文档编号：AN_164

上海东软载波微电子有限公司

应用笔记

ES8H04xx

修订历史

版本	修订日期	修改概要
V1.00	2025-06-16	初版

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

目 录

第1章　ES8H04xx应用注意

本文档适用于 ES8H0464、ES8H0484、ES8H0494 系列芯片。

1. 1　配置字DEBUG功能

调试时：CFG_SWD 要使能。

生产正式产品时：CFG_SWD 必须禁止，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

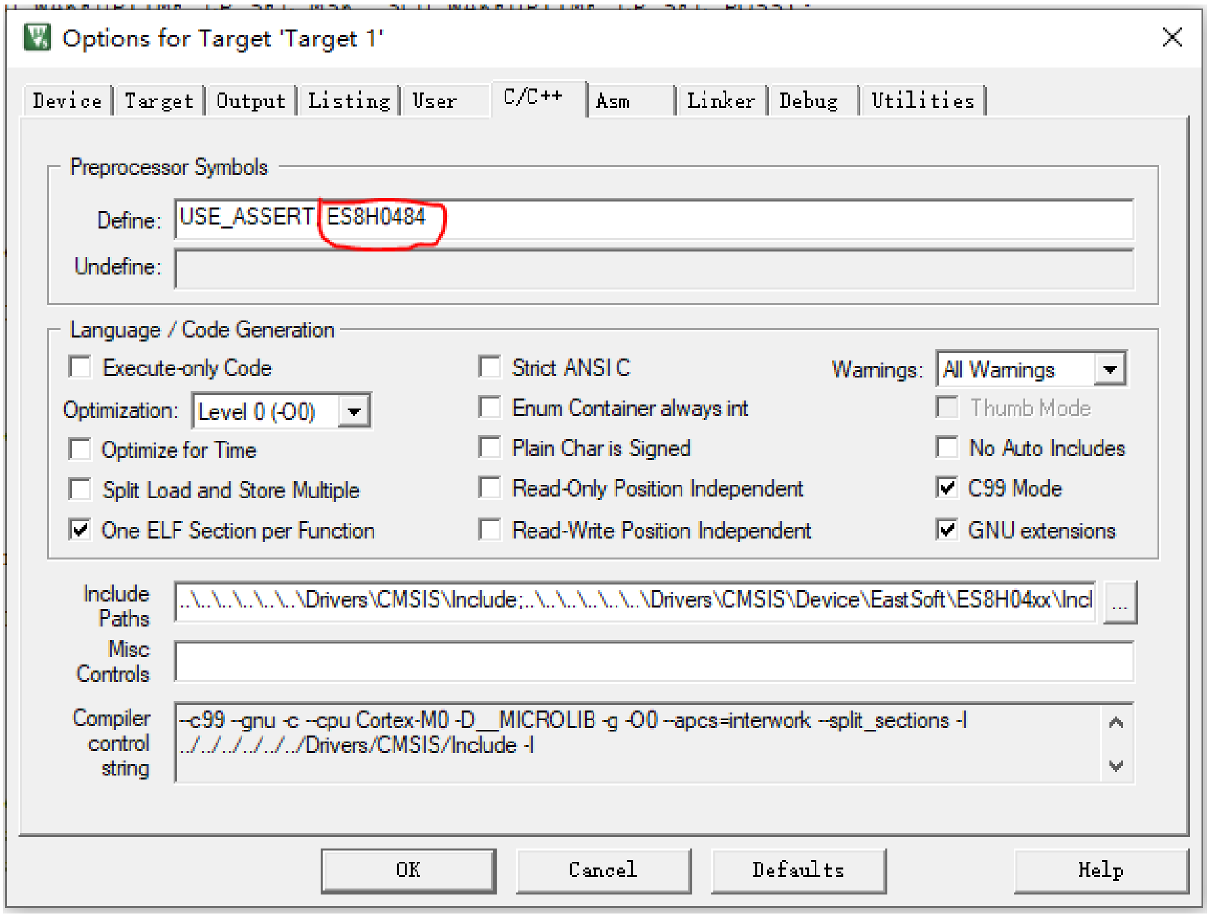
IDE	推荐版本	插件包
Keil5	V5.2x 及以上版本	Keil 5 芯片支持包 Eastsoft.ES32_DFP.1.0.17.pack
IAR	IAR for ARM 8.11.1 及以上版本	IAR 插件

Keil5 使用说明：8P/8H 产品在 Keil5 下开发需要进行如下步骤：

1. 从 keil 官网下载 keil5 的 arm 版本，安装 keil5。
2. 安装 keil5 的芯片支持包，支持包版本大于 1.0.17 以上。

1. 3　区分不同型号芯片

库文件针对不同型号的芯片做了条件编译，需要在工程配置里添加对应芯片的宏定义。例如：Keil5 工程添加 ES8H0484 宏定义。



1. 4 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. 4. 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_ECCSTA，SCU_WAKEUPTIME，SCU_DBGHALT，SCU_FLASHACCR，SCU_SOFTCFG，SCU_LVDCON，SCU_CCM，SCU_PLLLKCON，SCU_HRCBCON，SCU_SCLKEN0，SCU_SCLKEN1，SCU_PCLKEN0，SCU_PCLKEN1，SCU_LPCLK，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. 4. 2 IAP写保护

对 IAP_FLASHKEY 寄存器连续写入 0x8ACE0246 和 0x9BDF1357 可去除写保护，写入其他值或中间插入其他操作将失效。

通过检查 IAP_FLASHKEY.STATUS 是否为 0，判断 Flash 是否处于保护状态。处于保护状态

时，无法进行擦除和编程操作。

库函数提供 IAP_FLASH_Unlock 函数解除写保护，IAP_FLASH_Lock 函数使能写保护。

1. 4. 3 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. 4. 4 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. 5 CRC 写保护

对 CRC_UL 寄存器以字方式写入 0x43524355 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。

