发环境

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 SCU_NMIC，SCU_PWRC，SCU_LVDC，SCU_PCLKEN0，SCU_PCLKEN1，SCU_SCLKEN0，SCU_SCLKEN1，SCU_TBLRMEN，SCU_TBLOFFS。

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 GPIO_PAFUN0，GPIO_PAFUN1，GPIO_PAFUN2，GPIO_PAFUN3，GPIO_PBFUN0，GPIO_PBFUN1，GPIO_PBFUN2，GPIO_PAPUEN，GPIO_PBPUEN，GPIO_PAPDEN，GPIO_PBPDEN。

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，对该寄存器写 0x00000000 会使能写保护。

RTC_WP 保护的 RTC 寄存器为 RTC_CON, RTC_CALC, RTC_CALF, RTC_WA, RTC_DA, RTC_SEC, RTC_MIN, RTC_HOUR, RTC_WEEK, RTC_DAY, RTC_MON, RTC_YEAR, RTC_IE。

1.3.4 WDT写保护

对 WDT_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。

WDT_LOCK 保护的寄存器为 WDT_LOAD，WDT_CON，WDT_INTCLR。

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≤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：

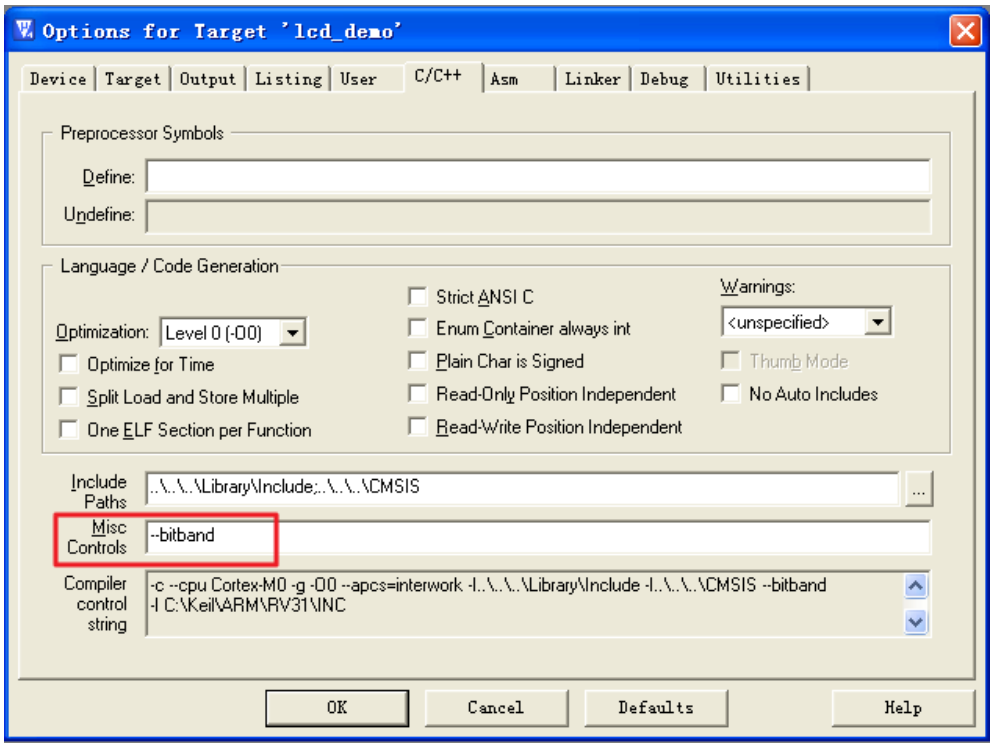
AliasAddress_A_N = 0x2200_0000 + (A – 0x2000_0000) x 32 + N x 4

