文档编号：AN2068

上海东软载波微电子有限公司

软硬件设计指南

ES32H040x

修订历史

版本	修订日期	修改概要
V1.0	2025-08-01	初版发布
V1.1	2026-02-11	1、增加串口应用相关注意点 2、增加 IIC 应用相关注意点 3、增加 ADC 应用相关注意点 4、增加 LCD 应用相关注意点 5、增加烧录仿真相关注意点

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/

上海东软载波微电子有限公司

目 录

内容目录

第1章 软件设计指南

1.1 概述

1.1.1 库函数选择

ES32 系列芯片提供 2 种类型库函数 ALD 和 MD：

ALD：提供较为完善的封装，提供更为人性化的 API，适合大部分用户；

MD：基本上只提供寄存器位域级别的“读”、“写”接口，适合对芯片底层较为熟悉的用户。

如果用户对速度不是要求非常严格，一般情况下推荐用户使用 ALD 库。可以减少用户学习时间，增加代码可移植性，最终缩短用户产品的开发周期。

1.1.2 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) 。库函数中均提供相应宏定义进行解除保护和使能保护。

1.1.2.1 系统写保护

系统控制寄存器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 SYSCFG_PROT。对 SYSCFG_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。

可通过读 SYSCFG_PROT 寄存器确认写保护状态,读出值为 0x1,表示当前处于写保护状态;读出值为 0x0 表示当前处于写保护解除状态。

SYSCFG_PROT 保护的寄存器为除 SYSCFG_PROT 寄存器外的 SYSCFG、PMU、CMU、RMU 模块所有寄存器。

1.1.2.2 RTC 写保护

对 RTC_WPR 寄存器以字方式写入 0x55AAAA55 会解除写保护,写入其他值使能写保护。

可通过读 RTC_WPR 寄存器确认 RTC 模块是否处于写保护状态,读出值为 0x1,表示当前处于写保护状态;读出值为 0x0 表示 RTC 模块处于写保护解除状态。

该寄存器保护除自身外的 RTC 所有寄存器。

1.1.2.3 IWDT 写保护

对 IWDT_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护,写入其他值使能写保护。

可通过读 IWDT_LOCK 寄存器确认 IWDT 模块是否处于写保护状态,读出值为 0x1,表示当前处于写保护状态;读出值为 0x0 表示 IWDT 模块处于写保护解除状态。

该寄存器保护除自身外的 IWDT 所有寄存器。

执行 CLWDT 至少需要间隔 3 个时钟才能生效,在执行 CLWDT 后需要延时 128uS 后,再操作

WDT 或休眠指令。

1.1.2.4 WWDT 写保护

对 WWDT_LOCK 寄存器以字方式写入 0x1ACCE551 会解除写保护,写入其他值使能写保护。

可通过读 WWDT_LOCK 寄存器确认 WWDT 模块是否处于写保护状态,读出值为 0x1,表示当前处于写保护状态;读出值为 0x0 表示 WWDT 模块处于写保护解除状态。

该寄存器保护除自身外的 WWDT 所有寄存器。

1.1.3 写 1 清零寄存器

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。对该类寄存器操作需要以字方式进行写。

例:清除 DMA 模块通道 0 的中断标志:

正确写法:DMAx->ICFR = 1;

错误写法:DMAx->ICFR |= 1;

1.1.4 位带操作

位带扩展区把每个 bit 扩展为一个 32-bits 的字,通过访问这些字可达到访问原始 bit 的目的;某个 bit 所在字的地址为 A,位序号为 N(0≤N≤31) ,则该 bit 位带扩展后的地址为:

SRAM：AliasAddr=0x22000000+(A-0x20000000)x32+Nx4

外设：AliasAddr=0x42000000+(A-0x40000000)x32+Nx4

库函数中提供位带操作 API：

RAM 位带：void BITBAND_SRAM(uint32_t *addr, uint32_t bit, uint32_t val);

外设位带：void BITBAND_PER(volatile uint32_t *addr, uint32_t bit, uint32_t val);

1.2 系统控制

1.2.1 系统时钟

系统上电默认使用内部 24MHz 高速时钟（HRC）作为系统时钟。

若系统运行 48MHz/32MHz 主频，建议将 APB2 总线进行 2 分频，否则可能会导致低速外设运行不正常，分频操作建议使用 ALD 库中 API：ald_cmu_div_config(CMU_PCLK_2, CMU_DIV_2);

几种常用系统时钟配置：

1.2.1.1 内部高速 24MHz（默认时钟）

此种系统时钟不需要用户做任何配置。

1. 2. 1. 2 48MHz（使用 HRC 倍频）

配置方式如下：

ald_cmu_pll1_config(CMU_PLL1_INPUT_HRC_6, CMU_PLL1_OUTPUT_48M);

ald_cmu_clock_config(CMU_CLOCK_PLL1, 48000000);

1. 2. 1. 3 外部时钟 HOSC（4~16MHz）

外部高速时钟要求为 4MHz 的倍数，如：4MHz、8MHz、12MHz、16MHz。

首先要确认焊接了外部高速时钟，并已知外部高速时钟的频率，假如外部高速时钟为 12MHz，则配置方式如下：

ald_cmu_clock_config(CMU_CLOCK_HOSC, 12000000);

1. 2. 1. 4 48MHz（使用 HOSC 倍频）

外部高速时钟要求为 4MHz 的倍数，如：4MHz、8MHz、12MHz、16MHz。

首先要确认焊接了外部高速时钟，并已知外部高速时钟的频率，假如外部高速时钟为 12MHz，则配置方式如下：

ald_cmu_pll1_config(CMU_PLL1_INPUT_HOSC_3, CMU_PLL1_OUTPUT_48M);

ald_cmu_clock_config(CMU_CLOCK_PLL1, 48000000);

1. 2. 1. 5 外部低速时钟（LOSC）

首先要确认焊接了外部低速时钟，配置方式如下：

ald_cmu_clock_config(CMU_CLOCK_LOSC, 32768);

需要注意的是，当系统时钟配置为低速时钟时（低于 1MHz），SysTick 中断将会被迫关闭。ALD 提供的延迟类函数禁止使用。

1. 2. 2 IAP 操作程序

芯片内置 IAP 自编程固化模块，由硬件电路实现。推荐使用 IAP 方式对 FLASH 进行擦、写操作，可以减少用户代码量。

1. 2. 2. 1 Flash 页擦除

页擦除可擦除固定一页空间，其中程序区一页大小可选择为 2048 Bytes 或 512 Bytes，信息区一页大小为 512 Bytes，一次页擦除耗时约 2ms。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；
2. 通过 MSC_FLASHKEY 解除 Flash 程序区或信息区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入需擦除页的首地址；
5. 设置信息区是否需使能；