库函数提供 CRC_UNLOCK 宏解除写保护，CRC_LOCK 宏使能写保护。

1. 5 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. 5. 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. 5. 2 位带使用方法

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。

1. 6 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 = T16N_IF_MAT0IF_MSK；// T16N_IF_MAT0IF_MSK = 0x0000 0001

禁止进行位操作，如下操作均为错误的写法：

T16N0->IE = 1；

T16N0->IF |= T16N_IF_MAT0IF_MSK；// T16N_IF_MAT0IF_MSK = 0x0000 0001

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. 7 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 提供的标准库并不限定具体超时机制，用户需要根据实际系统需求设计超时机制。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. 8 SCU注意事项

如果没有使用库函数 md_scu_clock_config 用来初始化时钟代码，MCU 开始工作的时候一定要使能寄存器位 ADC_CON1.VBGEN，参考 md_scu_clock_config 接口的第一句代码。

1. 9 LVD注意事项

选择 LVDO 低电平产生中断时（即 SCU_LVDCON.IFS = 0x3），如需同时使能 LVD 滤波，则应按照如下步骤配置：①使能 LVD 滤波；②使能 LVD；③等待 500us；④使能 LVD 中断。

1. 10 IWDT模块

1. 通过配置字使能 IWDT 为硬件看门狗（硬件强制使能），可使 IWDT 脱离软件配置，系统则会更可靠。

2. 当 IWDT 做为软件看门狗使用时，配置字 IWDTEN 配置成“由软件控制”，并通过寄存器对 IWDT 进行初始化。

3. 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，上电默认的溢出周期典型值约 0.5 秒，用户可通过程序修改 IWDT_LOAD 寄存器来调整计数周期。IWDT 模块固定

为使能，IWDT 复位和中断也固定为使能，软件无法关闭，寄存器 IWDT_CON 的 CLKS，RSTEN，IE，EN 位均无效。

4. 可靠的系统在休眠下也应该保持 IWDT 处于工作状态，用户可以使用 WWDT 定时唤醒后对 IWDT 进行喂狗。在深度睡眠模式下，必须将寄存器 SCU_WAKEUPTIME 的 CLKFLT_EN 设置为 0。

1. 11 WWDT模块

WWDT 禁止用户在窗口内喂狗，建议用户在 WWDT 中断服务程序内喂狗。

1. 12 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_IF 寄存器的 TIDIF 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 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，以避免芯片不停地进入发送缓冲空中断。

1. 13 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 SCKOD，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 I2C_TBW 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. 14 SPI主机编程注意项

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. 15 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 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. 16 PWM输出

1. 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/T32N 定时器来实现。
2. T16N 支持 PWM 模式 MAT/PREMAT/TOP 缓冲更新方式选择，有“即时更新”和“当前 PWM 周期结束后才更新”两种方式，库函数 md_t16n_output_struct_init 默认选择后者。
3. T16N 通过配置两个 MAT 值和 TOP 值，可输出中心对齐的 PWM，例如：MAT0 = 800，MAT1 = 1200，TOP = 2000，MAT0 匹配后的端口选择置 1，MAT1 匹配后的端口选择清 0，输出正极性，则可输出占空比 20%的中心对齐 PWM，通过调节 MAT0 和 MAT1 值改变占空比。但用此方法输出占空比 100%或 0%的 PWM 时，会在周期起始处产生毛刺，需选择合适的 MAT 匹配电平，方可输出干净的信号，例如：MAT0 和 MAT1 匹配后的端口皆选择置 1，则输出占空比 100%的 PWM。

1. 17 GPIO端口输出电平位操作

GPIO 端口位操作寄存器 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. 18 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低电平并悬空，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. 19 GPIO端口用作输出功能时的注意事项

外部复位与 GPIO 复用同一个管脚，默认为外部复位 MRSTN 功能，内部集成了约 55K 欧姆的上拉电阻，用户可通过芯片配置字 CFG_WORD0 的 CFG_MRSTN 位将其设为 GPIO 功能，对该管脚按与其他 GPIO 管脚一样的方式使用即可。

在使用 GPIO 端口输出驱动 LED，三极管，继电器等外围元器件时，建议在元器件的输入端添加适当的下拉电阻（例如 10K 欧姆），避免在芯片开始工作前，元器件输入端因无有效驱动信号而出现误触发。

1. 20 ADC模块应用注意事项

1. ADC 正常工作时，必须开启 VBGEN，否则会导致 ADC 工作异常。
2. 因每次 A/D 转换使能位 EN 重新使能后，均需要执行 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 A/D 转换使能控制信号，保持为 1，只在进入深睡眠模式前，关闭 ADC。
3. 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. 21 系统时钟配置注意事项

1. 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。
2. PLL 输入时钟源选择 HRC 时，HRC 频率只能选择 16MHz，选择其他频率无效。
3. 使用外部晶振时，需软件设置 XTAL 对应的 GPIO 为模拟口，即数字输入输出均关闭，然后再使能外部晶振。
4. 当 HRC 时钟频率为 16MHz 时，需将寄存器 SCU_HRCBCON 的时钟偏置系数位 HRCBSET<2:0>设置为 111，当 HRC 为 48MHz 时，该寄存器位需保持为默认值 000。

