前言

此应用笔记说明本芯片使用时所需的规范，包括各周边使用时的注意事项及硬件设计时所要求的元件规格。

目录

1 概述 7
1.1 开发环境 7
1.2 库函数选择 7
1.3 寄存器写保护 7
1.3.1 FC 写保护 7
1.3.2 IWDT 写保护 8
1.3.3 GPIO 写保护 8
1.4 寄存器读保护 8
1.4.1 IWDT 读保护 8
1.5 写 1 清零(C_W1)寄存器 8
2 系统控制 9
2.1 系统时钟 RCU 9
2.1.1 HOSC 时钟（HOSCCLK) 9
2.1.2 HRC 时钟（HRCCLK) 9
2.1.3 PLL 时钟（PLLCLK) 10
2.1.4 LRC 时钟（LRCCLK) 10
2.2 时钟安全系统（CSS) 10
2.3 特殊区块 IP 11
2.4 MCO 时钟（MCOCLK) 11
2.5 低功耗唤醒流程 12
2.6 配置字 12
3 外设 13
3.1 CALC 模块 13
3.1.1 概要说明 13
3.1.2 常用功能描述 13
3.1.3 配置说明 13
3.1.4 应用限制 13
3.1.5 实际应用建议 13
3.2 CRC 模块 14
3.2.1 概要说明 14
3.2.2 常用功能描述 14
3.2.3 配置说明 14
3.2.4 应用限制 14
3.3 DMA 模块 14
3.3.1 概要说明 14
3.3.2 常用功能描述 14
3.3.3 配置说明 14
3.3.4 应用限制 15
3.3.5 实际应用建议 15
3.4 FLASH 模块 15
3.4.1 概要说明 15
3.4.2 常用功能描述 15
3.4.3 配置说明 15
3.4.4 读保护（RP） 15
3.4.5 写保护（WP） 15
3.4.6 用户代码读保护（UCRP） 15
3.4.7 等待周期设置 16
3.5 EXTI 模块 16

3.5.1 概要说明 16
3.5.2 常用功能描述 16
3.5.3 配置说明 16
3.5.4 应用限制 17
3.6 GPIO 模块 17
3.6.1 概要说明 17
3.6.2 常用功能描述 17
3.6.3 配置说明 17
3.6.4 应用限制 17
3.6.5 实际应用建议 18
3.7 UART 模块 18
3.7.1 概要说明 18
3.7.2 常用功能描述 18
3.7.3 配置说明 18
3.7.4 应用限制 18
3.7.5 中断处理建议 19
3.7.6 实际应用建议 19
3.8 I2C 模块 19
3.8.1 概要说明 19
3.8.2 常用功能描述 19
3.8.3 配置说明 19
3.8.4 应用限制 19
3.8.5 中断处理建议 20
3.9 SPI 模块 20
3.9.1 概要说明 20
3.9.2 常用功能描述 20
3.9.3 配置说明 20
3.9.4 中断处理 21
3.9.5 模式故障处理 21
3.9.6 结束条件 21
3.10 ADC 模块 22
3.10.1 概要说明 22
3.10.2 常用功能描述 22
3.10.3 配置说明 22
3.10.4 应用限制 22
3.10.5 VDD（V_{REFP}）计算 22
3.10.6 真实电压转换 23
3.10.7 温度测量 23
3.10.8 ADC 采样的时间限制 24
3.10.9 ADC 校准 24
3.10.10 校准注意事项 25
3.11 OPAMP 模块 26
3.11.1 概要说明 26
3.11.2 常用功能描述 26
3.11.3 配置说明 26
3.11.4 应用限制 26
3.11.5 低功耗行为 26
3.12 TIMER 模块 26
3.12.1 概要说明 26
3.12.2 常用功能描述 26