6. 选择页大小；
7. 写入 MSC_FLASHCMD.CMD 命令触发页擦除；
8. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
9. 判断 MSC_FLASHSR.SERA 标志位是否置起；
10. 设置 Flash 操作请求禁止。

注：数据 Flash 页擦除流程与普通 Flash 页擦除流程一致，仅触发命令不同。

1. 2. 2. 2 页擦函数

- 函数功能：擦除指定的页
- 入口地址：0x10000004
- 输入参数：addr-擦除页的首地址，sect-擦除页大小
- 返回值：函数执行状态（0 为成功，1 为失败）

PGSZ	Bit 8	R/W	页擦除区域大小选择 0：512B 1：2KB

```
/**
  * @brief  Erases a specified page.
  * @param  addr: The beginning address of the page to be erased.
  * @param  sect: Page erase area size selection.
  * @retval The result:
  *           - 0: SUCCESS
  *           - 1: ERROR
  */
uint32_t ald_iap_erase_page(uint32_t addr, uint32_t sect)
{
    uint32_t status;
    IAP_PE iap_pe = (IAP_PE)(*(uint32_t *)IAP_PE_ADDR);

    __disable_irq();
    status = (*iap_pe)(addr, sect);
    __enable_irq();

    return !status;
}
```

1. 2. 2. 3 Flash 字编程

程序区字编程可一次编程 4 Bytes 空间，信息区字编程可一次编程 4 Bytes 空间，一次字编程耗时约 25us。具体步骤如下：

1. 检查 MSC_FLASHSR.BUSY 标志是否处于空闲状态；

2. 通过 MSC_FLASHKEY 解除 Flash 程序区或信息区保护状态；
3. 设置 Flash 操作请求使能；
4. 写入需编程地址；
5. 设置信息区是否需使能；
6. 写入需编程数据 MSC_FLASHDR.DATA；
7. 写入 MSC_FLASHCMD.CMD 命令触发字编程；
8. 等待 MSC_FLASHSR.BUSY 标志再次变为空闲状态；
9. 判断 MSC_FLASHSR.PROG 标志位是否置起；
10. 设置 Flash 操作请求禁止。

注：数据 Flash 字编程流程与普通 Flash 字编程流程一致，仅触发命令不同。

1. 2. 2. 4 单字编程函数

- 函数功能：向 Flash 指定地址写入一个字(32 位)
- 入口地址：0x10000008
- 输入参数：addr -待编程的 Flash 地址，data -待编程数据
- 返回值：函数执行状态（0 为成功，1 为失败）

```c
/**
  * @brief  Programs a word at a specified address.
  * @param  addr: Specifies the address to be programmed.
  *         Bit0-1 must be zero.
  * @param  data: Specifies the data to be programmed.
  * @retval The result:
  *           - 0: SUCCESS
  *           - 1: ERROR
  */
uint32_t ald_iap_program_word(uint32_t addr, uint32_t data)
{
    uint32_t status;
    IAP_WP iap_wp = (IAP_WP)(*(uint32_t *)IAP_WP_ADDR);

    if (addr & 0x3)
        return 1;

    __disable_irq();
    status = (*iap_wp)(addr, data);
    __enable_irq();

    return !status;
}
```

1. 2. 2. 5 多字编程

- 函数功能：向 Flash 指定地址写入多个字
- 入口地址：0x10000000
- 输入参数：addr -待编程的 Flash 首地址，data -放在 SRAM 空间的编程数据首地址，len -编程数据长度，erase -当编程到页首时是否先进行页擦除（非零为擦除，零为不擦除）
- 返回值：函数执行状态（0 为成功，1 为失败）

```c
/**
  * @brief  Programs datas at a specified address.
  * @param  addr: Specifies the address to be programmed.
  *         Bit0-1 must be zero.
  * @param  data: Specifies the data to be programmed.
  * @param  len: Specifies the data length to be programmed.
  *         Bit0-1 must be zero.
  * @param  erase: Erase page flag before programming.
  * @retval The result:
  *           - 0: SUCCESS
  *           - 1: ERROR
  */
uint32_t ald_iap_program_words(uint32_t addr, uint8_t *data, uint32_t len,
uint32_t erase)
{
    uint32_t status;
    IAP_WSP iap_wsp = (IAP_WSP)(*(uint32_t *)IAP_WSP_ADDR);

    if ((addr & 0x3) || (len & 0x3))
        return 1;

    __disable_irq();
    status = (*iap_wsp)(addr, data, len, erase);
    __enable_irq();

    return !status;
}
```

1.2.3 FLASH 读保护

规则描述：当 FLASH 的读保护级别设置为 level1 或 level2 时，运行在 SRAM 上的程序不能有读 FLASH 操作。

典型应用 1：运行在 SRAM 上的程序想读取 FLASH。将读 FLASH 操作放在 FLASH 上执行；

典型应用 2：程序运行在 SRAM 上，响应中断请求。将中断向量表拷贝至 SRAM 中，并设置中断向量偏移地址（SYSCFG_VTOR 和 SYSCFG_MRMP.VTOEN）。

1.2.4 低功耗模式

1.2.4.1 降低功耗

进入 STOP1/STOP2 模式之前，置位 LPSTOP 位可降低 STOP 功耗。

PMU 控制寄存器（PMU_CR）																													
偏移地址：00_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FSTOP	FPD	MTSTOP	LPSTOP	LPRUN	LPCD	LPVS		LPVSTK		Reserved									CSHUTOFF	CSTANDBYF	CWUF	LPM	

LPSTOP	Bit 20	R/W	**STOP 模式 LDO 低功耗使能位** 0：禁止 1：使能

1. 2. 4. 2 时钟低功耗模式

以下时钟在低功耗模式下默认禁止，若在低功耗模式下所用外设时钟源为以下时钟则需要使能。

外设时钟低功耗模式使能寄存器（CMU_LPENR）																													
偏移地址：060_H																													
复位值：$00000000\_00000000\_00000000\_00000000_B$																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												HOSCEN	HRCEN	LOSCEN	LRCEN

