

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	2025-09-15	初版发布

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/

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承担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

<b>第 1 章</b>	<b>软件设计指南</b> .....	<b>4</b>
1.1	概述 .....	4
1.1.1	寄存器写保护 .....	4
1.1.2	写 1 清零寄存器 .....	5
1.1.3	位带操作 .....	5
1.2	系统控制 .....	5
1.2.1	系统时钟 .....	5
1.2.2	IAP 操作程序 .....	6
1.2.3	低功耗模式 .....	9
1.3	外设 .....	11
1.3.1	GPIO 模块 .....	11
1.3.2	TIMER 模块 .....	12
1.3.3	串口模块 .....	13
1.3.4	I2C 模块 .....	14
1.3.5	SPI 模块 .....	14
1.3.6	RTC 模块 .....	14
1.3.7	ADC 模块 .....	15
1.3.8	TK 模块 .....	16
1.3.9	LCD 模块 .....	17
1.4	其他 .....	18
1.4.1	IWDT 唤醒 .....	18
1.4.2	LOSC 配置 .....	18
1.4.3	低功耗中断配置 .....	18
<b>第 2 章</b>	<b>硬件设计指南</b> .....	<b>19</b>
2.1	电源 .....	19
2.2	MRST 管脚 .....	19
2.3	烧录口 .....	20
2.4	外部晶振口 .....	20
2.5	VLCD 管脚 .....	21
2.6	BOOT-PIN 脚 .....	22
2.7	TK 设计要求 .....	22
2.7.1	元器件规格要求 .....	22
2.7.2	PCB 设计要求 .....	22
2.8	普通 GPIO .....	24

## 第1章 软件设计指南

### 1.1 概述

#### 1.1.1 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。

##### 1.1.1.1 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。

##### 1.1.1.2 RTC 写保护

对 RTC\_WPR 寄存器以字方式写入 0x55AAA55 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。

##### 1.1.1.3 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。

##### 1.1.1.4 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。

### 1.1.2 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;

### 1.1.3 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( $0 \leq N \leq 31$ ),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 512Bytes，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 <sup>Ⓐ</sup>	Bit 8 <sup>Ⓐ</sup>	R/W <sup>Ⓐ</sup>	页擦除区域大小选择 <sup>Ⓐ</sup> 0: 512B <sup>Ⓐ</sup> 1: 2KB <sup>Ⓐ</sup>
-------------------	--------------------	------------------	---

```

/**
 * @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)

```
/**
 * @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)

```

/**
 * @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 低功耗模式

#### 1.2.3.1 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: 060H																																	
复位值: 00000000_00000000_00000000_00000000b																																	
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	<b>HOSC 时钟低功耗模式使能位</b> 0: 禁止 1: 使能			

HRCEN	Bit 2	R/W	<b>HRC 时钟低功耗模式使能位</b> 0: 禁止 1: 使能
LOSCEN	Bit 1	R/W	<b>LOSC 时钟低功耗模式使能位</b> 0: 禁止 1: 使能
LRCEN	Bit 0	R/W	<b>LRC 时钟低功耗模式使能位</b> 0: 禁止 1: 使能

### 1.2.3.2 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	<b>LDO 低功耗模式使能位</b> 0: 禁止 1: 使能 注：使能时，系统主频不能超过 2MHz
-------	--------	-----	--

### 1.2.3.3 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;
```

### 1.2.3.4 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
LOFP48	LOFP32									

```

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 <sup>o</sup>	Bit 3 <sup>o</sup>	R/W <sup>o</sup>	<p><b>输出比较 1 预载使能<sup>o</sup></b></p> <p>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。↓</p> <p>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。↓</p> <p>注意：↓</p> <p>仅在单脉冲模式下（GP16C4Tn_CON1 寄存器中</p>
-----------------------	--------------------	------------------	---

		的 SP MEN 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中，当发送和接收分别在主循环和中断中处理时有概率发生冲突，新版本已做更新和区分。

```
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_periperhal(RMU_PERH_UART0); //复位外设模块
    uart_init();                               //重新初始化外设
    return;
}
```

### 1.3.3.4 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);
```

### 1.3.3.5 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 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 参考电压使能

