文档编号：AN2038

上海东软载波微电子有限公司

应用笔记

ES32F0283

修订历史

版本	修改日期	更改概要
V1.0	2022-04-19	初版
V1.1	2022-06-08	1. 修改 AN 章节架构 2. 补充 DMA 模块注意事项 3. 补充温度感测模块注意事项 4. 移除 Bootrom demo 说明章节，改为另外文件说明 5. 增加最小系统电路
	2022-06-09	1. 增加 ADC 推算真实电压方法 2. 修改实际温度的方法
	2022-06-13	增加低功耗唤醒注意事项
V1.2	2022-09-21	1. 增加低功耗唤醒注意事项 2. 修正 ADC 补偿使用方式
V1.3	2022-11-02	1. 修改低功耗唤醒注意事项 2，加上建议于 WKUP 脚上增加 RC 滤波的描述 2. 修改温度传感器误差为±5°C 3. 增加电源方案描述
	2023-03-29	I2C 模块增加注意事项 3
V1.4	2023-05-09	1. 勘误 Bootloader 为 Bootrom 2. 修改 I2C 模块，注意事项 1/2/3 3. 新增"UART-BOOT 刻录接口"章节
V1.5	2023-06-06	修改 ADC 模块，注意事项 5
	2023-07-13	修改 I2C 模块，增加 I2C NBYTEs 长度的使用限制
V1.6	2024-09-11	FLASH 模块小节补充 FLASH 编程注意事项
V1.7	2025-01-08	最小系统电路中增加对 VDD 电压波动范围的限制
V1.8	2025-12-02	1. 完善“低功耗唤醒流程”章节中关于清标识的描述

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 软件开发注意项目

1.1 开发环境

推荐用户使用 Keil5 、IAR8.11 进行固件开发。由于 Keil4 不支持 PACK 机制，故不推荐用户使用 Keil4。

1.2 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，适合大部分使用者。

MD

基本上只提供寄存器位域级别的“读”、“写”接口，适合对晶元底层较为熟悉的使用者。

对于一些复杂模块，如 USB，由于用户独立配置相关寄存器实现对应功能较为繁琐，故这些模块仅提供 ALD 库，不再提供 MD 库。 如果用户对速度不是要求非常严格，一般情况下推荐使用者使用 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。

1.3 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。FC、IWDT、GPIO 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。

1.3.1 FC写保护

闪存控制器 FC，FC_CMD 和 FC_CTL.OPRLD 寄存器，写访问操作是受保护的状态。用户无法直接对闪存进行编程与擦除，为了避免闪存数据被意外擦除或修改。若用户需对闪存执行编程或擦除功能时，必须先进行解锁流程。其解锁流程，必需连续输入 2 组解锁密钥，若输入错误或连续输入超过 2 组密钥时，解锁失效，寄存器依旧保持上锁状态。当完成解锁流程后，用户可经由 FC_STA.CMDULK 位来判定是否解锁成功，当此位状态为 1 时，表示成功解锁，可进行寄存器写访问。

解锁流程如下所示：

1. 检查 FC_STA.CMDULK 为 0，属于锁定状态
2. 对 FC_UL 填入第一组密钥 0x00112233
3. 对 FC_UL 填入第二组密钥 0x55667788
4. 检查 FC_STA. CMDULK 为 1，确认解锁成功

当用户完成闪存的编程与擦除操作后，建议将锁定功能恢复，对 FC_UL 填入非密钥值(例如 0x00000000)重新上锁，避免闪存数据被意外擦除或是覆盖。

1. 3. 2 IWDT写保护

IWDT 寄存器，IWDT_BKPR、IWDT_BKRLR 和 IWDT_BKWINR，写访问操作是受保护的状态。解锁方式是对 IWDT_BKKR 写入 0x00005555 解锁码，即可解除写保护功能。若对 IWDT_BKKR 写入非密钥值，重新启动寄存器写访问保护机制。 由于 IWDT 寄存器，存放在备用寄存器中，当对其寄存器写访问时，需等待 33 个 PCLK 时钟更新才完成，可读取 IWDT_BKFR.BUSY 位判别寄存器是否完成更新。

1. 3. 3 GPIO写保护

透过 GPIOx_LCK 寄存器，经由 IO 锁定流程，可锁定寄存器写访问功能，包括 GPIOx_MOD、GPIOx_OT、GPIOx_PUD、GPIOx_AFL 和 GPIOx_AFH 寄存器。

IO 锁定流程，需操作 GPIOx_LCK 寄存器，写 2 次相同数值，每次需 32 位数值写入，而 GPIOx_LCK[31:16]必须是 GPIOx_LCK[15:0]取反值。读去判别 GPIOx_LCK.LCKK 位为 1，锁定流程正确且开启 IO 锁定功能，此功能开启后无法取消，只允许系统复位清除。

1. 4 写 1 清零寄存器

中断标志寄存器都是用“写 1 清零(C_W1)”的方式来操作。对于“写 1 清零(C_W1)”的寄存器，不可使用“读-修改-写”的方式来进行“写 1 清零(C_W1)”，否则会引起标志误清，进而产生漏中断的后果。