Reserved	Bit 31-4	—	保留
HOSCEN	Bit 3	R/W	**HOSC 时钟低功耗模式使能位** 0：禁止 1：使能
HRCEN	Bit 2	R/W	**HRC 时钟低功耗模式使能位** 0：禁止 1：使能
LOSCEN	Bit 1	R/W	**LOSC 时钟低功耗模式使能位** 0：禁止 1：使能
LRCEN	Bit 0	R/W	**LRC 时钟低功耗模式使能位** 0：禁止 1：使能

1. 2. 4. 3 LPRUN 主频限制

当 LDO 低功耗模式使能时系统主频不能超过 2M。

LPRUN	Bit 19	R/W	**LDO 低功耗模式使能位** 0：禁止 1：使能 注：使能时，系统主频不能超过 2MHz

1. 2. 4. 4 关闭 SysTick

若芯片 Systick 模块处于使能状态，且 Systick 中断也处于使能状态，在芯片进入 Sleep 模式的瞬间（执行到 WFI 指令时），若产生了 Systick 中断，会使 CPU 误判断为唤醒，而此时 SOC 接收到了 Sleep 模式的指令，停止了 CPU 时钟，无法唤醒 CPU。因此，在进入低功耗模式前需关闭 SysTick，操作方法如下：

```
    SysTick->CTRL &= ~SysTick_CTRL_ENABLE_Msk;
    __WFI();
    SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;
```

1. 2. 4. 5 解锁 PMU_BKPCR

配置寄存器 PMU_BKPCR0、PMU_BKPCR1、PMU_BKPSR 时除了解锁外还需将 bit15~bit8 配置为 0x5A 并同其他配置一起写入寄存器才可生效。

```
void ald_pmu_standby_enter(pmu_wakeup_pin_t pin, pmu_wakeup_level_t levle)
{
    uint32_t tmp = 0x5A00;

    assert_param(IS_PMU_WAKEUP_PIN(pin));
    assert_param(IS_PMU_WAKEUP_LEVEL(levle));

    SET_BIT(tmp, PMU_BKPCR0_WKPEN_MSK | PMU_BKPCR0_STBWKEN_MSK);
    MODIFY_REG(tmp, PMU_BKPCR0_WKPS_MSK, pin << PMU_BKPCR0_WKPS_POSS);
    MODIFY_REG(tmp, PMU_BKPCR0_WKPL_MSK, levle << PMU_BKPCR0_WKPL_POS);

    SYSCFG_UNLOCK();
    PMU->BKPCR0 = tmp;
    MODIFY_REG(PMU->CR, PMU_CR_LPM_MSK, PMU_LP_STANDBY <<
PMU_CR_LPM_POSS);
    SYSCFG_LOCK();

    SysTick->CTRL &= ~SysTick_CTRL_ENABLE_Msk;
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
    __WFI();
    SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;

    return;
}
```

1. 3 外设

1. 3. 1 GPIO 模块

1. 3. 1. 1 功能配置

使用复用功能时必须先对 GPIO 端口复用功能寄存器 0 或 GPIO 端口复用功能寄存器 1 进行配

置，对应的配置参数请查看数据手册管脚功能复用表，表中 ALT 号对应程序 FUNC 号。

◇ 当复用功能为输入时，端口必须配置为输入模式（浮空、上拉或下拉），且输入引脚由外部驱动。

◇ 当复用功能为输出时，端口必须配置为输出模式（推挽、开漏）。在配置开漏模式时，GPIO 输入功能有效，此时可以用作双向模式。

如果端口复用功能为输出，则引脚会和片上外设的输出信号连接，如果对应外设没有被激活，GPIO 的输出信号将不确定。

▪3.2 管脚功能定义

Pin Number		PIN NAME	ALT0 (复位后功能)	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
LQFP48	LQFP32									

```
    gpio_init_t x;
    x.mode = GPIO_MODE_OUTPUT;       //输入输出模式
    x.odos = GPIO_PUSH_PULL;         //开漏推挽配置
    x.pupd = GPIO_PUSH_UP;           //上下拉配置
    x.podrv = GPIO_OUT_DRIVE_1;      //PMOS 驱动配置
    x.nodrv = GPIO_OUT_DRIVE_1;      //NMOS 驱动配置
    x.flt  = GPIO_FILTER_DISABLE;    //输入滤波配置
    x.type = GPIO_TYPE_CMOS;         //端口类型配置
    x.func = GPIO_FUNC_1;            //复用功能配置
    ald_gpio_init(GPIOA, GPIO_PIN_0, &x);
```

1.3.1.2 电平输出控制

控制 IO 输出高低电平时，推荐使用 GPIO_BSRR 寄存器控制。若使用 GPIO_DOUT 寄存器控制 IO 输出高低电平，当主循环和中断函数中控制同一个 PORT 口上的 IO 时，有概率在循环中将中断函数控制的 IO 覆盖掉，导致中断函数中控制的 IO 不起作用。

```
void ald_gpio_write_pin(GPIO_TypeDef *GPIOx, uint16_t pin, uint8_t val)
{
    assert_param(IS_GPIO_PORT(GPIOx));
    assert_param(IS_GPIO_PIN(pin));

    if ((val & (0x01)) == 0x00)
        GPIOx->BSRR = (pin << 16);
    else
        GPIOx->BSRR = pin;

    return;
}
```

1.3.2 TIMER 模块

1.3.2.1 使能输出比较预载

应用 PWM 模式时若需改变周期或占空比，使能输出比较预载，可使输出 PWM 波形连续。

CH1OPREN	Bit 3	R/W	**输出比较 1 预载使能** 0：GP16C4Tn_CCVAL1 的预载寄存器禁止。GP16C4Tn_CCVAL1 在任何时候都可写，新写入的值将立刻生效。 1：GP16C4Tn_CCVAL1 的预载寄存器使能。读/写操作可访问预载寄存器。每当发生一次更新事件，GP16C4Tn_CCVAL1 预载入值将会被填入有效寄存器。 注意： 仅在单脉冲模式下（GP16C4Tn_CON1 寄存器中
			的 SPMEN 设置为 1），PWM 模式可在不经过验证预载寄存器的情况下使用。其他情况下的行为不做保证。

1.3.2.2 正确配置管脚上下拉

PWM 模式未输出时通道的空闲电平会受到上下拉控制影响（上拉为高电平，下拉为低电平）。计数器模式做水位检测应用时输入引脚不应配置上下拉，否则会影响检测电平。

1.3.2.3 捕获频率限制

使用 GP16C4T 和 DMA 反复循环读取捕捉寄存器数值时，需要满足系统时钟大于等于 20 倍以上的外部信号频率。DMA 从请求产生到读取寄存器，每一次需要约 18 个系统时钟的时间，如果当使用系统时钟为 12M 去捕捉 1M 的信号时，捕捉周期仅 12 个时钟，将不满足 DMA 单次读取时间。