3.12.3 配置说明 26
3.12.4 应用限制 27
3.12.5 刹车功能 27
3.12.6 通道设置 27
3.12.7 锁定等级（写保护功能） 27
3.12.8 蜂鸣器控制 28
3.13 IWDT 模块 28
3.13.1 概要说明 28
3.13.2 常用功能描述 28
3.13.3 配置说明 28
3.13.4 超时时间计算 29
3.14 WWDT 模块 29
3.14.1 概要说明 29
3.14.2 常用功能描述 29
3.14.3 配置说明 29
3.14.4 超时时间计算 29
3.14.5 中断与复位行为 30
4 系统电路与版图注意事项 31
4.1 最小系统电路 31
4.1.1 LQFP48 封装芯片最小系统电路 31
4.1.2 QFN40 封装芯片最小系统电路 31
4.2 外围电路 32
4.2.1 外部晶振 HOSC 32
4.2.2 PTEM / PTEM-V1LITE 与调适接口 32
4.2.3 UART-BOOT 刻录接口 32
4.2.4 上下电斜率 32
5 文件版本纪录 33

表格目录

表 1. HOSC 时钟模式列表 9
表 2. 计算参数说明 10
表 3. Clock Source 11
表 4. 低功耗唤醒注意事项列表 12
表 5. 低功耗模式唤醒后系统执行状态栏表 12
表 6. 配置字使用注意事项列表 12
表 7. 读保护层级说明 15
表 8. FLASH 等待时钟数(Wait Cycle) 16
表 9. LOCKLVL 层级说明 27
表 10. 文件修订版本记录 33

图形目录

图 1. RP 设置状态图.....16
图 2. LQFP48 封装芯片最小系统电路.....31
图 3. QFN40 封装芯片最小系统电路.....31

1 概述

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

PT2E 系列芯片提供 2 种类型库函数 HAL 和 LL：

- HAL

  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(后续视需求开发)。

- LL

  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用户。

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防止被保护的寄存器被错误操作。
- 支持模块：FC、IWDT、GPIO。
- 写保护流程：1. 解锁（允许写入） 2. 设置或写入 3. 锁回（禁止写入）
- 函数库中均提供相应的宏定义，可进行解除保护和启用保护。

1.3.1 FC 写保护

FC 中对于 FC_CMD、FC_CTL.OPRLD 等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，解锁后才可对其进行修改，修改完成后建议重新上锁避免闪存数据被意外擦除或是覆盖。

项目	说明
受保护的寄存器	FC_CMD、FC_CTL.OPRLD
保护的目的	避免闪存数据被误擦除或覆写
解锁步骤	1. 检查 FC_STA.CMDULK = 0（锁定状态） 2. 写入 FC_UL = 0x00112233（第一组密钥） 3. 写入 FC_UL = 0x55667788（第二组密钥） 4. 确认 FC_STA.CMDULK = 1（解锁成功）
修改完成后建议重新上锁	写入 FC_UL = 0x00000000

1.3.2 IWDT 写保护

IWDT 中部分寄存器写访问操作是受保护的状态，必须通过以下流程进行解锁及状态检查后方可进行修改。

项目	说明
受保护的寄存器	IWDT_BKPR、IWDT_BKRLR、IWDT_BKWINR
解锁步骤	1. 对 IWDT_BKKR 写入 0x00005555 2. 由于 IWDT 寄存器，存放在备用寄存器中，写入后须等待 33 个 PCLK 时钟更新才完成，可通过 IWDT_BKFR.BUSY 检查状态
重新上锁	写入任意非密钥值

1.3.3 GPIO 写保护

通过 GPIOx_LCK 寄存器，经由 IO 锁定流程，可锁定寄存器写访问功能，此功能开启后无法取消，只允许系统复位清除。

项目	说明
受保护的寄存器	GPIOx_MOD、GPIOx_OT、GPIOx_PUD、GPIOx_AFL、GPIOx_AFH
IO 锁定流程	1. 连续两次写入相同 32 位值至 GPIOx_LCK（写入的 GPIOx_LCK[31:16] = ~GPIOx_LCK[15:0] ） 2. 读取 GPIOx_LCK.LCKK = 1 表示锁定成功