第2章 系统控制

2.1 系统时钟 RCU

系统时钟(SYSCLK)提供以下来源：

- HRC (High Speed Internal RC Oscillator) - 内部高速 4 MHz RC 振荡器
- HOSC (High Speed External Oscillator) - 外部高速时钟振荡器
- PLL (Phase Locked Loop) - 锁相环
- HRC48 (High Speed Internal 48 MHz RC Oscillator) - 内部高速 48 MHz RC 振荡器

系统重置后，默认使用内部 4MHz HRC 作为系统时钟。经由 RCU_CFG.SW 寄存器，切换不同的系统时钟源；切换系统时钟源之前，必须确保目标时钟源开启并且时钟已稳定，此时才可更换系统时钟。读取 RCU_CFG.SWS 寄存器，可以确认当前系统时钟源是否已更换完成。提醒注意，无法透过 RCU_CON 寄存器，关闭当前的系统时钟源；当误操作时，RCU_CON 寄存器是无反应的。另一方面，当 PLL 选用于系统时钟源时，PLL 所采用的参考时钟 HRC、HOSC 或 HRC48，同样无法被 RCU_CON 寄存器关闭。

每种时钟源不使用时，皆可独立设置开启或关闭，进而优化系统功率消耗；并提供分频电路，让用户依据应用场景与功耗需求，配置 AHB 与 APB 操作频率，AHB 与 APB 最高可配置为 72MHz。所有周边外设时钟状态，都依据所属的总线(Bus)时钟而定，如 AHB 总线时钟为 HCLK，APB 总线时钟为 PCLK。

除了下列几个特殊区块：

- I2SCLK 时钟
- USBCLK 时钟
- RTCCLK 时钟
- ADCCLK 时钟
- IWDTCLK 时钟
- STCLK 时钟

其他的时钟源:

- LRC (Low Speed Internal RC Oscillator) - 内部低速 32 kHz RC 振荡器，提供独立看门狗(IWDT)与实时时钟(RTC)使用
- LOSC (Low Speed External Oscillator) - 外部低速 32.768 kHz 时钟振荡器，提供实时时钟(RTC)与时钟同步单元(CSU)使用

2.1.1 HOSC时钟 (HOSCCLK)

外部时钟源 (HOSC Bypass)

由外部提供有效的时钟源，设定寄存器 RCU_CON.HOSCBYP 与 RCU_CON.HOSCON 2 个位启用 HOSC Bypass 功能；外部时钟源必须由 HOSCI 引脚输入。

外部石英晶体振荡器 (HOSC Crystal)

配置 RCU_CON.HOSCON 位控制 HOSC 开启与关闭；HOSC 时钟状态可藉由 RCU_CON.HOSCRDY 标志位确认，当 HOSC 时钟稳定时 标志位将被硬件自动配置为高电位。

注意事项 1: 当开启 RCU_CON.HOSCON 后，等待 RCU_CON.HOSCRDY 标志位时，必须加入 1ms 超时机制，当这段时间等不到 RCU_CON.HOSCRDY=1 时，必须将 RCU_CON.HOSCON 关闭后再开启，直到标志位 RCU_CON.HOSCRDY=1 为止。

2. 1. 2 HRC时钟 (HRCCLK)

HRC 时钟信号是由内部高速 4 MHz RC 振荡器产生。配置 RCU_CON.HRCON 位控制 HRC 开启与关闭；HRC 时钟状态可藉由 RCU_CON.HRCRDY 标志位确认，当 HRC 时钟稳定时，标志位将被硬件自动配置为高电位。

2. 1. 3 PLL时钟 (PLL0CLK)

此锁相回路(Phase-Locked Loop，PLL)是非整数锁相回路(Fractional-N PLL)，其输入时钟频率应介于 3~16 MHz，透过 RCU_CFG.PLLSRC 选择 HRC、HOSC 或 HRC48 作为输入时钟源，再搭配 RCU_CFG.PREDIV 预分频将频率除到范围内，建议 PLL 输入时钟频率维持在 4MHz 左右；输出时钟频率范围介于 4~72 MHz，输出时钟频率是压控振荡器(Voltage Controlled Oscillator，VCO)频率除以一个可变的倍率 RCU_PLL0.FM，FM 根据输出时钟频率区间调整分别是 8、16、32 或是 64，而压控振荡器的频率为输入时钟频率乘以一个可变的倍率，该倍率可以是整数也可是非整数，由 FN 与 FK 控制，FN 表示整数的倍率，FK 表示小数的倍率，其公式如下。配置 RCU_CON.PLL0ON 位控制 PLL0 开启与关闭；PLL0 时钟状态可藉由 RCU_CON.PLL0RDY 标志位确认，当 PLL0 时钟稳定时，标志位将被硬件自动配置为高电位。

$$f_{VCO} = f_{PLL0IN} \times \left(FN + \frac{FK}{2^{19}}\right),\ 256MHz \leqslant f_{VCO} \leqslant 576MHz$$

$$f_{PLL0} = \frac{f_{VCO}}{FM},\ FM = 8,16,32,64$$

PLL 的配置(包括输入时钟源的选择、预分频和倍率设定)必须在 PLL 开启之前完成，开启 PLL 后便无法修改配置，必须关闭 PLL 才可以重新配置。修改 PLL 配置的流程如下:

- 关闭 PLL0，设定 RCU_CON.PLLON=0
- 确认 RCU_CON.PLL0RDY=0 后，PLL0 才完成关闭的流程
- 配置用户需求的修正
- 再次开启 PLL0，设定 RCU_CON.PLLON=1
- 等待 RCU_CON.PLL0RDY=1，等待 PLL0 输出时钟稳定

2. 1. 4 HRC48 时钟 (HRC48CLK)

配置 RCU_CON.HRC48ON 位控制 HRC48 开启与关闭；HRC48 时钟状态可藉由 RCU_CON.HRC48RDY 标志位确认，当 HRC48 时钟稳定时 标志位将被硬件自动配置为高电位。

HRC48 时钟除了用于系统时钟源之外，亦可用于 USB 或 I2S 应用。在 USB 的应用中，HRC48 RC

时钟振荡器搭配同步单元(CSU)，提供 USB 外设高精准度的时钟；同步单元 CSU 可使用 USB 的 SOF 封包、LOSC 或外部输入信号，自动地微调整 HRC48 振荡器频率。

2. 1. 5 LOSC时钟 (LOSCCLK)

外部时钟源 (LOSC Bypass)

由外部提供有效的时钟源，并设定寄存器 RCU_LCON.LOSCBYP 与 RCU_LCON.LOSCON 2 个位启用 LOSC Bypass 功能；外部时钟源必须由 LOSCI 引脚输入。

外部石英晶体振荡器 (LOSC Crystal)

LOSC 时钟是由外部 32.768kHz 石英晶体振荡器或一个稳定低速时钟源提供，可用于实时时钟(RTC)计数时间与日历。配置 RCU_LCON.LOSCON 位控制 LOSC 开启与关闭；LOSC 时钟状态可藉由 RCU_LCON.LOSCRDY 标志位确认，当 LOSC 时钟稳定时 标志位将被硬件自动配置为高电位。

2. 1. 6 LRC时钟 (LRCCLK)

LRC 内部低速 RC 振荡器，是一个低功耗时钟源，并可以在低功耗模式下让外设独立看门狗(IWDT)和实时时钟(RTC)持续运行；时钟频率大约为 32kHz。配置 RCU_LCON.LRCON 位控制 LRC 开启与关闭。LRC 时钟状态可藉由 RCU_LCON.LRCRDY 标志位确认，当 LRC 时钟稳定时 标志位将被硬件自动配置为高电位。

2. 1. 7 时钟安全系统 (CSS)

HOSC 时钟安全系统(Clock Security System，CSS)是一个系统安全机制；当系统时钟选用 HOSC 时，为了防止 HOSC 发生故障，而导致系统死机，避免应用陷入不安全的状态。可以透过 **RCU_CON**.CSSON 开启或关闭时钟安全系统，只有当 HOSC 时钟开启且信号稳定才能开启时钟安全系统，并且在侦测到 HOSC 停止时关闭，并自动切换成 HRC。

2. 1. 8 特殊区块IP

以下这些特殊区块只能支持特定时钟源

Source / IP	AHB	APB	HOSC	PLL0	HRC48	LOSC	LRC
ADC 时钟(ADCCLK)		◉					
RTC 时钟(RTCCLK)						◉	◉
I2S 时钟(I2SCLK)			◉	◉	◉		
USB 时钟(USBCLK)				◉	◉		
IWDT 时钟(IWDTCLK)							◉
System Tick 时钟(STCLK)	◉						

表 2-1 Clock Source

2. 1. 9 MCO时钟 (MCOCLK)

微控制器时钟输出功能允许将时钟信号输出到 MCO，除了提供检查以外也可当作其他装置或模块的输入时钟源。透过配置RCU_CFG.MSW选择输出时钟，根据需求可以配置RCU_CFG.MPRE将微控制器时钟输出进行分频，可以分频 2、4、8、16、32、64 或 128 倍。可以选择各种时钟输出。

2. 2 低功耗唤醒流程

注意事项 1: 反复进入低功耗模式，唤醒后需增加等待时间

低功耗模式(STOP/STANDBY0/STANDBY1/SHUTDOWN)，当经由唤醒引脚或是唤醒事件唤醒后，必须清除唤醒标志位 SYSCFG_WKSR.WKCLR，检查唤醒标志位 SYSCFG_WKSR.FLAG 已清除，必须等待 125us 之后，才能再次进入低功耗休眠唤醒流程。建议在进入低功耗模式之前，先设定 SYSCFG_WKSR.WKCLR 来清除唤醒标志位 SYSCFG_WKSR.FLAG，以确保在进入低功耗模式时，该标志位为 0。

注意事项 2: 开启低功耗流程时，需等待系统执行完开启流程后才可送出唤醒讯号。依据不同等级的低功耗模式，所需要等待的时间不同。当使用外部唤醒引脚(WKUPx)唤醒时，建议在使用的唤醒引脚加上滤波电路，滤波电路可参考下表进行配置。

低功号模式	所需等待时间(us)	RC 电路建议搭配
STOP	123	0.1uF 电容搭配 1.1KΩ 电阻
STANDBY0	401	0.1uF 电容搭配 3.6KΩ 电阻
STANDBY1	185	0.1uF 电容搭配 1.6KΩ 电阻
SHUTDOWN	185	0.1uF 电容搭配 1.6KΩ 电阻