1.3.2.4 Class B 计数值初始化

未初始化的时钟自检计数值为随机值，可能导致自检失效，因此需进行初始化清零操作。

1.3.3 串口模块

1.3.3.1 使用外部晶振

串口通讯波特率冗余度约±3%，通讯双方应至少有一方使用外部晶振作为时钟源，保证通讯在全温下波特率没有太大偏差。

1.3.3.2 使用新版本 SDK

原版本 SDK 发送和接收状态放在同一个变量 state 中，且收发配置过程都会受变量 lock 控制。全双工通信当发送和接收分别在主循环和中断中配置时有概率发生冲突，导致配置不成功。最新版本已做收发 state 的区分，并修正 lock 变量的配置冲突影响。

```
void ald_uart_reset(uart_handle_t *hperh)
{
  . . . . . .
    hperh->state    = UART_STATE_RESET;
    hperh->rx_state = UART_STATE_RESET;
  . . . . . .
}
```

1. 3. 3. 3 添加异常恢复处理

当传输受干扰出现错误，如上溢错误、帧错误时在错误处理回调函数中增加错误恢复处理。并清空数据帧出错的接收数组，复位数组指针和数据长度。

```
void uart_error(uart_handle_t *arg)
{
   /* Handler ERROR */
  ald_rmu_reset_periperal(RMU_PERH_UART0);  //复位外设模块
  uart_init();                              //重新初始化外设
    return;
}
```

1. 3. 3. 4 USART 的 IDLE 中断使能

IDLE 中断需要在接收到一个数据之后再打开，否则该中断会立即触发，并且中断标志不能被清除。

1. 3. 3. 5 UART 配合 DMA 使用

配合 DMA 使用时，DMA 的 burst 必须为 1，R_power 必须为 0。

1. 3. 3. 6 UART 读写 FIFO 处理

在中断模式下，发送时若向 TX_FIFO 中写入多个数据，则每次写数据前需确认 FIFO 是否满，避免 FIFO 上溢。接收时需要在接收中断函数中将 RX_FIFO 读空，否则会导致后续再收到数据也进不了 RX 中断。该项应用注意在 ALD 库中已经做了处理，使用 MD 库时需自行处理。

1. 3. 3. 7 UART 溢出错误处理

当发生溢出错误时，需要将 RX_FIFO 进行复位。该项应用注意在 ALD 库中已经做了处理，使用 MD 库时需自行处理。

1. 3. 3. 8 UART 波特率冗余度

数据接收时最大波特率误差需在±3%内。

1. 3. 3. 9 LPUART 时钟源选择

LPUART 默认使用低速时钟，当波特率需要达到 9600 以上时请使用高速时钟配置初始化函数。void ald_lpuart_highspeed_init(lpuart_handle_t *hperh)(); 同时确认函数内的 CLKDIV 配置是否正确，若不正确请直接对 CLKDIV 赋值。

1. 3. 3. 10 LPUART 发送禁止中断

LPUART 为低速外设，在向 TXDR 写入数据后需等待 TX-FIFO 为非空，当使用高速时钟作为时钟源时判断等待时间也变短。若在写入数据和等待之间发生中断，可能在退出中断后 TX-FIFO 已

空，导致判断失败而发送异常。因此，需要在发送前关闭中断，等发送完成后再打开中断。

```
ald_status_t ald_lpuart_send(lpuart_handle_t *hperh, uint8_t *buf,
uint16_t size, uint32_t timeout)
{
......
        __disable_irq();
        LPUART0->TXDR = *buf++;
        cnt = 0x10000;
        while ((LPUART0->STAT & 0x4000) && --cnt);
        __enable_irq();
......
}
```

1.3.4 I2C 模块

1.3.4.1 释放时钟线

I2C 工作在主机模式时，若从机一直拉低时钟线，主机可以通过关闭 I2C 模块来释放时钟线。

1.3.4.2 解除死锁

I2C 作为主机与外部 EEPROM 进行通信，若 SDA 线一直被 EEPROM 器件拉低，可以通过以下 2 种方式解除死锁现象：

1. 通过重新给 EEPROM 上电解除死锁
2. 主机 SCL 线模拟输出 9 个时钟信号后，再模拟发送一个停止信号，EEPROM 会终止此次通信并释放 SDA 线。

1.3.4.3 地址匹配后使能 STOP 中断

I2C 工作在从机模式时，会响应 stop 命令的中断，需在地址匹配标志置起后再使能 stop 中断，否则会误响应其他地址的 stop 命令。

1.3.4.4 读取时再写发送数据

I2C 工作在从机模式时，主机发送读取命令之前写入发送 buffer 的值无效。

1.3.4.5 及时发送 STOP

I2C 工作在主机模式，若发送数据后从机返回的是 NACK，默认会重发当前字节，直至收到 ACK 为止。如果应用时与通讯协议不符，请使用 GPIO 软件模拟 I2C 主机。

1.3.5 SPI 模块

SPI 在只读模式下（CON1.RXO 置位），STAT.BUSY 标志会一直被置起，故不能通过该位判断通信是否完成。若需要清除 STAT.BUSY 标志，需要将 CON1.RXO 清零。

注意不支持 DMA 模式下的 CRC 功能。

1.3.6 RTC 模块

1.3.6.1 寄存器两次写入

设置时间和日期时需对寄存器连续写入两次，否则复位时值不保存，参考库函数处理。

```
ald_status_t ald_rtc_set_time(rtc_time_t *time, rtc_format_t format)
{
    ......
    RTC_UNLOCK();
    WRITE_REG(RTC->TIME, tmp);
    for (tmp1 = 0; tmp1 < 200; tmp1++)
    {
        __NOP();
    }
    WRITE_REG(RTC->TIME, tmp);
    RTC_LOCK();
    ......
}

ald_status_t ald_rtc_set_date(rtc_date_t *date, rtc_format_t format)
{
    ......
    RTC_UNLOCK();
    WRITE_REG(RTC->DATE, tmp);
    for (tmp1 = 0; tmp1 < 200; tmp1++)
    {
        __NOP();
    }
    WRITE_REG(RTC->DATE, tmp);
    RTC_LOCK();
    ......
}
```