即使使用外部输入做参考，也需要将内部参考电压 VREFEN 使能。

#### 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。

```
/**
 * @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 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。

### 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一致。

#### 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: 000H																															
复位值: 00000000_00110000_00000000_00000000 <sub>B</sub>																															
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	<b>TKS工作时钟分频选择位（分频后为充放电开关时钟）</b> 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	<b>TKS工作时钟预分频选择位</b> 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)

## 1.4 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: 00H																															
复位值: 00000000_00000000_00000000_00000000 <sub>b</sub>																															
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

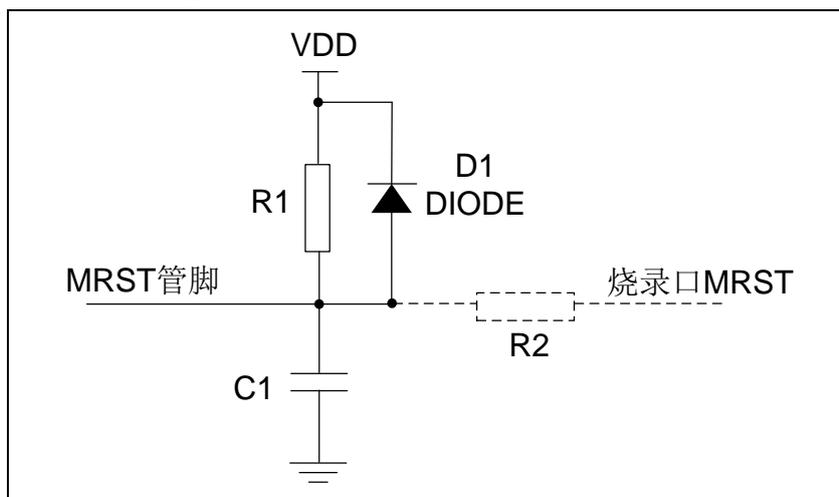
电源纹波应控制在 $\pm 200\text{mV}$ 以内，VDD与VSS之间的去耦电容并联使用 $4.7\mu\text{F}$ 和 $0.1\mu\text{F}$ ，并靠近芯片引脚放置，单独给芯片供电。