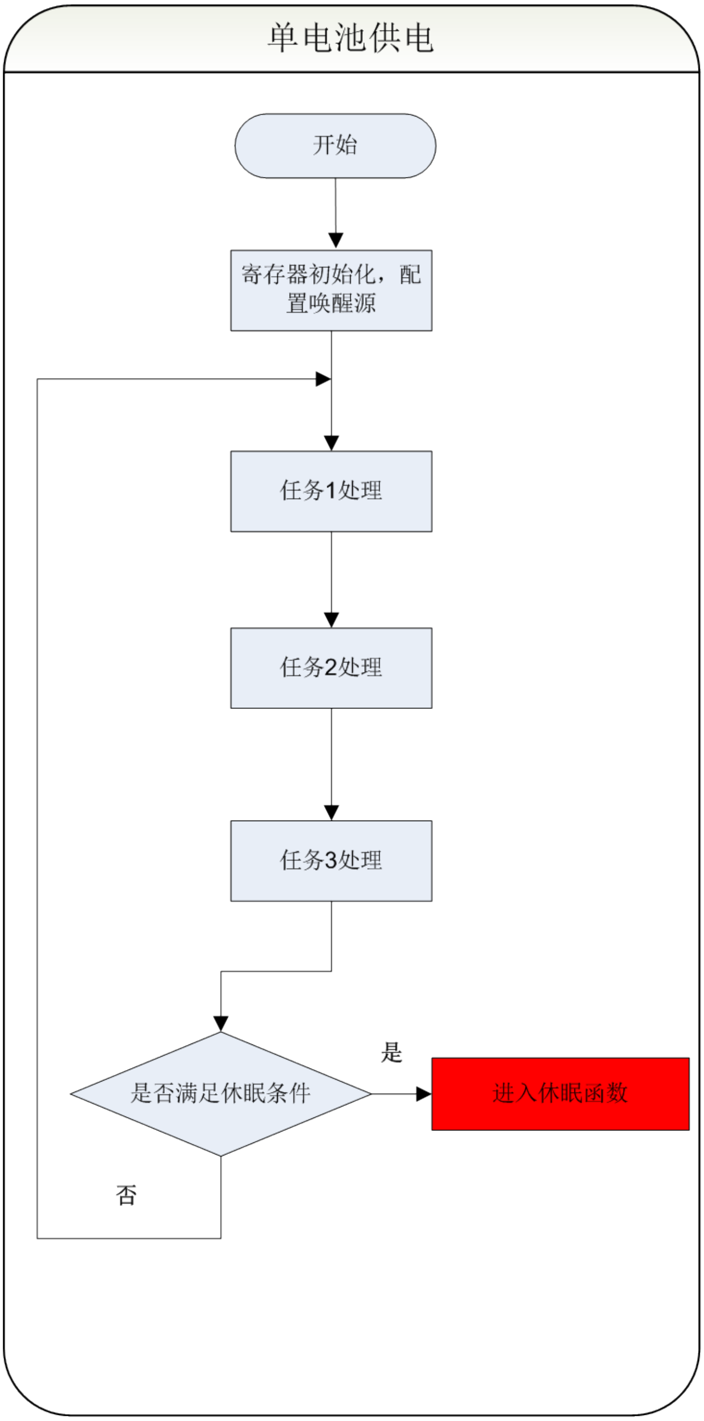
1. 22 低功耗系统程序设计注意事项与推荐结构

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_SWD 配置字需禁止。GFG_SWD 配置字禁止后，SWD 的两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. 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 做为定时唤醒源，WWDT 唤醒后对 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。
5. 芯片执行 WFI 进入低功耗模式前：
   如果 SysTick 为使能状态，需要临时关闭 SysTick 计数器使能位（ENABLE@SysTick->CTRL）；
   如果有非唤醒源的中断为使能状态，需要临时禁止对应的 IRQ 中断（NVIC_DisableIRQ(XXX_IRQn)）。