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. 4. 3 位带注意事项

并不是所有的外设寄存器都支持位带功能，以下寄存器不支持位带功能，请勿对其进行位带操作：GPIO_PADIRS，GPIO_PADIRC，GPIO_PADIRI，GPIO_PAS，GPIO_PAC，GPIO_PAI，GPIO_PBDIRS，GPIO_PBDIRC，GPIO_PBDIRI，GPIO_PBS，GPIO_PBC，GPIO_PBI， ，RTC_WA，RTC_DA，RTC_HOUR。

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;
```

或者进行位域操作（必须对 C 编译器进行“--bitband”设置）：

```
T16N0->IF.MAT0IF = 1;
```

如果进行位域操作(T16N0->IF.MAT0IF = 1)而没有对 C 编译器进行“--bitband”设置，或者用按位“与、或、非”的方式(T16N0->IF.Word |= (uint32_t)0x01)，其最终都是按照“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 GPIO端口数据寄存器

对 GPIO 端口数据寄存器写操作时，实际是写 GPIO 端口输出的数据；读操作时，实际是读取 GPIO 端口的电平状态。如果对 GPIO 端口数据寄存器进行位写操作是按照“读-修改-写”的方式执行的，那么可能会出现端口误操作的现象。例如：PA0 输出高电平(寄存器状态为“1”)，但是如果 PA0 的负载较大，将其电压拖低，这时如果对 PA 进行“读-修改-写”来操作 PA1(GPIO->PA.Word |= (uint32_t)0x02)，那么读到 PA0 实际值为“0”，再将“0”写回到 PA0，那么 PA0 就失去了对外部电路进行高电平驱动的能力，从而导致 GPIO 端口误操作。用户如果需要对 GPIO 端口进行位写操作可以采用对“PAS(置位)”、“PAC(清零)”、“PAI(取反)”寄存器进行直接赋值来实现。

1.8 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。

1.9 LOSC0 和LOSC1 管脚处理

在应用系统中，若不使用外部 LOSC 晶振，则需将芯片配置字 LOSCEN 写 0，HRCCEB 写 1。 ，并将 LOSC0 和 LOSC1 管脚均下拉到地 VSS（推荐下拉电阻约 1K），否则可能会导致芯片功耗异常或降低芯片抗干扰性能。

1. 10 串行总线操作

串行总线 I2C 发送数据时，需等待 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 IDLF 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 TXIDLEIF 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. 11 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.