1. 3. 7 ADC 模块

1. 3. 7. 1 参考电压配置

即使使用外部输入做参考，也需要将内部参考电压 VREFEN 使能。

正向参考电压选择 VREFP 管脚时需使能 Buffer。

内部 2V 参考电压非高精度参考源，不推荐作为正向参考电压应用于 ADC 测量要求较高的场景。

1. 3. 7. 2 结果数据处理

若发现在输入某一电压后数据转换的结果有较大波动，在对采样结果做处理时请使用采样多次去最大最小后取平均的方式，推荐使用 ALD 库中的数据处理方式。

```c
/**
  * @brief  Get the average value of the normal channel.
  * @retval Average voltage.
  */
int32_t get_normal_average_voltage(void)
{
    uint32_t i, tmp, min = 0xFFFF, max = 0, vol, sum = 0;

    for (i = 0; i < 18; ++i)
    {
        /* Start normal convert */
        ald_adc_normal_start(&h_adc);

        /* Polling convert finish and read result */
        if (ald_adc_normal_poll_for_conversion(&h_adc, 5000) == OK)
            tmp = ald_adc_normal_get_value(&h_adc);
        else
            return -1;

        max  = tmp > max ? tmp : max;
        min  = tmp < min ? tmp : min;
        sum += tmp;
    }

    sum = sum - max - min;  //滤除最大和最小值
    vol = sum >> 4;         //数据取平均

    return vol;
}
```

1.3.7.3 输入信号处理

当输入信号驱动能力较弱时，在应用函数例程中初始化可使能 TRMEN、IREFEN、VCMBUFEN、VRBUFEN。当输入信号不稳定时可在检测口增加一滤波电容。

1.3.7.4 IO 灌/拉电流影响

IO 引脚的灌拉电流对其他引脚电压有拉高低影响，会影响 ADC 采样结果，每灌/拉 1mA 电流最大约拉高/低 2mV。

1.3.7.5 温感精度

内部温感温度偏差最大达 5 摄氏度，注意应用精度需求。应用时正向参考电压需选择内部 2V。

1.3.8 TK 模块

1.3.8.1 使用新版本库

新版本触控库针对上电和进休眠时的基线确认进行了滤波和异常恢复处理，可避免原版本的上电按键误触和唤醒异常问题。同时注意库版本不要混用，触控各文件版本保持统一。

1.3.8.2 使用内部充电电压

默认使用内部充电电压，防止因使用 VDD 造成的纹波干扰和板间差异影响。若充电电压要用 VDD 以提升信号，则必须进行电源电压冗余度测试以及整机强干扰环境信噪比评估测试。

1.3.8.3 管脚配置

TK 通道应配成模拟输入，且不应设置上下拉，否则会造成按键误触等问题

1.3.8.4 频率配置

充放电频率不宜过高否则充放电不完全，软件抖频时造成数据波动，不建议使用高于 4M 的充放电频率。频率通过分频寄存器配置。

TKS 控制寄存器 0（TKS_CON0）

偏移地址：000_H

复位值：$00000000\_00110000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TKDIV			CMUDIV	Reserved		EXTEN	CCSEL	Reserved	TKDCS	DCEN	SMPSEL		DUTY	Reserved	AVEEN	RESMD	RESSEL	Reserved	CHSE	NCHOE	BUSY	START	EN

TKDIV	Bit 23-21	R/W	**TKS工作时钟分频选择位（分频后为充放电开关时钟）** 000：1分频 001：2分频 010：4分频 011：8分频 100：16分频 101：32分频
CMUDIV	Bit 20	R/W	**TKS工作时钟预分频选择位** 0：1分频 1：3分频

1.3.9 LCD 模块

1.3.9.1 寄存器配置

高驱动模式电压驱动器必须使能，驱动电流需配置为最大，驱动电阻需配置为最小。

```
void lcd_init(lcd_handle_t *h_lcd)
{
    ......
    h_lcd->init.lcd_dsld   = 0xF;              //驱动电流
    h_lcd->init.lcd_dshd   = 0xF;              //驱动电流
    h_lcd->init.lcd_reshd  = LCD_RES_1MOHM;    //驱动电阻
    h_lcd->init.lcd_resld  = LCD_RES_1MOHM;    //驱动电阻
    h_lcd->init.lcd_vbufhd = LCD_FUNC_ENABLE; //高驱电压驱动
    ......
}
```

1.3.9.2 端口禁用

当LCD电源选择charge pump供电，LCD时钟需选1MHz或者当LCD电源选择VLCD且VLCD电压高于VDD时以下端口禁止作为SEG使用

PC0（SEG22）
PC1（SEG23）
PB10（SEG10）
PB11（SEG11）
PB12（SEG12）
PB8（SEG21）

1.3.9.3 复用LED

复用为LED功能时，不可驱动共阳型的数码管，且最大支持16个显示段（SEG0~15）和8个公共端（COM0~7）

1.3.10 PIS模块

PIS模块不支持TIMER的触发事件，控制另外一个TIMER。

1.4 其他

1.4.1 IWDT唤醒

若要使用IWDT进行唤醒并复位，须保证FLASH低功耗暂停（FSTOP）和掉电（FPD）未使能，否则无法正常唤醒复位。

PMU控制寄存器（PMU_CR）

偏移地址：00_H

复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FSTOP	FPD	MTSTOP	LPSTOP	LPRUN	LPCD	LPVS		LPVSTK		Reserved									CSHUTOFF	CSTANDBYF	CWUF		LPM

FSTOP	Bit 23	R/W	**Flash低功耗模式暂停使能位** 0：禁止 1：使能
FPD	Bit 22	R/W	**Flash低功耗模式掉电使能位** 0：禁止 1：使能

1.4.2 LOSC配置

在standby模式下使用RTC，需要配置字LOSCEN配置为强制使能。在stop2模式下使用RTC，需要配置CMU->CLKENR.LOSCEN=1同时CMU->LPENR.LOSCEN=1，或者配置字LOSCEN配置为强制使能。否则，在低功耗阶段，LOSC无法工作。