表 2-2 完全进入低功耗模式所需时间

2. 3 FLASH模块

注意事项 1: FLASH 读保护(RP)配置字

- 配置 RP 为 level0 等级，为默认不保护
- 配置 RP 为 level1 保护后，将无法进行程序下载，但可调试；可修改 RP 等级；如果降低 RP 等级为 level0，将触发 Flash 全擦除。
- 配置 RP 为 level2 保护后，将无法进行程序下载与调试，不可修改 RP 等级。

注意事项 2: FLASH 等待时钟数(Wait Cycle)

系统频率变化时，必须确保有足够的 FLASH 等待周期 FC_CTL.WAIT，支持最多 3 个等待周期，如下表。

系统时钟	FC_CTL.WAIT
>72Mhz	3
72Mhz≥ f_{SYSCLK} >48Mhz	2
48Mhz≥ f_{SYSCLK} >24Mhz	1

24Mhz≥ f_{SYSCLK}	0(无等待)

表 2-3 FLASH 等待时钟数(Wait Cycle)

注意事项 3：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。

2.4 配置字

注意事项 1: 配置字修改与重置

当使用 ESLink II(miin/Pro)，修改芯片配置字之后，需要断电后重新上电，才能正常调试。

注意事项 2: BOR 配置字

BOR 电压等级(Level)，必须设定高于 MCU 工作电压；电压等级数值，请参考 ES32F0238_Datasheet.pdf

注意事项 3: IWDT 配置字

IWDT 配置字开启后，软件无法关闭，预设 1 秒发生 IWDT 复位，支持软件修改复位时间。

第3章 外设

3. 1 GPIO模块

注意事项 1: 未使用的 GPIO 埠处理

系统中未使用的 GPIO 引脚，建议设置 GPIOx_MOD 为模拟模式(11b)并悬空，以避免漏电情形。若设置为输入模式，需设置 GPIOx_PUD，上拉电阻接到电源或加下拉电阻接到地。

注意事项 2: GPIO 端口复用流程

任何外设要设置 IO 复用功能时，先配置 GPIOx_AFx 寄存器，选择 AFx 复用种类，再将 GPIOx_MOD 配置为复用功能模式(10b)。

注意事项 3: 低功耗模式唤醒引脚

当系统进入低功耗模式时(STOP/STANDBY0/STANDBY1/SHUTDOWN)，PB00~PB07 属于唤醒引脚，AFx 复用种类要选择 WKUPx，另外开启唤醒功能 SYSCFG_WKUP.WKEN，并选择唤醒事件 SYSCFG_WKUP.WKEG 是上升沿或下降沿模式。

注意事项 4: 低功耗模式引脚维持现状

当系统进入低功耗模式时(STANDBY0/STANDBY1/SHUTDOWN)，PB00~PB07 可维持进入低功耗模式前的引脚设定状态，如 GPIOx_MOD 输入/输出模式、GPIOx_PUD 上拉和下拉、GPIOx_OT 推挽或开漏和 GPIOx_DS 输出驱动，仅维持设定状态，不维持寄存器数值。

3. 2 DMA模块

注意事项 1: DMA 配置信道控制数据结构(DMA_PRI_CH0x_x 与 DMA_ALT_CH0x_x)的方法

当配置 DMA 信道控制数据寄存器时，写入配置值后，需确认读取数值与写入是否一致，不同时需重复写入读取确认，直到相同为止，此设定值才真正被写入。

注意事项 2: 当 DMA 运行中，CPU 对同一个外设操作时的方法 当 DMA 运行中，CPU 要对同一个外设区块进行操作前，必须先确认当前 DMA 信道搬移是否已完成，再进行 CPU 的操作。DMA 信道搬移是否已完成，可检查 DMA_CHENSET 寄存器信道状态。

3.3 UART模块

注意事项 1: 自动波特率

若发生侦测波特率超时(UART_RIF.ABTO)，波特率开关会自动清除(UART_MCON.ABREN)，建议启用自动波特率侦测时，同时启用重复侦测自动波特率(UART_MCON.ABRREPT)功能。若自动波特率侦测成功，但数据非预期时，用户需重新使能自动波特率侦测(UART_MCON.ABREN)功能。

注意事项 2: 发送空中断处理方式

UART TX 数据传输缓存，包含 UART_TXDATA 寄存器与 TX 位移寄存器；预设 UART_TXDATA 为空，空中断发生，当写数据进入 UART_TXDATA 后，数据立即被转移到 TX 位移寄存器，此时 UART_TXDATA 为空，再次发生空中断。默认空中断与第一笔数据的空中断，过于紧凑，因此中断处理流程中，优先清除中断再填写 UART_TXDATA。

3.4 I2C模块

注意事项 1: 当总线 SDA 遇到异常下拉的波形时，可能出现传输状态错误

发生条件为，当 I2C 配置为主机，I2C_CON2.START 写 1 时，从机输出 SDA 低电平，I2C 可能会进入从机接收模式；当从机输出不受控的 SDA 低电平时，请系统流程中加入软件超时机制，在 I2C_CON2.START 写后开始计时，若 I2C 通讯超时，则复位 I2C。复位 I2C 的方式为 I2C_CON1.PE 写 0，重新配置 I2C，然后 I2C_CON1.PE 写 1