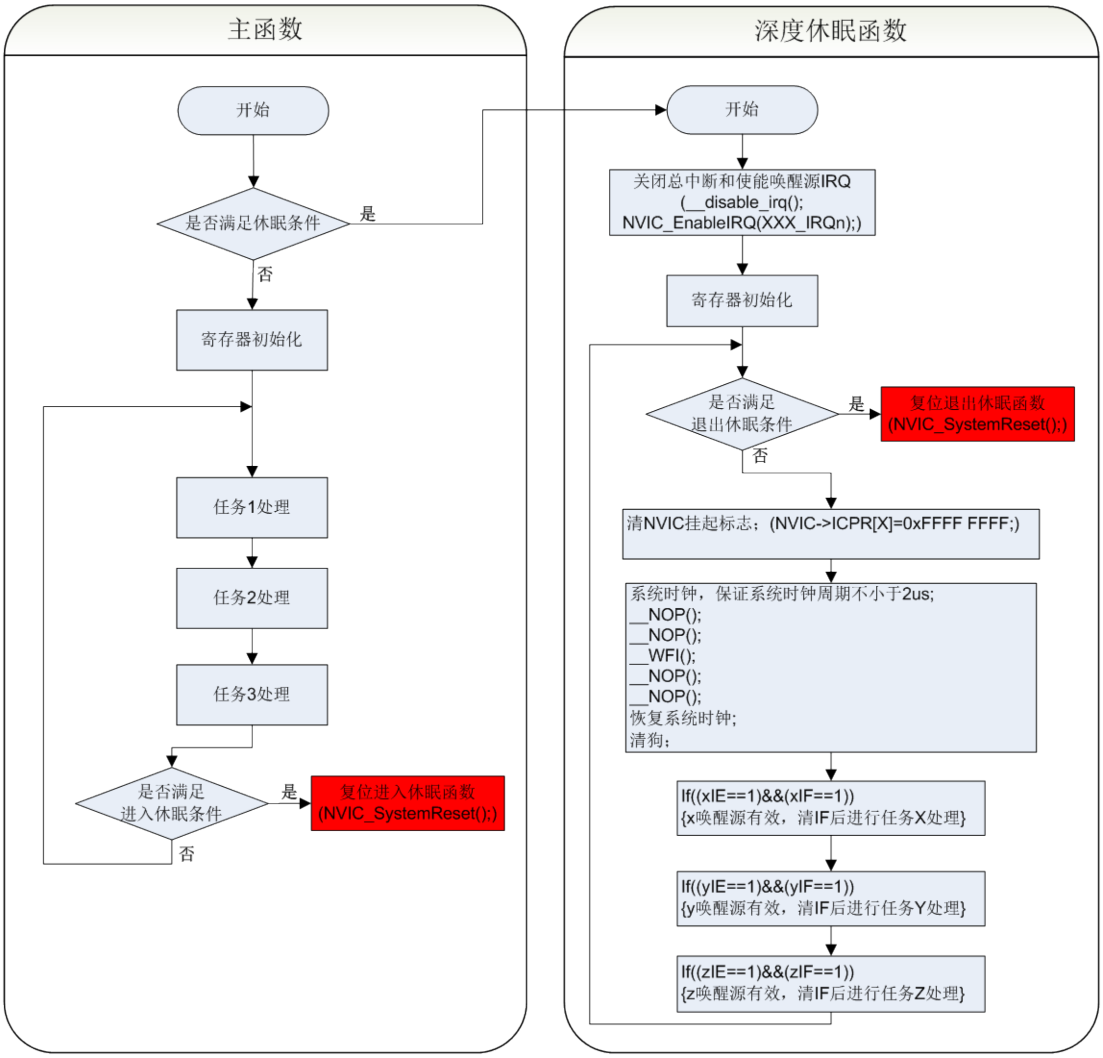
1. 22. 1　单电池供电系统低功耗设计

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. 22. 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);) 。