1.4 寄存器读保护

1.4.1 IWDT 读保护

若需读取 IWDT_BKPR.PR、IWDT_BKRLR.RL、IWDT_BKWINR.WIN、IWDT_BKSR.CNT 等寄存器，需依下列流程进行：

1. 读取前须先解锁: 向 IWDT_BKKR 写入 0x00005555
2. 解锁后根据读取的寄存器设置不同操作

读取目标	操作步骤
IWDT_BKPR.PR	设置 IWDT_BKPR.R_PR=1 → 等待 IWDT_BKFR.BUSY=0 → 读取 IWDT_BKPR.PR
IWDT_BKRLR.RL	设置 IWDT_BKRLR.R_RL=1 → 等待 IWDT_BKFR.BUSY=0 → 读取 IWDT_BKRLR.RL
IWDT_BKWINR.WIN	设置 IWDT_BKWINR.R_WIN=1 → 等待 IWDT_BKFR.BUSY=0 → 读取 IWDT_BKWINR.WIN
IWDT_BKSR.CNT	设置 IWDT_BKSR.R_CNT=1 → 等待 IWDT_BKFR.BUSY=0 → 读取 IWDT_BKSR.CNT

1.5 写 1 清零(C_W1)寄存器

- 当中断、唤醒或复位等事件发生时，对应的事件状态寄存器标志会由硬件自动置为 1。其清除方式则需通过对应的事件清除寄存器写入 1 ，由硬件自动完成其目标清除。
- 若直接对事件状态寄存器写入 0，或以其他方式尝试清除标志，皆不会产生任何效果。

2 系统控制

2.1 系统时钟 RCU

系统时钟（SYSCLK）可由以下来源提供：

- HRC (High Speed Internal RC Oscillator) - 内部高速 16 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
- LRC (Low Speed Internal RC Oscillator) - 内部低速 32 kHz RC 振荡器

系统重置后，默认使用内部 16 MHz HRC 作为系统时钟。可通过 RCU_CFG.SW 寄存器切换系统时钟来源；切换前必须确保目标时钟源已开启并稳定。切换后可读取 RCU_CFG.SWS 以确认是否切换完成。

Note:

1. 无法通过 RCU_CON 关闭当前正在使用的系统时钟源，错误操作时，寄存器无反应。
2. 当 PLL 为系统时钟时，作为其输入的 HRC/HOSC 同样不可关闭。
3. 每个时钟源在未使用时均可独立关闭以降低功耗。
4. 周边状态时钟依总线时钟而定，即 AHB/APB，时钟最高可配置为 80 MHz。
   - AHB：主要系统总线时钟（HCLK）
   - APB：外设总线时钟（PCLK）

2.1.1 HOSC 时钟 (HOSCCLK)

表 1. HOSC 时钟模式列表

模式	说明
外部时钟源 (HOSC Bypass)	由外部提供有效的时钟源，并通过 HOSCI 引脚输入。设置寄存器 RCU_CON.HOSCBYP 与 RCU_CON.HOSCON 2 个位启用功能。
外部晶体振荡器 (HOSC Crystal)	通过 RCU_CON.HOSCON 控制开关。RCU_CON.HOSCRDY = 1 表示时钟稳定。

注意事项：当开启 RCU_CON.HOSCON 后，等待 RCU_CON.HOSCRDY 期间应加入 1 ms 超时机制。若逾时未达稳定状态，须关闭后再重新启用，直到标志位 = 1。

2.1.2 HRC 时钟 (HRCCLK)

- HRC 时钟信号是由内部高速 16 MHz RC 振荡器产生。
- 配置 RCU_CON.HRCON 位控制 HRC 开启与关闭。
- HRC 时钟状态可借由 RCU_CON.HRCRDY 标志位确认，当 HRC 时钟稳定时，RCU_CON.HRCRDY = 1。

2.1.3 PLL 时钟（PLLCLK）

PLL 为非整数锁相环（Fractional-N PLL）。输入时钟频率范围为 3~16 MHz，可由 RCU_PLLCFG1.PLLSRC 选择 HRC 或 HOSC。通过 RCU_PLLCFG1.PREDIV 进行预分频（建议维持输入约 4 MHz）。输出范围为 4~80 MHz，由压控振荡器（VCO）提供。