注意事项 2: 最大传输 NBYTE

当重载模式关闭时，I2C 支持最大 2^{16} -1 =65535 Bytes 的传输数量，分别配置 I2C_CON1.NBYTES 与 I2C_CON2.NBYTES 寄存器。

当重载模式开启时，最大传输 NBYTE 为:

- 传送模式下，NBYTE 最大设定值为 65535，当传输数量需求大于 65535 时，在传输前配置 I2C_CON2.RELOAD 位，当第一笔 65535 Bytes 传输完后，SCL 将被拉低，此时可重新填写第二笔传输的 NBYTES 数量；当不使用 NBYTES 重载模式时，需将 I2C_CON2.RELOAD 清除，如果开启 I2C_CON2.AUTOEND 功能，I2C 会在 I2C_CON2.RELOAD 清除后，自动发送 STOP 讯号。
- 接收模式下，NBYTE 最大设定值为 255，当传输数量需求大于 255 时，在传输前配置 I2C_CON2.RELOAD 位，当第一笔 255 Bytes 传输完后，SCL 将被拉低，此时可重新填写第二笔传输的 NBYTES 数量；当不使用 NBYTES 重载模式时，需将 I2C_CON2.RELOAD 清除，如果开启 I2C_CON2.AUTOEND 功能，I2C 会在 I2C_CON2.RELOAD 清除后，自动发送 STOP 讯号。

注意事项 3: I2C 控制寄存器 2 (I2Cx_CON2)訪問限制

I2C 控制寄存器 2 (I2Cx_CON2)，必须按字(32 位)訪問。

注意事项 4: 发送空中断处理方式

I2C TX 数据传输缓存，包含 I2C_TXDATA 寄存器与 TX 位移寄存器；预设 I2C_TXDATA 为空，空中断发生，当写数据进入 I2C_TXDATA 后，数据立即被转移到 TX 位移寄存器，此时 I2C_TXDATA 为空，再次发生空中断。默认空中断与第一笔数据的空中断，过于紧凑，因此中断处理流程中，优先清除中断再填写 I2C_TXDATA。

3.5 SPI模块

注意事项 1: 发送 FIFO 缓存中断处理方式

SPI 发送 FIFO 中断标志为分为两种，发送 FIFO 缓存空中断(SPI_IFM.TXE)与发送 FIFO 缓存低于阀值中断(SPI_IFM.TXTH)。当进入中断处理流程时，务必优先清除中断再填写 SPI_DATA。根据不同的应用，为了避免填写 SPI_DATA 时，发生 FIFO 溢出的错误操作，提供以下两种方式检查:

- 方法 1: 确认 FIFO 剩余空间，读取 SPI_STAT.TXFLV 寄存器，表示 FIFO 已填入数据数量(Bytes)，可判断再填入多少数据。
- 方法 2: 确认 FIFO 是否已满，读取 SPI_STAT.TXF 寄存器，表示 FIFO 已满，不可在填入数据。

注意事项 2: 接收 FIFO 缓存中断处理方式

SPI 接收 FIFO 中断标志为分为三种，接收 FIFO 缓存非空中断(SPI_IFM.RXNE)、接收 FIFO 缓存满中断(SPI_IFM.RXF)与接收 FIFO 缓存超过阀值中断(SPI_IFM.RXTH)。FIFO 功能主要为了提升 SPI 传输效率，因此大多使用 RXF 与 RXTH 中断事件，效率比较显着；当进入中断处理流程时，务必优先读取(搬移)SPI_DATA，避免 FIFO 满溢造成数据丢失，提供以下两种方式检查:

- 方法 1: 确认 FIFO 缓存空间，读取 SPI_STAT.RXFLV 寄存器，表示 FIFO 接收 Bytes 数量，可判断读取多少数据。
- 方法 2: 确认 FIFO 是否已空，读取 SPI_DATA 数据，判断 SPI_STAT.RXNE 寄存器为 0，表示 FIFO 已空，数据已读取完毕。

注意事项 3: 从机模式下，发送数据操作方法

SPI 配置为从机模式时，填写 SPI_DATA 数据的依据，建议使用发送 FIFO 缓存低于阀值状态标志位(SPI_STAT.TXTH)或中断(SPI_IFM.TXTH)。

3.6 RTC模块

注意事项 1: 设定日期与开启流程

开启 RTC 计数之前，先以 BCD 格式设定日期(RTC_CAL)与时间(RTC_TIME)；由 RTC_CTRL.CKSEL 选择时钟来源为 LOSC 或 LRC；设置分频系数 RTC_CTRL.PSCALE 与 RTC_CTRL.SCALE 来产生准确 1 秒(1MHz)计数信号。寄存器默认值为 SCALE=0xFF，PSCALE=0x7F，使用外部 32.768kHz 晶振时，频率为 32768/(PSCALE*SCALE)=1Hz。完成基本设定后，方可开启 RTC(RTC_CTRL.RTCEN)，期间必须关闭 RTC，才可修改时间与日期寄存器。