1.4.3 低功耗中断配置

在低功耗模式下注意将未使用的中断都关闭，否则可能导致误唤醒并影响如TK等功能的正常工作。

第2章 硬件设计指南

2.1 电源

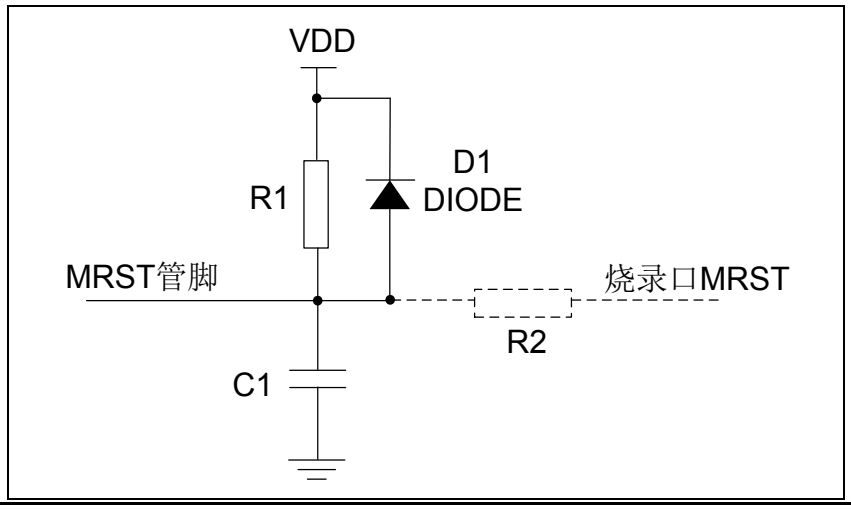
电源纹波峰峰值应控制在 100mV 以内，VDD 与 VSS 之间的去耦电容并联使用 4.7uF 和 0.1uF，并靠近芯片引脚放置，单独给芯片供电。

常温供电 5V 条件下，VDD 上升速率需大于 800us/V，下降速率需大于 100us/V。

为保证上电基线稳定，TK 应用要求 VDD 上电速率需快于 7.5V/S。

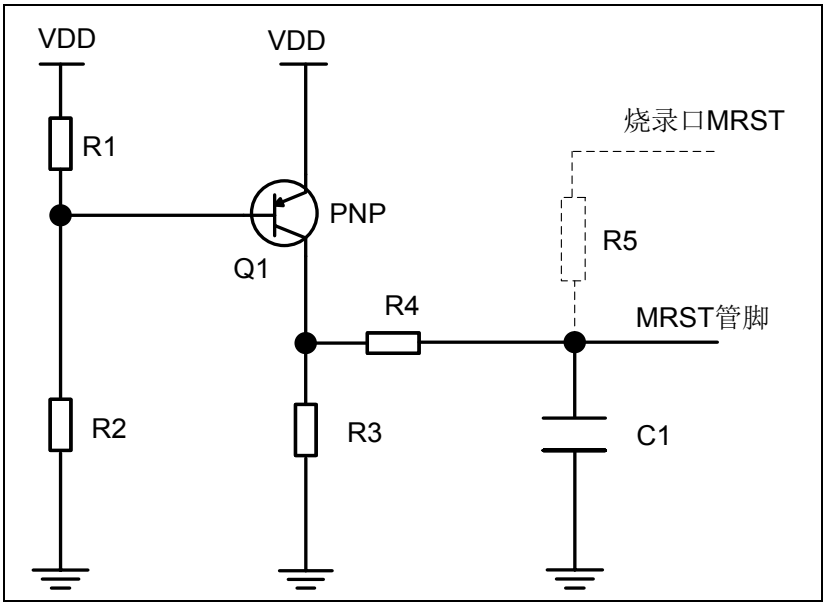
2.2 MRST 管脚

以下为 MRST 参考应用电路一：



注：采用 RC 复位，其中 47KΩ≤R1≤100KΩ，电容 C1=0.1μF，在有外部烧录口的应用系统中，需要串接 R2 作为限流电阻，0.1KΩ≤R2≤1KΩ。

以下为 MRST 参考应用电路二：



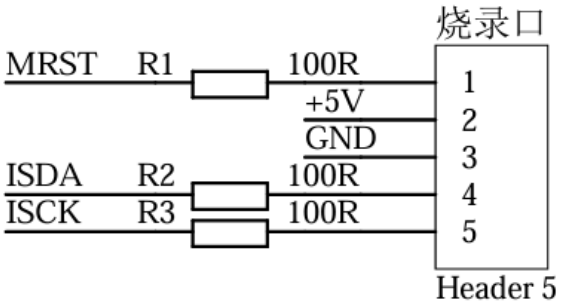
注：采用 PNP 三极管复位，通过 R1（2KΩ）和 R2（10KΩ）分压作为基极输入，发射极接 VDD，集电极一路通过 R3（20KΩ）接地，另一路通过 R4（47KΩ≤R4≤100KΩ）和 C1（0.1μF）接地，C1 另一端作为 MRST 输入，

当 VDD 为 3.3v 时，建议 R1 为 4.7KΩ，R2 为 8KΩ；VDD 为 5v 时，建议 R1 为 1.5KΩ，R2 为 5.1KΩ；在有外部烧录口的应用系统中，需要串接 R5 作为限流电阻，0.1KΩ≤R5≤1KΩ。

2.3 烧录口

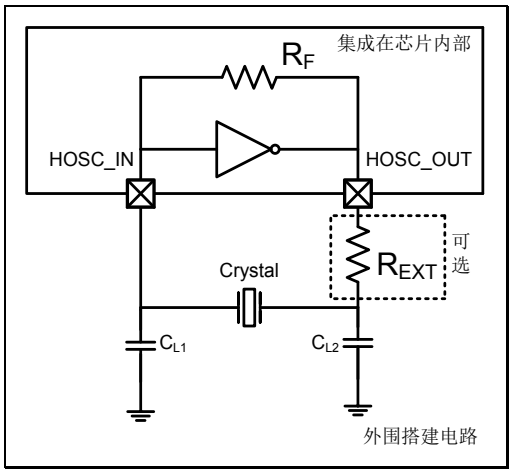
若板上有预留烧录口，在靠近端子处应串接电阻增强电气防护。

因 4 线 SWD 做调试接口时不能被占用,故烧写引脚复用为 GPIO 功能后,keil 里面不能下载程序,可使用 5 线制 SWD 并在 keil 中使用 under Reset 功能。或者使用我司 ESlinkII 工具用 ESburner 软件进行下载。烧录过程需保证 FLASH 编程电压为校准值，请使用最新版本的官方烧录工具。