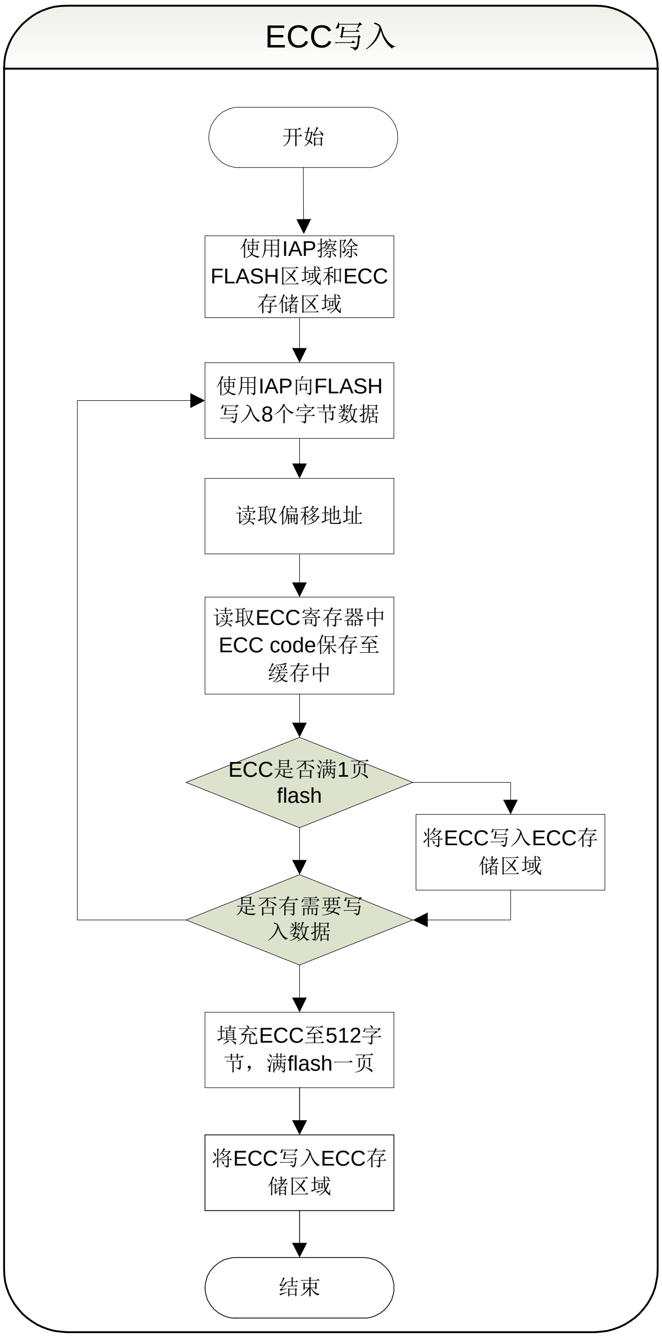


1. 23 IAP防误擦

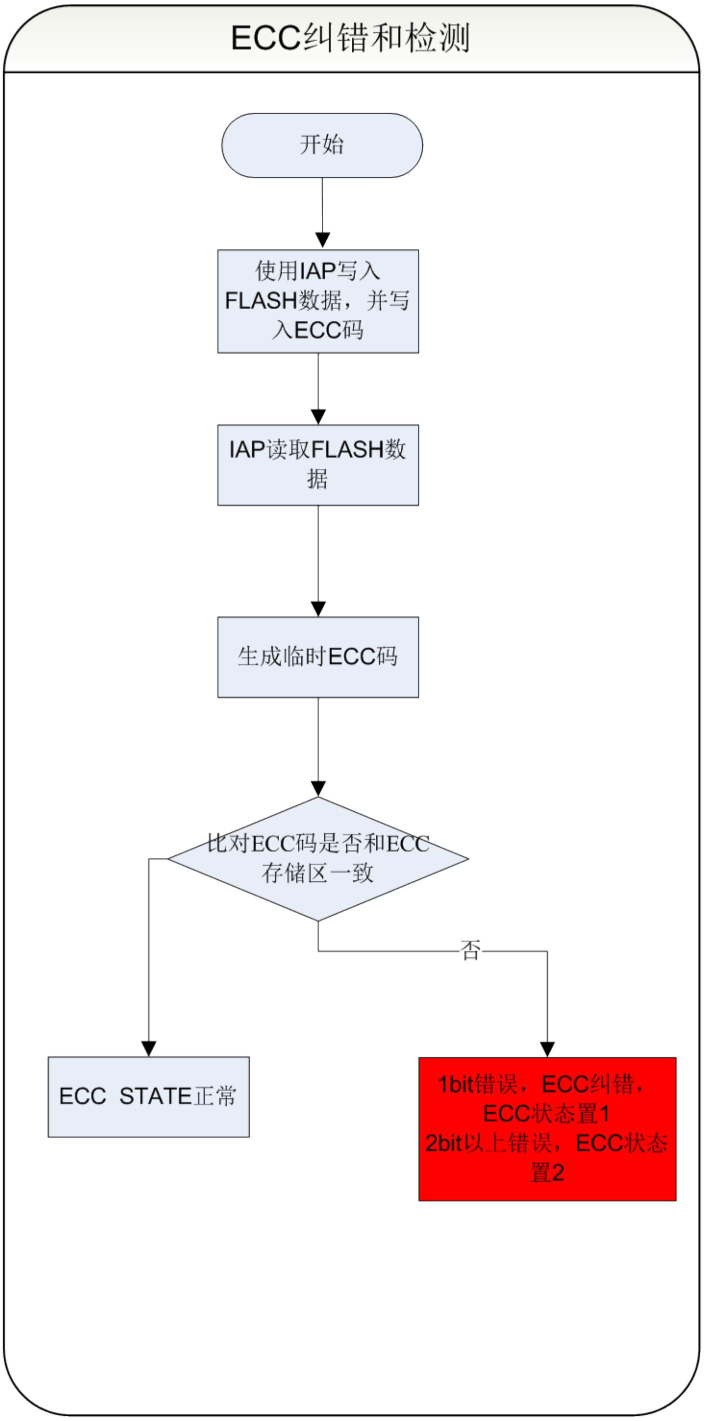
在 EMC 干扰较强的系统，为防止 PC 跑飞到 IAP 自编程固化模块并执行，可以禁止 IAP 模块时钟。这样即使 PC 跑飞到 IAP 自编程固化模块并执行，也不会对 Flash 的内容产生任何影响。可以通过 SCU_PCLKEN0 的 IAP_EN 位来使能和禁止 IAP 模块的时钟。需要注意的是 IAP_EN 的写操作受到 SCU_PROT 的保护。

1. 24 ECC

使用 ECC 功能时，必须要在配置字中使能 ECC 的开关。ECC 启动后，按 8 比 1 字节数写入 ECC 码到 ECC 存储地址，每 8 个字节数据产生 1 个字节的 ECC 码，通过 IAP 按字写入，即 32 个字节产生 1 个字的 ECC 码，才能通过 IAP 写入到 flash，所以可以考虑使用 4K 字节对齐方式，每 4K 字节产生 1 页的 ECC 码。在 IAP 写入一个 8 字节后需要去对应 flash 偏移地址中读取写入的数据，ECC 寄存器才会产生对应 ECC 码，即写 8 个字节，读一次偏移地址，再读 SCU 中对应 ECC 寄存器的 code。

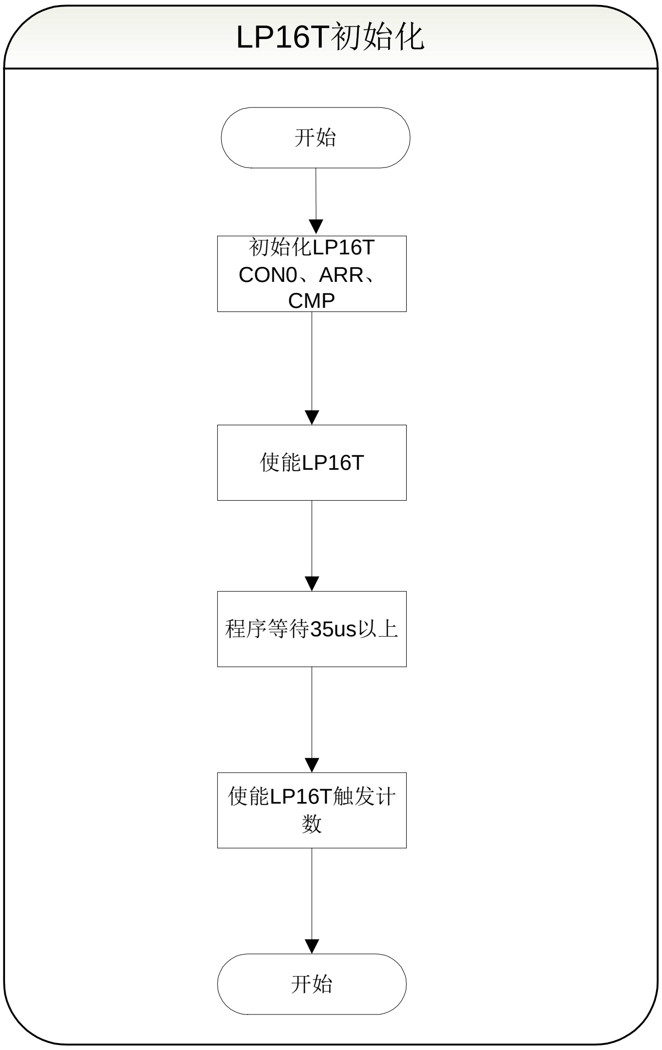


当 ECC 码和数据都写入 flash 后，ECC 纠错和检测错误功能可以验证，如在 0x0000 5000 写入数据，在 ECC 存储区域 0x0003 8A00 写完数据后，如果重新对 0x0000 5000 地址 IAP 写入数据，但是没对 0x0003 8A00 更改对应 ECC 码，在这个条件下，去读取 0x0000 5000 数据，这个时候就会触发 ECC 检测错误和纠错机制，因为读取 0x0000 5000 数据时，芯片对 0x0000 5000 地址生成临时 ECC 码并和 ECC 存储区域 0x0003 8A00 的 ECC 码比对，如果比对不正确，这会发生 ECC 纠错和 ECC 错误状态置位。