输出时钟频率计算：

$$f_{VCO} = f_{PLLIN} \times \left(FN + \frac{FK}{2^{19}}\right), 192MHz \le f_{VCO} \le 384MHz$$

$$f_{PLL} = \frac{f_{VCO}}{FM}, FM = 2, 4, 6, 8, 12, 16, 24, 32, 48$$

表 2. 计算参数说明

参数	说明
FN	整数倍数控制（RCU_PLLCFG2.FN)
FK	小数倍数控制（RCU_PLLCFG2.FK)
FM	分频控制，可设为 2、4、6、8、12、16、24、32、48 (RCU_PLLCFG2.FM)

PLL 配置流程：

1. 关闭 PLL：RCU_CON.PLLON = 0
2. 确认 RCU_CON.PLLRDY = 0
3. 设置预分频与倍率参数
4. 启用 PLL：RCU_CON.PLLON = 1
5. 等待 RCU_CON.PLLRDY = 1，确认稳定

Note:

1. PLL 的配置（包括输入时钟源的选择、预分频和倍率设置）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 才可修改配置。
2. RCU 没有针对 PLL 的输入时钟源做硬件检查，因此用户在开启 PLL 之前，需确保选择的输入时钟源已经开启且稳定。

2.1.4 LRC 时钟（LRCCLK）

- LRC 为内部低速 RC 振荡器，约 32 kHz，支持在低功耗模式下维持 IWDT 运作。
- 配置 RCU_LCON.LRCON 位控制 LRC 开启与关闭。
- LRC 时钟状态可借由 RCU_LCON.LRCRDY 标志位确认，当 LRC 时钟稳定时，RCU_LCON.LRCRDY = 1。

2.2 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。可以通过 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。

2.3 特殊区块 IP

以下这些特殊区块只能支持特定时钟源

表 3. Clock Source

Source / IP	SYS	AHB	APB	HRC	HOSC	PLL	LRC
UART 时钟 (UARTKCLK)	•		•	•			•
ADC 时钟 (ADCCLK)	•	•				•	
I2C 时钟 (I2CCLK)	•		•	•			
IWDT 时钟 (IWDTCLK)							•
System Tick 时钟 (STCLK)		•					•

2.4 MCO 时钟（MCOCLK)

- 微控制器时钟输出（MCO）功能可将系统时钟输出至外部引脚，供其他设备使用或进行分频测试。
- 通过配置 RCU_CFG.MSW 选择输出时钟，并依需求可配置 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)。
- 支持的输出来源：
  - LRC 时钟
  - HRC 时钟
  - HOSC 时钟
  - PLL 时钟
  - 系统时钟（SYSCLK)
  - AHB 时钟
  - APB 时钟
  - ADC 核心时钟
  - I2C 核心时钟
  - UART 核心时钟

2.5 低功耗唤醒流程

表 4. 低功耗唤醒注意事项列表

注意事项	说明
从低功耗模式唤醒后， 必须检查/清除唤醒标志。	1. 当经由引脚 WKUPx 唤醒后，必须清除唤醒标志 SYSCFG_WKSR.WKCLR。 2. 需检查唤醒标志 SYSCFG_WKSR.FLAG 是否已清除。 3. 由于 UART 预设引脚设置于 WKUPx 引脚上，因此在尚未清除标志前无法使用 UART 功能。
从低功耗模式（STOP0/STOP1）唤醒后，需检查系统时钟来源是否稳定。	1. 低功耗模式（STOP0/STOP1)唤醒后，系统时钟配置保持睡前状态 2. 为了确保时钟稳定，请务必确认时钟来源的 RDY 标志 3. 可由 RCU_CFG.SWS 取得当前的时钟源，在判定相对应的稳定标志 RDY。

表 5. 低功耗模式唤醒后系统执行状态栏表

低功耗模式	系统流程
SLEEP	原流程继续往下执行
STOP0	原流程继续往下执行
STOP1	原流程继续往下执行
STANDBY	系统重启，流程重头开始执行
SHUTDOWN	系统重启，流程重头开始执行

2.6 配置字

表 6. 配置字使用注意事项列表

注意事项	说明
配置字修改与重置	当使用 PTEM/PTEM-V1LITE，修改芯片配置字之后，需要断电后重新上电，才能正常调试。
BOR 配置字	BOR 电压等级（Level)，必须设置高于 MCU 工作电压；电压等级数值，请参考 PT2E01_Datasheet.pdf
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。