常温供电 $5\text{V}$ 条件下，VDD上升速率需大于 $800\mu\text{s/V}$ ，下降速率需大于 $100\mu\text{s/V}$ 。

为保证上电基线稳定，TK应用要求VDD上电速率需快于 $7.5\text{V/S}$ 。

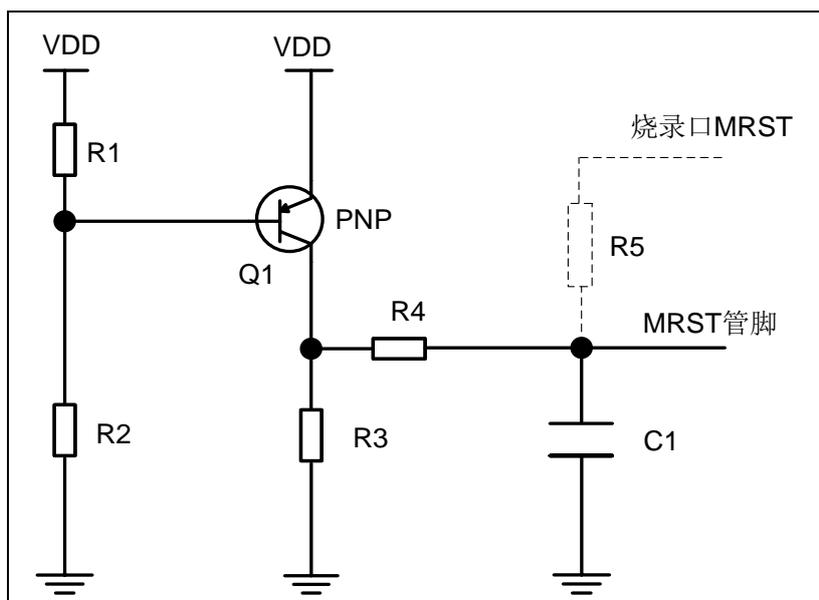
### 2.2 MRST 管脚

以下为MRST参考应用电路一：



注：采用RC复位，其中 $47\text{K}\Omega \leq R1 \leq 100\text{K}\Omega$ ，电容 $C1=0.1\mu\text{F}$ ，在有外部烧录口的应用系统中，需要串接 $R2$ 作为限流电阻， $0.1\text{K}\Omega \leq R2 \leq 1\text{K}\Omega$ 。

以下为MRST参考应用电路二：



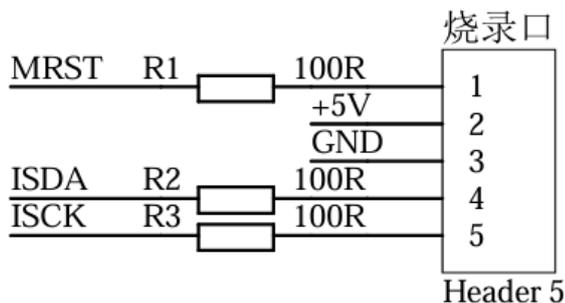
注：采用PNP三极管复位，通过 $R1$  ( $2\text{K}\Omega$ )和 $R2$  ( $10\text{K}\Omega$ )分压作为基极输入，发射极接VDD，集电极一路通过 $R3$  ( $20\text{K}\Omega$ )接地，另一路通过 $R4$  ( $47\text{K}\Omega \leq R4 \leq 100\text{K}\Omega$ )和 $C1$  ( $0.1\mu\text{F}$ )接地， $C1$ 另一端作为MRST输入

，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， $0.1K\Omega \leq R5 \leq 1K\Omega$ 。

## 2.3 烧录口

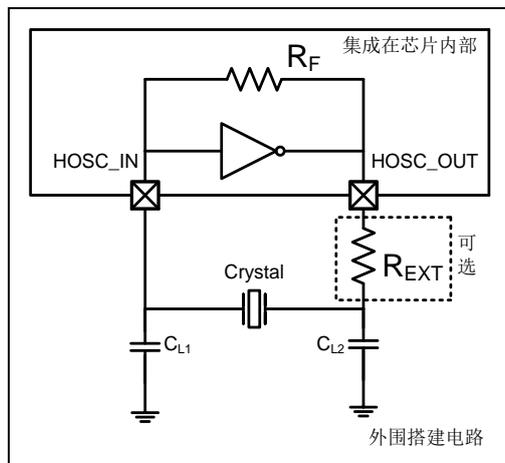
若板上有预留烧录口，在靠近端子处应串接电阻增强电气防护。

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：

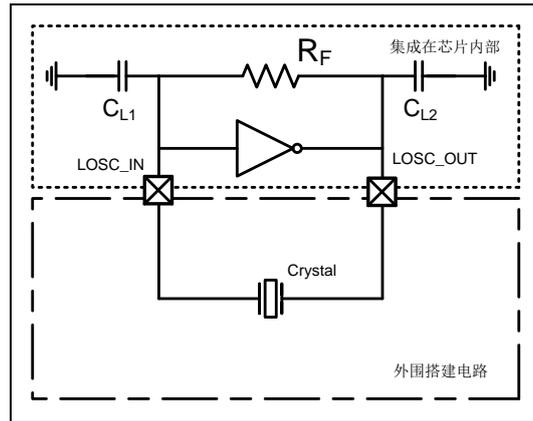


注 1:  $R_{EXT}$  为可选电阻，其阻值取决于具体晶振规格特性