1. 25 LP16T

ES8H04xx 搭载了 LP16T，提供低功耗定时器的使用，在深度休眠下可以利用 LP16T 来唤醒 MCU，初始化 LP16T，先通过 SCU 选择 LP16T 的时钟，然后配置相关 LP16T 的参数，重载值、比较值、分频系数和工作模式。如果 LP16T 选择了 LRC 时钟，但是 MCU 主时钟是 HRC 或 XTAL，启动 LP16T 计时需要等待 1 个 LRC 的计数周期，约等于 30us，软件等待大于 35us 以上即可。



1. 26 深度休眠注意事项

使用 md_scu_deep_sleep_enter()让 MCU 进入深度休眠时，需要先关闭 ADC_CON1_VBGEN(函数接口：md_adc_disable_vbg)，从深度休眠唤醒时，需要打开 ADC_CON1_VBGEN(函数接口：md_adc_enable_vbg)。如不使用接口，需要手动实现 VBGEN 的开关。

第2章　外设初始化步骤

2. 1　GPIO

(1) 选择端口信号类型为数字或模拟；
(2) 设置端口复用功能；
(3) 选择端口方向为输入或输出；
(4) 选择是否使能端口弱上拉或弱下拉；
(5) 选择端口为开漏输出或推挽输出；
(6) 选择端口驱动电流强弱；
(7) 选择端口电平类型为 CMOS 或 TTL；
(8) 选择是否使能端口 20ns 滤波功能；
(9) 如需使用 PINT 或 KINT 中断，选择 PINT 或 KINT 输入通道，配置触发中断的信号类型，并配置 NVIC 中断通道、中断优先级，使能所需的 PINT 或 KINT 中断；
(10) 若使能 PINT 或 KINT 中断，且相应标志位置 1，则产生中断，需软件清除标志位。

2. 2　Timer

输入捕捉模式：
(1) 初始化系统时钟，使能需要使用的外设时钟；
(2) 配置对应的 GPIO 为输入捕捉功能，并配置端口为输入方向；
(3) 初始化 Timer 基本配置，使用捕捉模式；
(4) 选择 CNT 匹配 MAT 后的工作模式；
(5) 设置预分频比例，初始化捕捉配置；
(6) 配置 NVIC 中断通道、优先级并使能，使能捕捉中断；
(7) 启动 Timer；
(8) 中断响应后，软件清除标志位，保存 MAT 值。

定时/计数模式：
(1) 初始化系统时钟，使能需要使用的外设时钟，并配置端口为输入方向；
(2) 配置对应的 GPIO 为定时/计数功能；
(3) 初始化 Timer 基本配置，使用定时/计数模式，时钟源选内部时钟时，为定时模式；时钟源选外部时钟时，为计数模式；
(4) 设置预分频比例、计数值、匹配值等；
(5) 选择 CNT 匹配 MAT 后的工作模式；
(6) 配置 NVIC 中断通道、优先级并使能，选择是否使能匹配中断；
(7) 启动 Timer；
(8) 若使能匹配中断，且相应标志位置 1，则产生中断，需软件清除标志位。

输出调制模式：
(1) 初始化系统时钟，使能需要使用的外设时钟，并配置端口为输出方向；
(2) 配置对应的 GPIO 为输出调制输出功能；
(3) 初始化 Timer 基本配置，使用调制模式；
(4) 选择 MATx 匹配后的输出端口工作模式；

(5) 初始化 PWM 输出配置；
(6) 根据需要输出的 PWM 周期、占空比，设置预分频比例、计数值、匹配值、峰值等；
(7) 配置 NVIC 中断通道、优先级并使能；
(8) 选择 CNT 匹配 MAT 后的工作模式；
(9) 选择是否使能匹配中断；
(10) 启动 Timer；
(11) 若使能匹配中断，且相应标志位置 1，则产生中断，需软件清除标志位。

2. 3 ADC

(1) 配置 ADC 采样管脚为模拟复用功能；
(2) 初始化 ADC 外设寄存器；
(3) 设置 ADC 中断使能，以及对应的 NVIC 中断向量优先级并使能；
(4) 选择 ADC 采样通道；
(5) 启动 ADC 开始转换。

2. 4 UART

(1) 配置 UART TX 和 RX 管脚为对应的数字外设复用功能，并配置正确的端口输入/输出方向；
(2) 初始化 UART 外设寄存器；
(3) 配置 UART 接收缓冲器满中断模式和发送缓冲器空中断模式；
(4) 打开接收中断（RBIE@UART_IE），并设置 UART 对应的 NVIC 中断向量优先级并使能；
(5) 配置发送使能（TXEN@UART_CON）和接收使能（RXEN@UART_CON）
(6) 发送中断（TBIE@UART_IE）须在发送数据时再打开，打开后会立即触发发送缓冲器空中断（TBIF@UART_IF）。向 UART_TBW 写完最后一个要发送的数据后须关闭发送中断（TBIE@UART_IE）。

2. 5 SPI

主机模式：

(1) 配置 NSS/SCK/MISO/MOSI 管脚为对应的数字外设复用功能，并配置正确的端口输入/输出方向；
(2) 初始化 SPI 外设寄存器，选择主机模式，使能 SPI 接收，选择接收缓冲器满和发送缓冲器空中断模式；
(3) 配置 NVIC 中断通道、中断优先级，并使能；
(4) 使能 SPI；
(5) 清空发送/发送缓冲器；
(6) 接收数据：使能接收缓冲器满中断，拉低片选，空闲状态时向发送缓冲写入同步数据，接收缓冲同时接收到数据，响应接收缓冲器满中断后读取接收缓冲里的数据，直到接收到最后一个数据后，关闭接收缓冲器满中断，拉高片选；
(7) 发送数据：使能发送缓冲器空中断，拉低片选，响应发送缓冲器空中断后向发送缓冲写入数据，直到写完最后一个数据后，关闭发送缓冲器空中断，需等到发送空闲状态，拉高片选。