3 外设

3.1 CALC 模块

3.1.1 概要说明

CALC 模块提供整数除法与平方根运算功能，可在硬件层加速运算过程，降低 CPU 负担。

3.1.2 常用功能描述

- 支持除法与平方根两种计算模式。
- 支持有符号与无符号运算。
- 运算结果自动更新于结果寄存器中。

3.1.3 配置说明

运算触发方式由 CALC_DIVCON.TRM 位控制：

触发条件	描述
TRM = 0	写入被除数后触发
TRM = 1	写入除数后触发

平方根运算在写入被开方数后即自动开始。

3.1.4 应用限制

- 当 CALC_STAT.BUSY = 1 时，禁止软件读取商数及余数，直至该标志值为 0 才会返回商数及余数。
- 若在 BUSY 状态下重新触发，先前结果将被覆盖。
- 有符号除法若发生溢位，结果不可靠（硬件不提供溢位标示）。
- 当除数为 0 时，CALC_STAT.DZ = 1，结果无效。
- 平方根运算会舍去小数部分。

3.1.5 实际应用建议

当被开方数非常小时，硬件计算误差较大。

可先将输入值左移 << 2n，计算后再将结果右移 >> n，以提高精度。

（注意：左移时需避免 Overflow。）

3.2 CRC 模块

3.2.1 概要说明

CRC（Cyclic Redundancy Check）模块提供硬件加速的循环冗余检查功能，用于数据完整性验证。

3.2.2 常用功能描述

- 支持 8/16/32-bit 数据宽度。
- 可设置多项式、初始值与输入反转。
- 运算结果自动保留于硬件寄存器。

3.2.3 配置说明

1. 启用 RCU 内的 CRC 时钟。
2. 重设 CRC 控制寄存器（CRC_CON.RESET = 1）。
3. 将待校验数据写入 CRC_DATA 寄存器，即自动开始运算，寄存器在 CRC 计算期间被锁住。

3.2.4 应用限制

- 使用前需重设 CRC 模块，以避免残留上一次的计算结果。
- 若对 32-bit 数据进行 8-bit CRC，可直接填入 32-bit 数据，或拆分为连续 8-bit 写入。

3.3 DMA 模块

3.3.1 概要说明

DMA（Direct Memory Access）模块可在不经 CPU 介入的情况下，直接在内存与外设间搬移数据，以提升传输效率。

3.3.2 常用功能描述

- 支持内存与外设之间的双向传输。
- 可设置通道优先权与仲裁率。
- 支持多种数据宽度（8/16/32-bit）。

3.3.3 配置说明

1. 开启对应外设与 DMA 控制器时钟。
2. 配置 DMA 通道来源与目的地址、数据长度、传输方向。
3. 设置通道优先权与触发来源。
4. 启用 DMA 通道（DMA_CHEN）。

3.3.4 应用限制

- DMA 与 CPU 共享 AHB 总线，若同时读写同一地址，CPU 读写会被暂停数个总线周期。
- 数据源与目的地宽度应一致，以避免数据错位或浪费空间。
- DMA 控制寄存器写入后，必须确认读回值一致才视为设置成功。
- CPU 若需操作相同外设，应先确认 DMA 传输已完成。
- 非循环模式可通过 DMA_NDTn.NRDT 确认是否传输完成。

3.3.5 实际应用建议

多通道同时使用时，请依数据重要性调整优先权；可搭配仲裁设置以平衡效能与稳定性。

3.4 FLASH 模块

3.4.1 概要说明

FLASH 模块负责储存程序与用户数据，支持多层级保护机制及等待周期调整。

3.4.2 常用功能描述

- 提供三种保护机制：读保护（RP）、写保护（WP）、用户代码读保护（UCRP）。
- 支持重新上电后保护设置重载。

3.4.3 配置说明

- 所有保护设置均需在重新上电或从 STANDBY / SHUTDOWN 模式恢复后生效。
- 解锁与保护设置需通过配置字重载完成。操作方式可参考 FC 写保护。