2.4 外部晶振口

外部高速振荡器的典型应用连接：

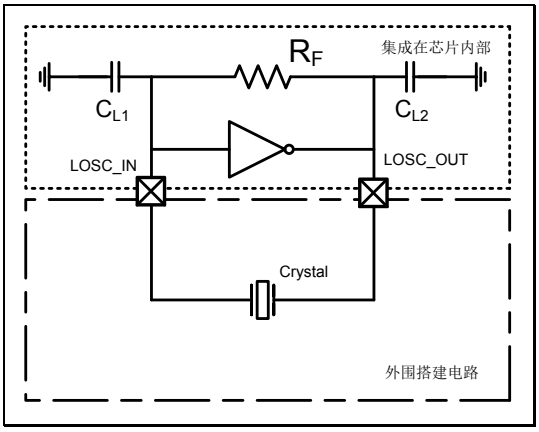


注 1： R_{EXT} 为可选电阻，其阻值取决于具体晶振规格特性

注 2： 当晶振频率为 1~8M，要求 ESR≤150Ω，且 C_{L1}、C_{L2} 容值需小于 15pF

注 3： 当晶振频率为 8~16M，要求 ESR≤40Ω，且 C_{L1}、C_{L2} 容值需小于 12pF，供电不建议超过 5V

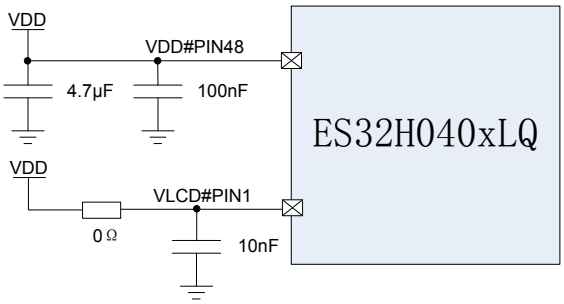
下图为外部低速振荡器的典型应用连接：



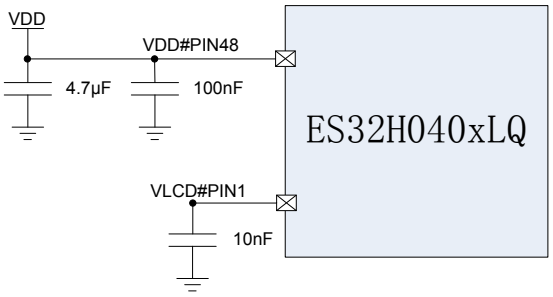
芯片上电后在载入芯片配置字之前 PC14、PC15 为 LOSC 功能,有短暂输出驱动能力,应用时避免作为关键控制脚使用。

2. 5 VLCD 管脚

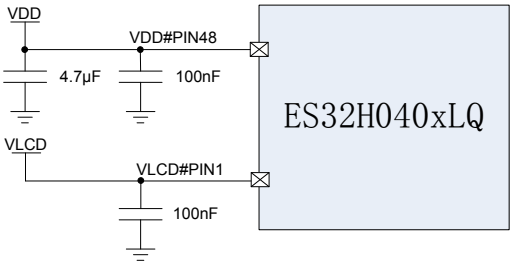
1.未使用内部升压泵的情况下，VLCD 管脚挂接 10nF 电容到 VSS，并串 0Ω 电阻到 VDD。



2.使用内部升压泵的情况下，VLCD 管脚挂接 10nF 电容到 VSS，且与 VDD 不连接。

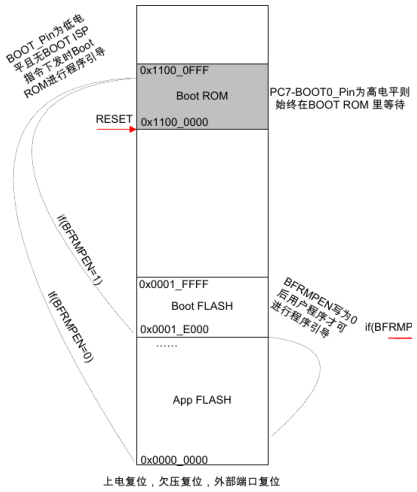


3.LCD 使用外部供电时，VLCD 管脚挂接 100nF 去耦电容并接到外部供电电压源。



2.6 BOOT-PIN 脚

根据启动引导说明 PC7 上电时需保持低电平，若为高则始终在 BOOTROM 中等待，不使用此功能时禁止接外部上拉电阻。



2.7 TK 设计要求

2.7.1 元器件规格要求

2.7.1.1 电源

类型	规格	其他要求
去耦电容	4.7uF+0.1uF	靠近芯片管脚
储能电容	10uF	靠近芯片管脚，使用瓷片电容或电解电容，如果触摸按键受纹波影响较大时需使用规格在 100uF 以上的电容

2.7.1.2 Cx 电容

类型	规格	其他要求
Cx 电容	4.7nF~10nF	靠近 Cx 管脚，使用 X7R 或 NP0 的低温漂材质。

2.7.1.3 触控通道电阻

类型	规格	其他要求
防 ESD 电阻	500Ω	靠近芯片，最大 1K

2.7.2 PCB 设计要求

2.7.2.1 按键

接触面积	80mm^2
键间距	5mm
地间距	2mm
线间距	2mm

弹簧方案走线只需要在弹簧区域留一个过孔接弹簧，并且弹簧区域内不要有其它走线。推荐使用多圈弹簧，根据面板厚度增加圈数，直径要求 10mm 以上。如果使用的是膜片，注意检查按键