可以使用直接操作寄存器的方法来控制 I2C 的传输。为了使代码的效率更高，在 ES8H296 的函数库中已经将部分判断中断使能和中断标志同时置位的语句封装成宏，如“#define I2C_RBIEandRBIF_IFSET() ((I2C->IE.RBIE)&&(I2C->IF.RBIF))”，程序可以直接调用。

1. 12 管脚复用

每个 IO 管脚都支持 1-4 个复用功能，通过 FUNC 寄存器选择。其中 PA18 由于与芯片内部基准电压测试输出复用，用户禁止使用 PA18，必须将 PA18 悬空，程序将 PA18 配置为 GPIO 输出口，且禁止内部上下拉。

1. 13 内部LDO VR30 和VR36

内部 3.0V LDO VR30 用于芯片内部 ADC 电路模块供电，为了使 ADC 有更好的性能，推荐使用 ADC 时将 VR30_EN 保持为使能状态，并且 VR30 端口外接稳压电容值推荐选用 0.01uF。当 VR30_EN 禁止时，ADC 使用的是芯片供电电源 VDD。

内部 3.6V LDO VR36 用于芯片内部 LCD 电路模块供电。当 VR36_EN 使能时，LCD 模块为内部 LDO 3.6V 供电；当 VR36_EN 禁止时，LCD 模块为芯片供电电源 VDD 供电；用户应该根据所

驱动的 LCD 来确定 VR36_EN 的使能与禁止。

VR30 和 VR36 都仅用于芯片内部电路供电，请勿用于芯片外部电路供电。

1. 14 ADC测量 0~5V信号