注意事项 2: 读取日期与时间

由于RTC属于备份寄存器区，系统读取时间或日期前，可以检查RTC_STA.SYNDONE信号为1，确保日期与时间已更新。

注意事项 3: 校准与时钟补偿

由于 LOSC 晶振非准确的 32.768 kHz，因此经由 RTC 频率补偿方法，以确保产生准确的 1 秒的 RTC 信号。计算补偿数值的方式有以下 2 种：

- 方法 1: 不开启 RTC 补偿功能，直接纪录 RTC 计时数日后，与标准时间的误差值(ppm)。
- 方法 2: 使用 RTC 1Hz 信号触发 Timer 计数方式，推算出 RTC 1Hz 的误差值(ppm)；此方法所推算出的误差值，将受限于系统时钟精准度的影响，但取得误差值的时间较方法 1 来的快许多。举例: 当系统频率为 8MHz，利用 RTC 1Hz 信号触发 Timer 计数，Timer 每秒所计数的数值会接近于 8000000，因此 Timer 计数数值与理想值(8000000)相减，即可推算出误差值(ppm)。

当取得误差值(ppm)数据后，将误差值先乘上 32768，再除上最小精度 1/4096，即可得到 RTC 的校准数值，将数值填入 RTC_CALIB.CALIB；依据误查值的正负值，设定 RTC_CALIB.MODE 选择正校准或负校准模式，最后开启硬件自动校准功能 RTC_CALIB.CALIBEN。

3.7 ADC模块

注意事项 1: 配置 ADC 预分频器

修改 ADC 预分频(ADC_SMPT1.CKDIV)配置时，必须确保 ADC 关闭的状态(ADC_CON.ADCEN)，才可修改预分频。

注意事项 2: ADC 稳定时间

开启 ADC 时，可由寄存器 ADC_RIF.ARDY 确认 ADC 是否稳定就绪；但当系统频率 APB_CLK 时钟高于 24MHz 时，除了确认 ADC_RIF.ARDY 标志位，必须保证等待时间大于 1.75us，ADC 才算是稳定。

注意事项 3: ADC 校准补偿

ADC 校准补偿方式，请调用校准函式 md_adc_calibration；由于校准补偿函式，将会进行 ADC 补偿与设定重置，建议 ADC 采样之前调用此函式。

```
ErrorStatus md_adc_calibration(ADC_TypeDef  *ADCx, md_adc_initial  *ADC_InitStruct)
{
    uint32_t          cal_value = 0;

    md_fc_read_info(ADC_CALIBRATION_ADDR, &cal_value);

    if (cal_value == 0xFFFFFFFF)
    {
        return md_adc_software_calibration(ADCx, ADC_InitStruct);
    }
    else
    {
        return md_adc_optionbyte_calibration(ADCx, ADC_InitStruct);
    }
return ERROR;
}
```

注意事项 4: ADC 转换电压范围

系统应用时，须注意 ADC 输入电压范围，是否符合电器特性规范 V_{AIN}。

注意事项 5: 使用内部参考电压 VREFINT，推算 ADC 参考电压 V_{REFP} 实际电压

由于 VDDA 与 VREFP 连接在一起，ADC 理论上满刻度电压为 5V(V_{DDA})，但依据系统应用不同，V_{REFP} 电压有可能不是完美的 5V 电压；因此必须使用内部参考电压 VREFINT(ADIN17)，来推算实际 VREFP 电压数值；另外设置 ADIN17 采样周期数(ADC_SMPT5.CHT17)，此周期数为 5T(ADC_CLK=12MHz, 0.543us)。

$$V_{REFP} = 5V \times \frac{ADC_{VREFINT\_CAL}}{ADC_{VREFINT}}$$

- $ADC_{VREFINT\_CAL}$: 读取配置字 SYSCFG_ADCVREF 数值
- $ADC_{VREFINT}$: 当下 ADC 取样 VREFINT(ADIN17)所得 ADC 数值

注 1: $ADC_{VREFINT\_CAL}$，批量生产是在 VDDA=5V 的状态下，ADC(校准后)取样 VREFINT 电压数值，存放在配置字中，因此每颗芯片数值不同。

注意事项 6: 转换 ADC 量测信道输入实际电压值

由于 ADC 满刻度的电压依据是跟随 VDDA 变化，若要取输入信道的实际电压值，必须先完成注意事项 5，取得 V_{REFP} 实际电压，再套用以下公式计算出输入信道的绝对电压。

$$V_{AIN}=\frac{V_{REFP}}{FULL\_SCALE}\times ADC_{AIN}$$

- FULL_SCALE : ADC 输出最大可表示的数值，12 位分辨率，FULL_SCALE 为 4095。

3.8 温度感测模块

注意事项 1: 温度感测配置

温度传感器输出电压 V_{TS}，是连接到 ADC 的信道 16(ADIN16)，因此须先配置 ADC 相关设定后，才可经由 ADC 取得温度数值(ADC_{TS})；注意设置 ADIN16 采样周期数(ADC_SMPT5.CHT16)，此周期数为 51T(ADC_CLK=12MHz, 4.375us)；ADC 量测温度之前，ADC 必须经过校准。经由以下公式可推算出 V_{TS} 电压值:

$$\text{Voltage Temperarute: } V_{TS}=ADC_{TS}\times\frac{V_{REFP}}{FULL\_SCALE}$$