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

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

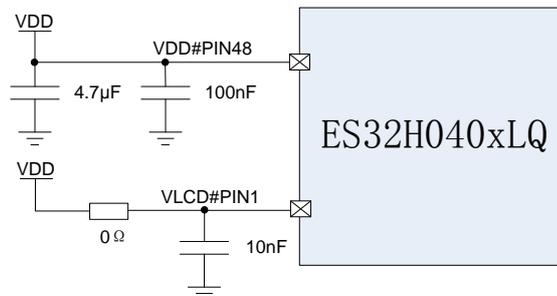
下图为外部低速振荡器的典型应用连接：



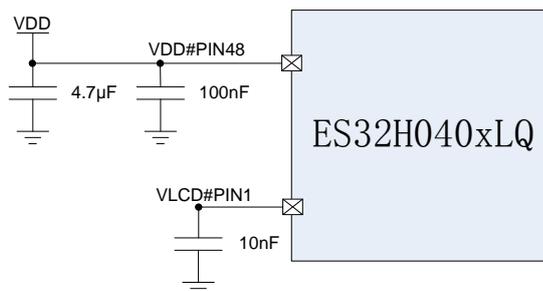
芯片上电后在载入芯片配置字之前 PC14、PC15 为 LOSC 功能,有短暂输出驱动能力,应用时避免作为关键控制脚使用。

## 2.5 VLCD 管脚

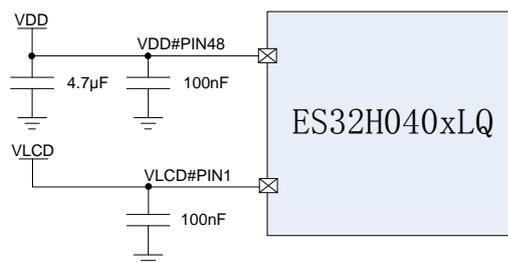
1.未使用内部升压泵的情况下，VLCD 管脚挂接 10nF 电容到 VSS，并串 0Ω 电阻到 VDD。



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。

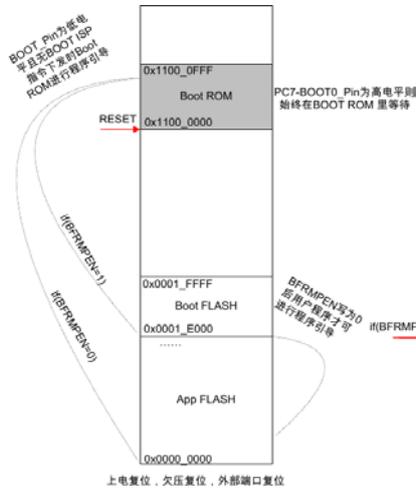


3.LCD 使用外部供电时，VLCD 管脚挂接 100nF 去耦电容并接到外部供电电压源。



## 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	500 Ω	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 <sup>2</sup>
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

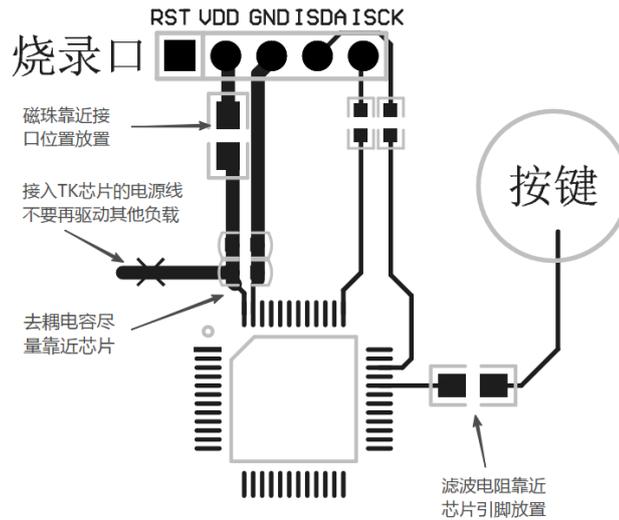
通道不能存在短路或开路，否则会造成按键无法正常响应。

### 2.7.2.2 电源

纹波要求	200mv 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

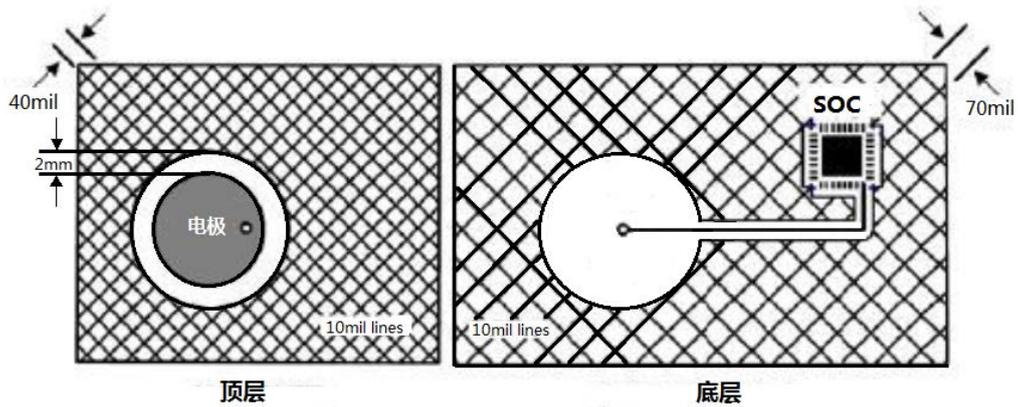
线宽	0.2mm	
线长	<按键面积/(3*线宽)	
线间距	触控通道	0.2mm
	通信或驱动走线	0.25mm
地间距	1mm	
键间距	2mm	

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。