从机模式：

(1) 配置 NSS/SCK/MISO/MOSI 管脚为对应的数字外设复用功能，并配置正确的端口输入/输出方向；
(2) 初始化 SPI 外设寄存器，选择从机模式，选择接收缓冲器满和发送缓冲器空中断模式；
(3) 配置 NVIC 中断通道、中断优先级并使能，使能发送缓冲器空中断和接收缓冲器满中断；
(4) 使能 SPI 接收，使能 SPI；
(5) 等待主机写读，主机写数据，从机响应接收缓冲器满中断，读取接收缓冲里的数据；主机读数据，从机响应发送缓冲器空中断，往发送缓冲写数据。

2. 6 I2C

主机模式：
(1) 配置 SCL/SDA 管脚为对应的数字外设复用功能，并配置正确的端口输入/输出方向；
(2) 初始化 I2C 外设寄存器，选择主机模式，选择接收缓冲器满和发送缓冲器空中断模式，选择接收模式（何时发送 ACK 或 NACK），使能起始位中断和接收缓冲满中断，配置 NVIC 中断通道、中断优先级并使能，使能 I2C；
(3) 接收数据：设置从机地址，读写方向，触发起始位，响应起始位中断后，判断是否为读状态，是则触发数据接收，响应接收缓冲器满中断，读取接收缓冲里的数据，直到接收到最后一个数据后，发送 NACK，触发停止位；
(4) 发送数据：配置从机地址，读写方向，触发起始位，响应起始位中断后，判断是否为写状态，是则向发送缓冲写入首个数据，并使能发送空闲中断，响应发送空闲中断后，继续向发送缓冲写数据，直到写入最后一个数据或 NACK 标志位置起后，触发停止位。

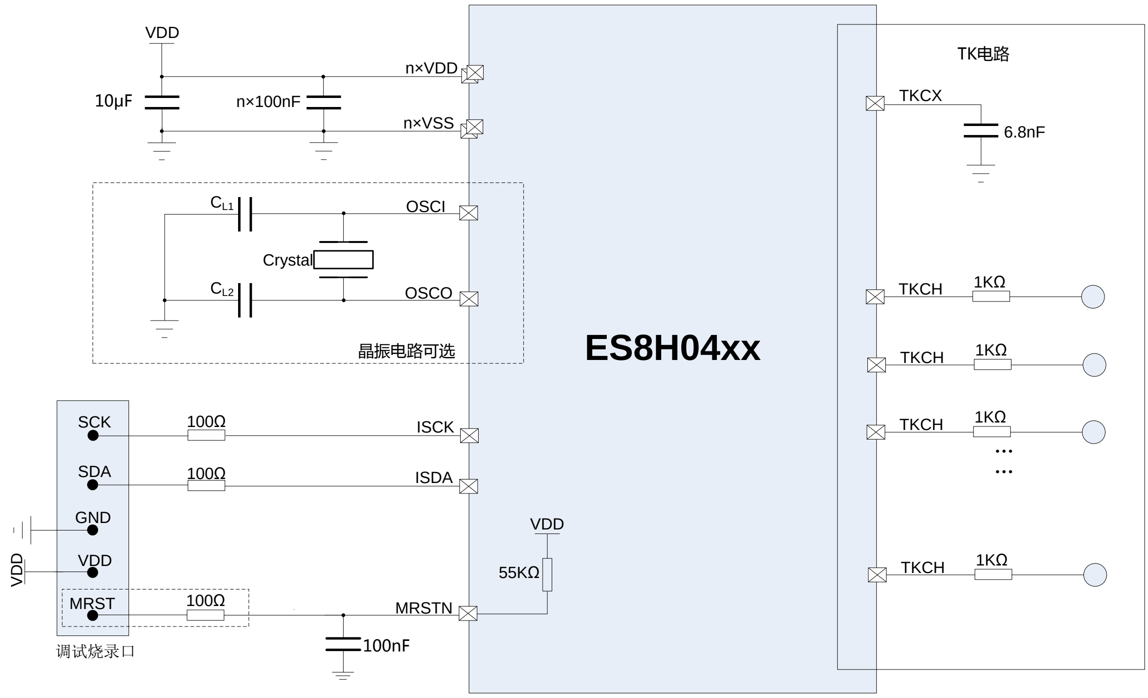
从机模式：
(1) 配置 SCL/SDA 管脚为对应的数字外设复用功能，并配置正确的端口输入/输出方向；
(2) 初始化 I2C 外设寄存器，选择从机模式，选择接收缓冲器满和发送缓冲器空中断模式，使能时钟线自动下拉等待请求，设置从机地址，使能起始位中断和停止位中断，配置 NVIC 中断通道、中断优先级并使能，使能 I2C；
(3) 接收到起始位并且地址匹配后，响应起始位中断，判断写读方向，方向为读则使能发送空闲中断，响应中断后往发送缓冲写数据，方向为写则使能接收缓冲器满中断，响应中断后读取接收缓冲里的数据；
(4) 接收到停止位后，响应停止位中断，关闭发送空闲中断和接收缓冲器满中断。

2. 7 LED

(1) 设置 COM、SEG 对应管脚的复用功能；
(2) 使用 md_led_init_t 设置 LED 的时钟分频系数，COM 的极性、SEG 的极性、死区时间、COM 口的复用数量；
(3) 使能需要显示的 LED 的 COM 个数、SEG 个数，打开 COM 使能、SEG 使能。如需要对 SEG 设置 PWM，设置对应 Mx 寄存器中对应 seg 位，再通过 md_led_set_com_seg_duty 调整 seg 的 PWM 占空比；
(4) 若需要使用帧中断模式来控制帧的刷新：打开帧中断即可(md_led_enable_it_frameie)，产生更新事件，在下一个帧周期完成像素数据更新。