注意事项 2: 实际温度推算

当 ADC 取得温度传感器电压数值(ADC_{TS})时，经由公式推算取得实际温度(Temperarute,°C)。计算前，必须取得 2 个参数，分别为温度传感器 30 度电压值(V_{30})与温度平均斜率(Avg_Slope)；V_{30} 的 ADC 数值(ADC_{30})存放在配置字中(SYSCFG_ADCTEMP.ADCTEMPL)，Avg_Slope 固定为 3.75 mV/℃(参考数据手册)。将数值套用到以下公式，可推算出实际温度，温度误差在±5°C 以内。

- 已知温度 30 度的电压值

$$\text{Voltage 30°C: } V_{30}=ADC_{30}\times\frac{V_{DDA}}{FULL\_SCALE}$$

- 现在温度的电压值

$$\text{Voltage in°C: } V_{TS}\text{ (参考注意事项 1)}$$

- 实际温度计算

$$\text{Temperarute(in°C)}=\frac{(V_{TS}-V_{30})}{Avg\_Slope}\times 1000+30°C$$

注 1: 温度平均斜率(Avg_Slope)，经由量测温度曲线(-10~70°C)100 个样本，所得到平均斜率，单为位 mV/°C，请参考数据手册。

注 2: 温度 30 度时 ADC 数值(ADC_{30})，批量生产控温于 30 度(±5°C)时，ADC(校准后)取样数值，存放在配置字中，每颗芯片数值不同。

注 3: 温度 30 度时 ADC 数值(ADC_{30})，是在 VDDA=5V 的状态下，所得到的 ADC 取样值；若应用时，VREFP 不是 5V 条件，需要另外计算电压比例。

第4章　系统电路与版图注意事项

4. 1　最小系统电路

4. 1. 1　LQFP64 封装芯片最小系统电路

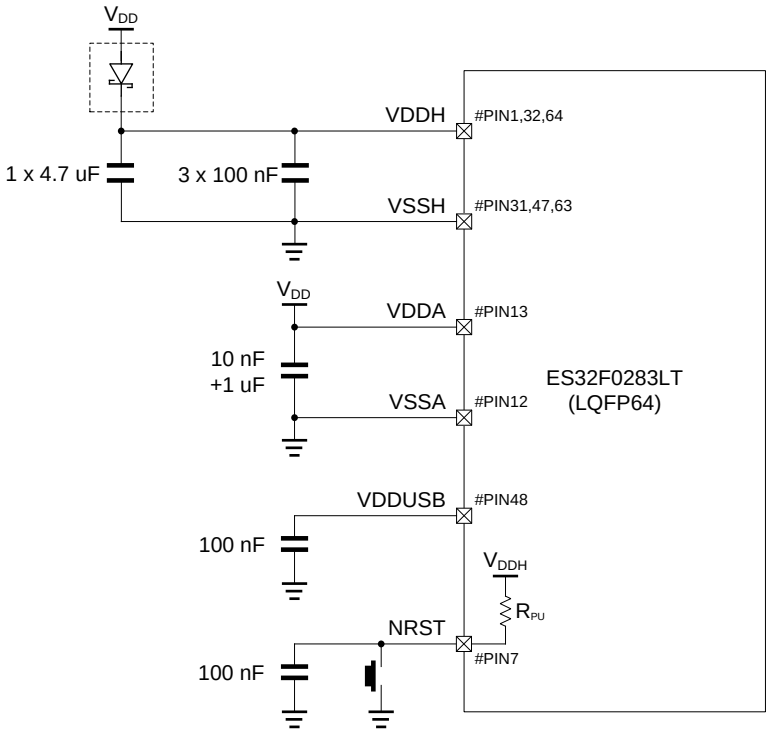

图 4-1　LQFP64 封装芯片最小系统电路

注 1: 芯片电源请务必依照图中建议进行配置，才能保证芯片电源在任何应用场景下都能够稳定。

注 2: 每一组电源必须连接陶瓷耦合电容(如图式)；这些电容必须尽可能靠近芯片的相应管脚，才能保证芯片的运行性能。

注 3: 需要为芯片提供稳定可靠的供电电源，当 VDD 电压波动时，电压跌落速率需慢于 200us/V，或者电压波动范围需小于±0.2V。如果 VDD 电源域有其他负载开关导致电压波动超出范围，可在 VDD 连接到 4.7μF 电解电容之前串联一个肖特基二极管。

注 4: NRST 引脚采用 RC 复位，支持内部上拉(上拉电阻值，请参考数据手册电器性章节)，电容采用 100nF。

注 5: VDDUSB 电源引脚，需连接稳压电容，预设此引脚与 VDDH 相连，开启 LDOUSB 内部电源稳压器后(SYSCFG_PWR.LDOUSBEN=1)，VDDUSB 引脚输出 3.3V 电压。