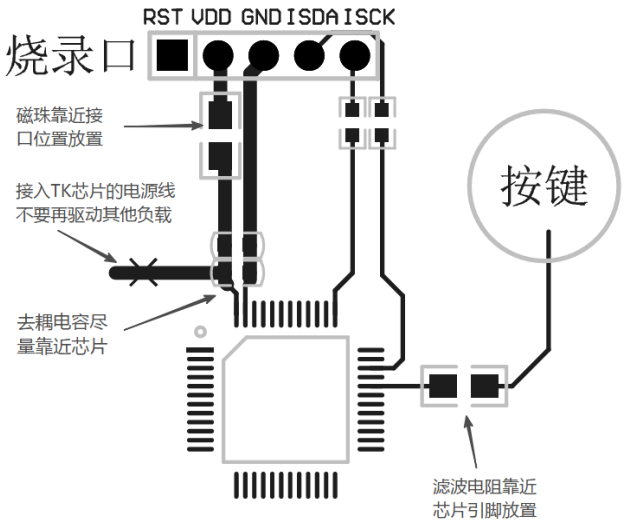
通道不能存在短路或开路，否则会造成按键无法正常响应。

2.7.2.2 电源

纹波要求	100mv 以内（5V 供电系统）
线宽	1mm

芯片端的去耦电容应尽量靠近芯片的电源和地管脚。

如下图所示，连接到触控芯片上的电源线不要再引出去驱动其他负载。



2.7.2.3 走线

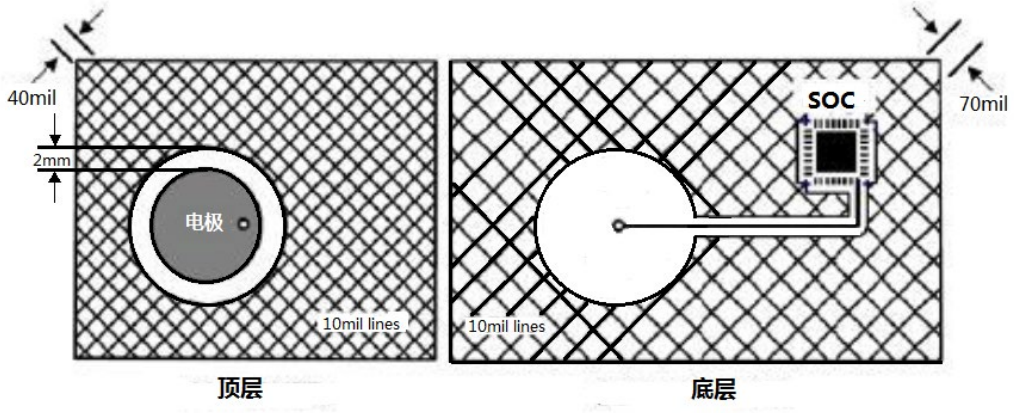
线宽		0.2mm
线长		<按键面积/(3*线宽)
线间距	触控通道	0.2mm
	通信或驱动走线	0.25mm
地间距		1mm
键间距		2mm

触控引脚除了可串联电阻，不可有其他元器件（如滤波电容，稳压二极管）挂在触控引脚上。

2.7.2.4 铺地

铺地方式		斜 45° 网格地
网格地线宽		0.25mm
网格地间隙	顶层	1mm
	底层	1.75mm
键间距		2mm
线间距		1mm

触控电极背面禁止铺地。



2. 8 普通 GPIO

IO 上的电压不可高于 VDD，否则将影响 TK 结果数据或造成 LOSC 停振。

未使用引脚需固定电平，建议软件中配成普通 IO 并输出低，硬件电路串电阻接地。

在上下电过程中端口可能存在弱驱动的脉冲输出,对地挂接 10K 以下电阻可有效避免脉冲。

应用时避免出现端口过压或负压的情况。

第3章 其他

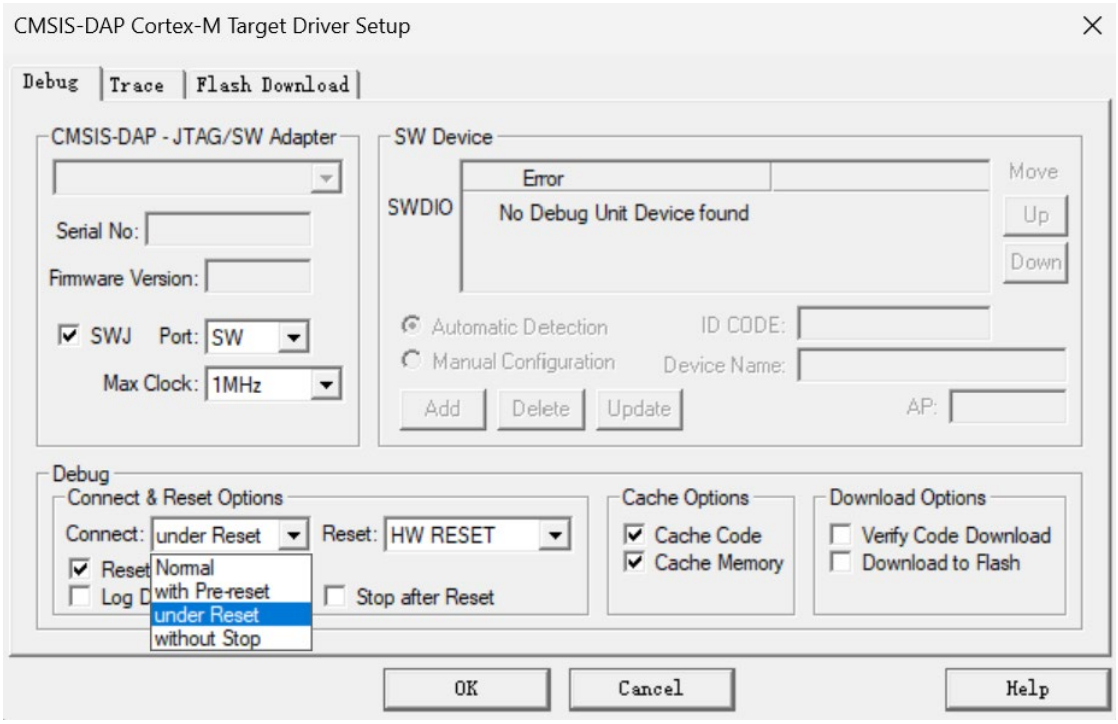
3.1 配置字

使用 ESLink II/ESLink II mini 修改芯片配置字之后，芯片需要断电后重新上电，才能正常调试。

使用 ESBurner 擦除芯片后，需要执行配置字编程，否则可能导致芯片运行不正常，即点击“擦除”按钮后，需要点击“配编”按钮。

3.2 烧录

当烧写引脚复用功能时，若在 keil 中无法识别芯片，可在 keil 中使用 under Reset 功能（需接 MRST），或者使用我司 ESlinkII 工具用 ESburner 软件进行下载。



3.3 仿真

可配置 DBGC_APB2FZ 寄存器使能 IWDT_STOP 位，使得在仿真调试暂停时不会因看门狗超时复位。

APB2 外设调试冻结寄存器（DBG_APB2FZ）

偏移地址：$00C_H$

上电复位值：$00000000\_00000000\_00000000\_00000000_B$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																			LP16T0_STOP	Reserved	RTC_STOP	WWDT_STOP	IWDT_STOP	Reserved							I2C1_SMBUS_TO	I2C0_SMBUS_TO

IWDT_STOP	Bit 8	R/W	**IWDT 调试暂停选择位** 0：内核停止时，仍正常计数 1：内核停止时，暂停计数

例如，下图程序在断点处停留即使超过 IWDT 设定时间，继续运行也不复位。