第3章 硬件开发注意事项

3. 1 ES8H04xx最小系统电路



注 1：若需支持 4 线 ISP 使用，需将用户配置字 CFG_PWRTEB 设置为 140ms 复位延时使能，此时 4 线 ISP 接口配置包括电源线 VDD、地线 VSS、时钟线 ISCK 和数据线 ISDA（即省去 MRSTN 复位线）。

注 2：芯片使用 HRC 作为内部高速时钟时，不需要外部晶振电路。支持 2 种 HRC 时钟频率，分别是 16 MHz 和 48 MHz 。

注 3：C_{L1} 和 C_{L2} 为晶振匹配电容，根据所使用的晶振，电容参考取值范围为 10~20pF，建议 1~20MHz 晶振匹配 15pf 电容，32.768KHz 晶振匹配 12pf 电容。为达到理想的晶振起振和稳定工作状态，对 1~8MHz 晶振参数选型时，推荐 ESR≤200Ω，C_L≤16pF（晶振的负载电容参数≤16pF）；对 9~20MHz 晶振参数选型时，推荐 ESR≤50Ω，C_L≤16pF（晶振的负载电容参数≤16pF）；对 32KHz 晶振参数选型时，推荐 ESR≤40KΩ。

注 4：具体的 TK 电路 PCB 设计注意事项请查看《ANxxx_ES8H04xx_TK 设计指南》

3. 2 Cx电容PCB设计注意事项

Cx 电容布线需遵循靠近芯片、减少干扰、合理接地等原则，以确保触摸功能的稳定性和灵敏度。具体规则如下：

(1) 尽量靠近芯片引脚：Cx 电容应尽量靠近芯片的相关管脚，推荐距离小于 10mm，这样可以减少走线长度，降低信号干扰和寄生电容的影响，保障触控电路正常工作；

(2) 电源与地线处理：尽量加宽电源、地线宽度，且地线宽度最好大于电源线宽度。电源和地线应先经过电容滤波后再接入芯片管脚，增强滤波效果；

(3) 避免信号干扰：Cx 电容相关走线应避免与通讯信号线相邻、平行或交叉，以免通讯脉冲信号对检测数据造成干扰。若无法避免交叉，需垂直交叉且尽量只交叉一次；

(4) 控制走线长度和宽度：触摸芯片引脚到 Cx 电容的走线应尽量短而细，以减少信号衰减和外界干扰；

(5) 合理接地：芯片 VDD 端口连接的消抖电容端的 GND、Cx 电容的 GND 和芯片的 GND，这三个 GND 必须尽可能靠近，且连接到总 GND 的走线应尽可能短，以提高 EFT 性能和 CS 性能；

(6) 注意走线间距：Cx 电容相关走线与其他触摸走线或非触摸走线之间，应保持足够的间距，一般建议在 2mm 以上，以减小寄生电容。

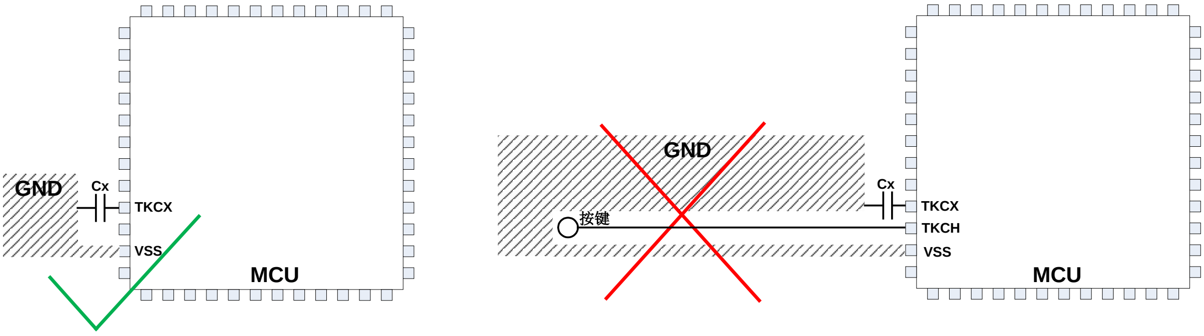


图 1

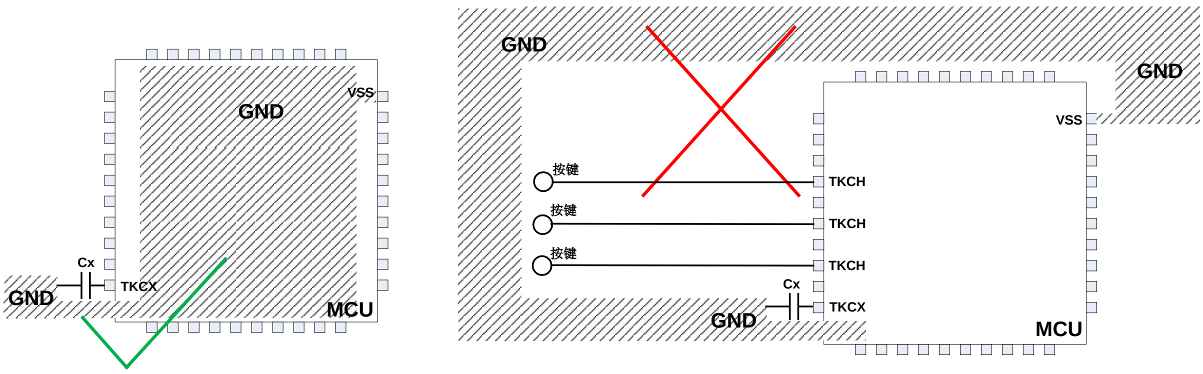


图 2

注 1：如图 1，Cx 电容与芯片 VSS 管脚相邻，单面板应用中禁止走线从 VSS 和 Cx 电容之间穿过；

注 2：如图 2，Cx 电容与芯片 VSS 管脚对角线排布，单面板应用中 VSS 需直接从芯片底部与 Cx 电容接地焊盘相连；

注 3：Cx 电容需使用 X7R 或 NP0 的低温漂材质。