4. 1. 2 LQFP48 封装芯片最小系统电路

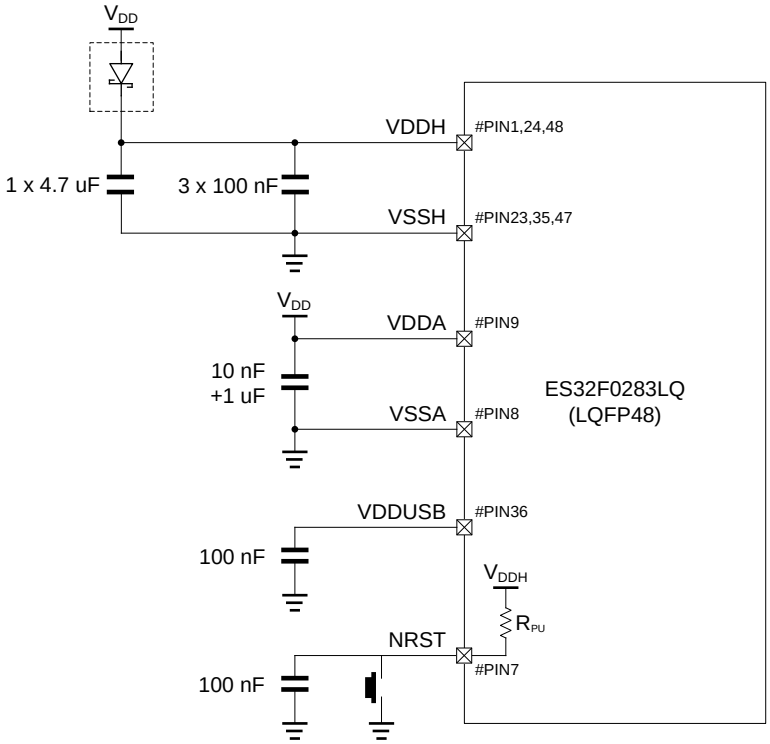

图 4-2 LQFP48 封装芯片最小系统电路

注 1: 芯片电源请务必依照图中建议进行配置，才能保证芯片电源在任何应用场景下都能够稳定。

注 2: 每一组电源必须连接陶瓷耦合电容(如图式)；这些电容必须尽可能靠近芯片的相应管脚，才能保证芯片的运行性能。

注 3: 需要为芯片提供稳定可靠的供电电源，当 VDD 电压波动时，电压跌落速率需慢于 200us/V，或者电压波动范围需小于±0.2V。如果 VDD 电源域有其他负载开关导致电压波动超出范围，可在 VDD 连接到 4.7μF 电解电容之前串联一个肖特基二极管。

注 4: NRST 引脚采用 RC 复位，支持内部上拉(上拉电阻值，请参考数据手册电器性章节)，电容采用 100nF。

注 5: VDDUSB 电源引脚，需连接稳压电容，预设此引脚与 VDDH 相连，开启 LDOUSB 内部电源稳压器后(SYSCFG_PWR.LDOUSBEN=1)，VDDUSB 引脚输出 3.3V 电压。

4.2 外围电路

4.2.1 外部晶振HOSC与LOSC

注意事项 1: 外部陶瓷电容

电容建议数值，请参阅 ES32F0283 数据手册，电器特性章节。

注意事项 2: 版图参考原则

外部晶振应尽量靠近 IC，提高稳定度。

4.2.2 USB

注意事项 1: 版图参考原则

USB DP/DM 为差分信号线传输数字信号，应遵循以下原则，以达到最佳的传输效果。

- 尽量缩短差分线路走线距离。
- 优先绘制差分线，一对差分在线尽量不要超过两对过孔（过孔会增加线路的寄生电感，从而影响线路的信号完整性），且需对称放置。
- 对称平行走线，保证两根线紧耦合，避免 90° 走线，弧形或 45° 是较好的走线方式。
- 差分线线长需匹配，长度差控制在 5mil 以内；线长不匹配，易发生时序偏移，或引入共模干扰，降低信号质量。
- 为了减少串扰，在空间允许的情况下，其他信号网络及地离差分线的间距至少 20mil，覆地与差分线的距离过近将对差分线的阻抗产生影响。

4.2.3 ESLINK II与调适接口

注意事项 1: 电路建议

PA14 与 PA13 引脚预设为 ESLINK II 与调适接口，PA14 为 SWCLK 默认为输入下拉模式，PA13 为 SWDIO 默认为输入上拉模式。当系统应用时，将此接口配置为其他 IO 功能时，如需维持 ESLINK II 与调适接口的功能，务必预留 NRST 引脚功能，才能在 NRST=0 时，恢复 SWCLK 与 SWDIO 功能。

注意事项 2:

Socket 进行烧录时，芯片电源方案务必符合图 4-1 与图 4-2 的电源方案要求。

4.2.4 UART-BOOT刻录接口

注意事项 1: 开机程序默认选择

系统开机流程中，提供 Bootrom 与 Main Flash 两种选择，而 ES32F0283 预设为 Main Flash；此配置可透过修改配置字更换默认值。

注意事项 2: UART-BOOT 接口选用

当配置字 BOOT BYPASS 选项为 Bootrom，表示系统开机从 Bootrom 开始，PB06 与 PB07 引脚预设为 UART1 通信接口，PB06 为 UART1_TX 输出，PB07 为 UART1_RX 输入；由 UART 通信接口，搭配 UART-BOOT 刻录工具下载或更新程序。

当配置字 BOOT BYPASS 选项为 Main Flash，表示系统开机直接转跳至 Main 区执行，PB06 与 PB07 引脚预设为 GPIO，不具备 UART-BOOT 刻录功能。