当 ADC 需要测量 0~5V 满量程信号，可以将芯片使用 5V 供电，将 VR30_EN 禁止(即 ADC 模块由 VDD 供电)，再将 ADC 正向参考电压选择位配置为外部参考电压 AVREFP，并在 AVREFP 端口提供 5V 参考电压。

1. 15 LCD时钟源选择

由于 LRC 时钟偏差较大，建议选择 LOSC 作为 LCD 时钟源。

1. 16 IAP操作程序

在 Flash IAP 自编程操作时，IAP 操作程序需要放在 RAM 中执行，具体设置有两种方法。

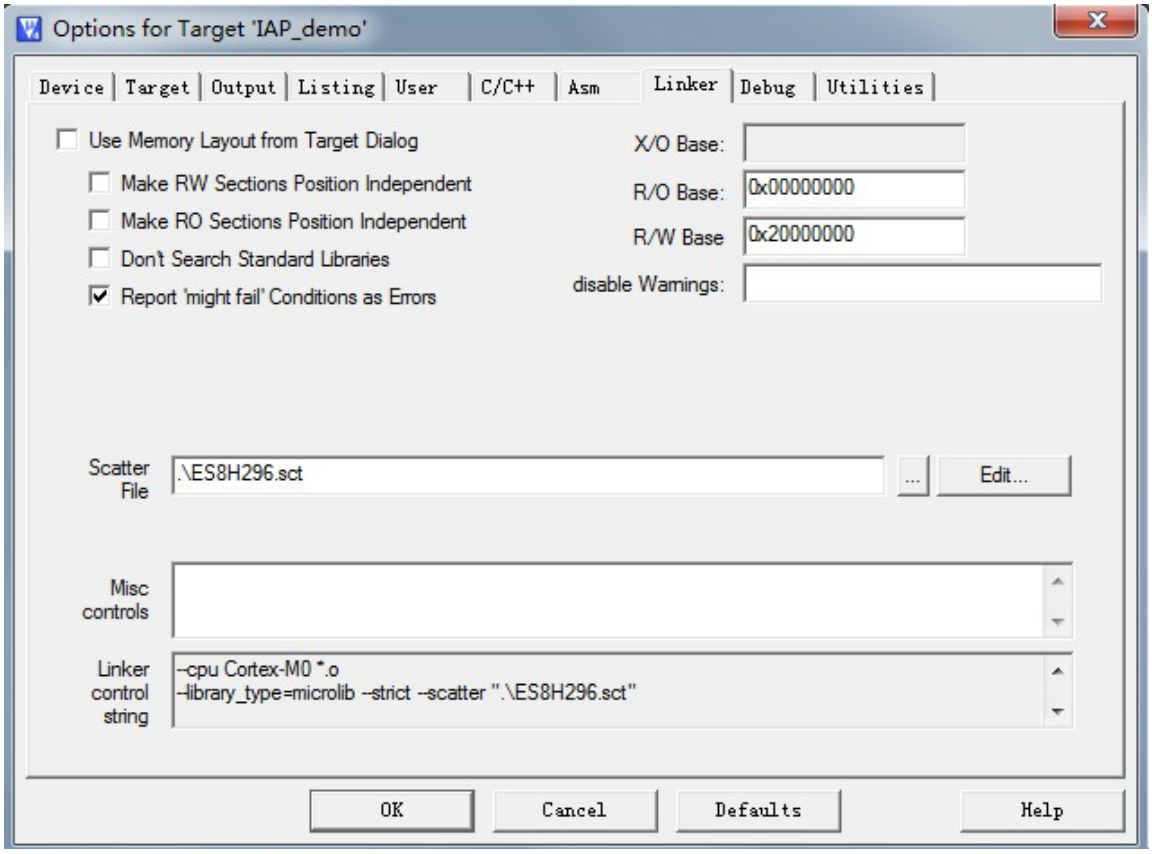
1. 使用 Scatter File 设置

通过编写 Scatter File 文件，实现芯片上电时自动将 IAP 操作程序拷贝到指定的 RAM 区域中，例如：将库函数编译生成的 lib_flashiap.o 内的程序自动复制到 RAM 区域。

Scatter File 文件格式如下所示：

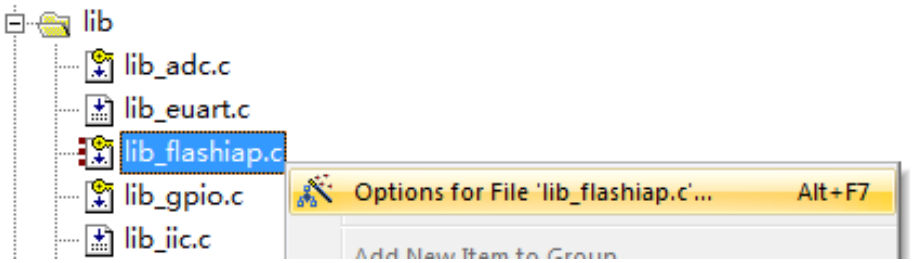
```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x00000000 0x00010000  {    ; load region size_region
  ER_IROM1 0x00000000 0x0010000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x00002000  {  ; RW data
   .ANY (+RW +ZI)
   lib_flashiap.o(+RO)
  }
}
```

在项目的“Options for Target”中，“Use Memory Layout from Targe Dialog”选项不必勾选，并指定好 Scatter File（可参考例程项目文件）。

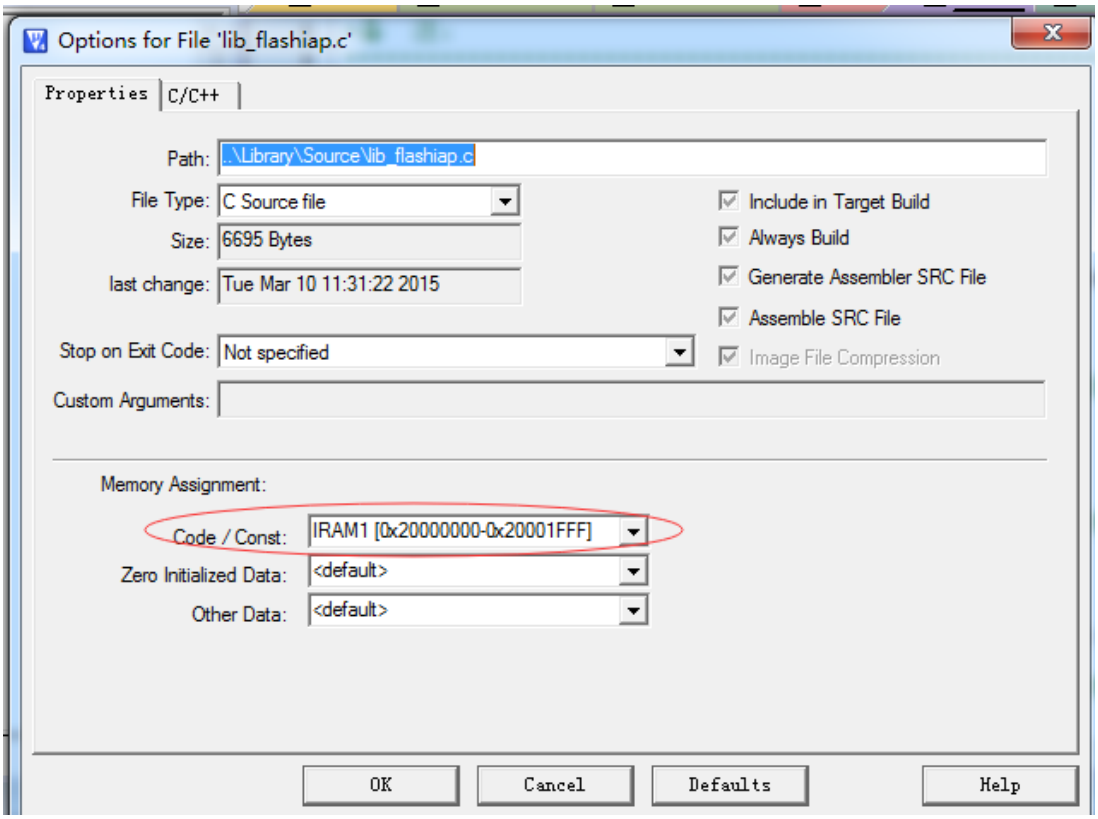


2. 使用对话框设置

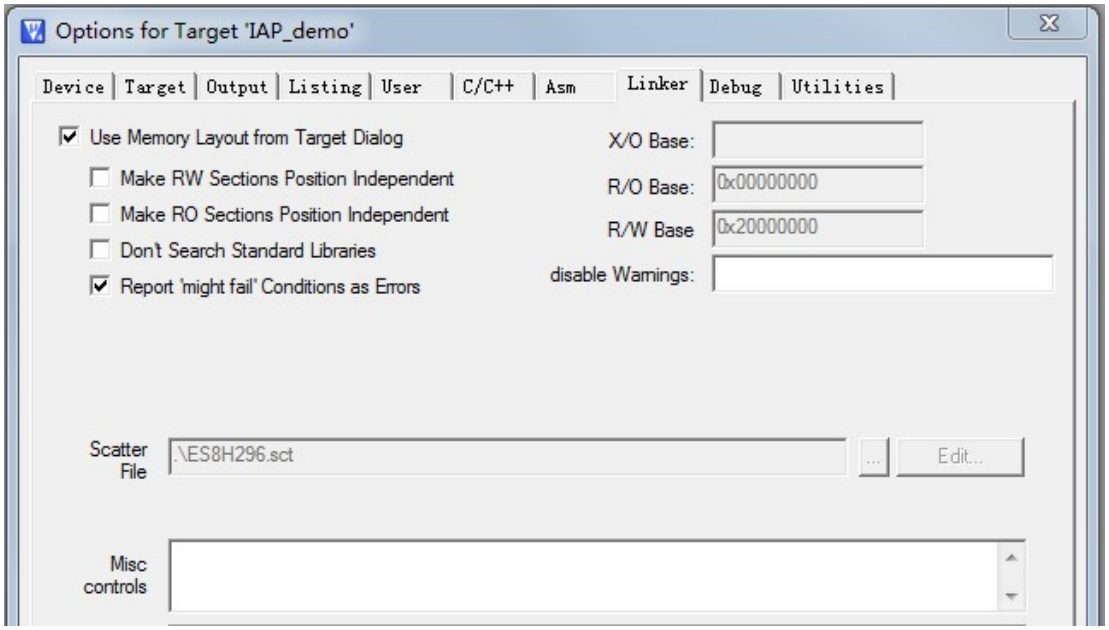
点击项目里选择 lib_flashiap.c 源文件，右键打开“Option for File”。



然后在属性中指定 IRAM 的地址范围。



当使用这种方法时，“Use Memory Layout from Targe Dialog”需要勾选。



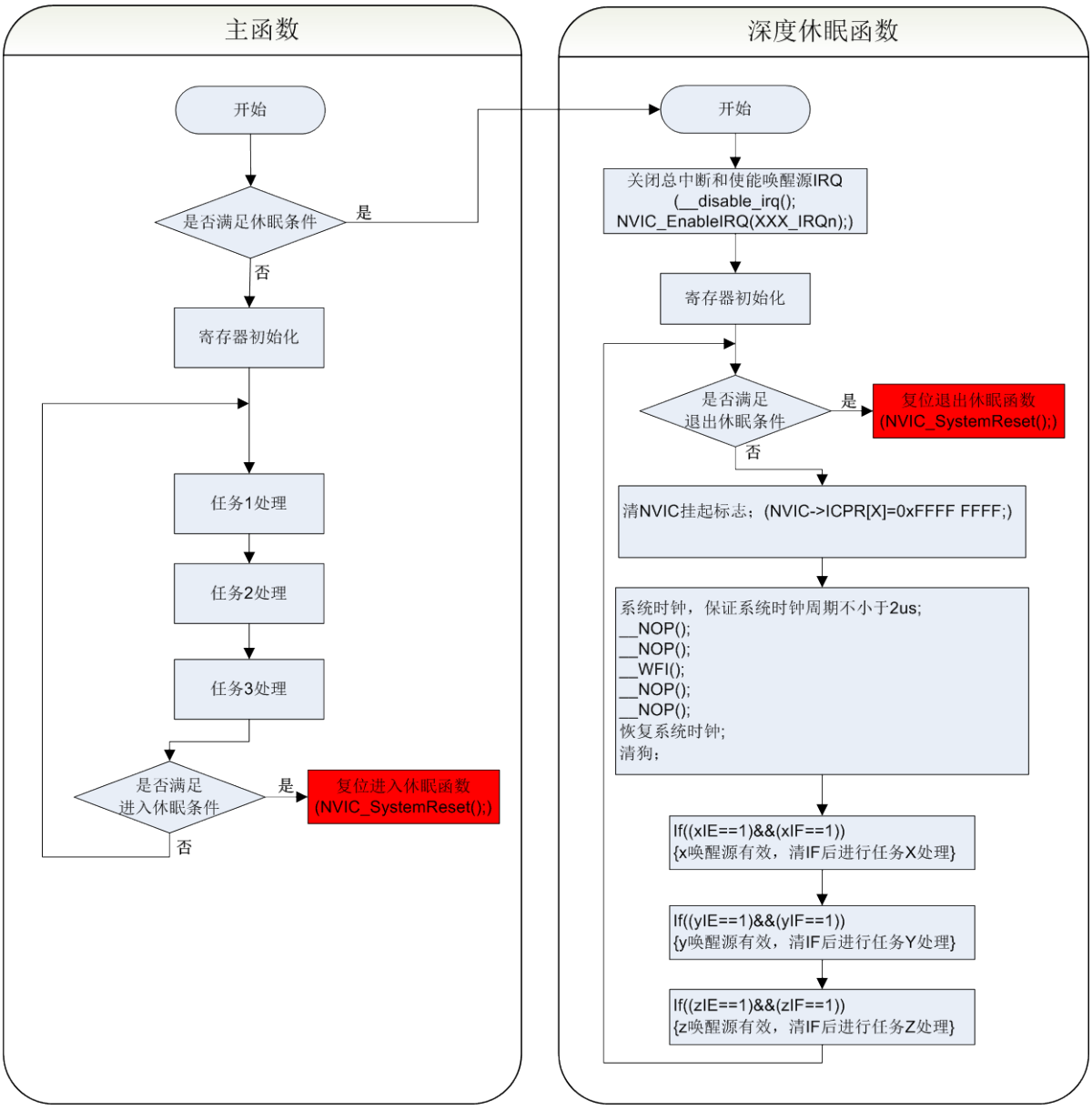
1. 17 低功耗系统程序设计注意事项与推荐结构

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. 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 复位后的时钟选择

复位后应该首先打开 HRC，并选择 HRC 作为系统时钟后关闭 PLL，再切换到自己需要的时钟源。如果需要选择 PLL 倍频时钟作为系统时钟，那么在系统时钟切换到 PLL 之前“PLL 作为系统时钟时失锁处理”(PLL_ULOCK_SET 寄存器)必须保持为 0，当系统时钟成功切换到 PLL 后 PLL_ULOCK_SET 才能选择其他配置。