3.4.4 读保护（RP）

表 7. 读保护层级说明

等级	描述
Level 0	不保护（预设）
Level 1	禁止程序下载，但内部程序可自我修改。若降级至 Level 0，会触发全擦除并需将烧入器重新上电后才可再进行烧入。
Level 2	禁止程序下载与调试，且不可再修改等级。

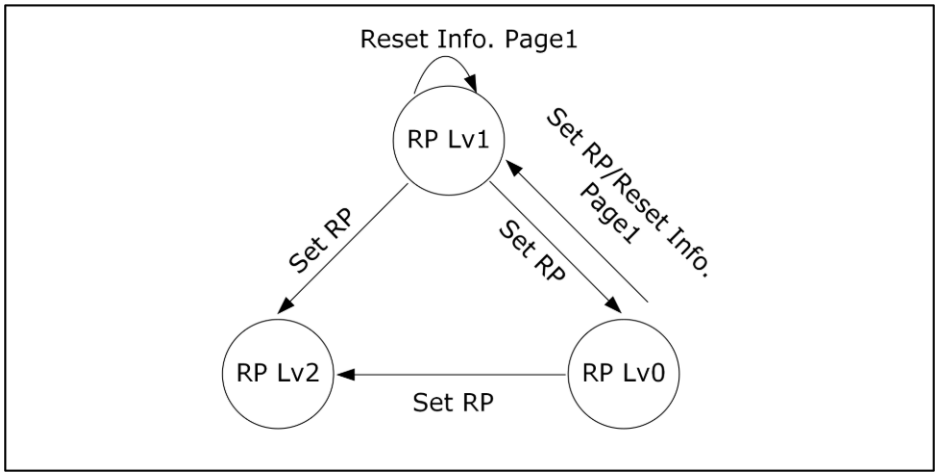
3.4.5 写保护（WP）

- 可设置两个区块（ WP0、WP1）。
- 启用后禁止擦除与写入，需通过外部工具解除保护。

3.4.6 用户代码读保护（UCRP）

- 可设置两个区块（UCRP0、UCRP1）。
- 保护后禁止读取、修改与覆写。

图 1. RP 设置状态图



3.4.7 等待周期设置

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。

表 8. FLASH 等待时钟数(Wait Cycle)

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)

3.5 EXTI 模块

3.5.1 概要说明

EXTI（External Interrupt/Event Controller）模块负责侦测外部事件，产生中断或事件触发，用于唤醒系统或驱动外设。

3.5.2 常用功能描述

- 支持 GPIO 引脚外部中断。
- 可用于 ADC 触发、低功耗唤醒等外部事件来源。
- 提供上升沿、下降沿或双边沿触发模式。

3.5.3 配置说明

1. 配置对应 GPIO 引脚为输入模式。
2. 设置 EXTI 引脚对应寄存器：
   - ICFG1 控制 GPIO0 ~ GPIO7。
   - ICFG2 控制 GPIO8 ~ GPIO15。

3. 设置触发条件（上升沿、下降沿或双边沿）。
4. 启用对应中断或事件。

3.5.4 应用限制

- 若用于 ADC 外部触发或低功耗唤醒，需确保对应 GPIO 已配置中断模式。
- 使用 EXTI 触发 CMP 中断前，请确认已开启并设置对应的 CMP 通道。
- 使用 EXTI 时必须先设置 EXTI 的 IO，再设置 EXTI_RTS 或 EXTI_FTS。

3.6 GPIO 模块

3.6.1 概要说明

GPIO（General Purpose Input/Output）模块提供通用输入与输出控制，可配置为数字输入、输出或多功能复用模式。

3.6.2 常用功能描述

- 支持输入、输出、开漏、推挽等多种模式。
- 可作为外设功能的复用引脚。
- 提供上拉、下拉电阻控制。
- 支持低功耗状态下的引脚保持与唤醒功能。

3.6.3 配置说明

1. 设置 GPIOx_MOD 指定引脚模式（输入、输出或复用）。
2. 若使用外设功能，先设置 GPIOx_AFx 选择复用种类，再将 GPIOx_MOD 设为复用模式(10b)。
3. 若为输入模式，需设置 GPIOx_PUD 为上拉或下拉。

3.6.4 应用限制

- 未使用的 GPIO：建议设为模拟模式（11b）并悬空，以避免漏电。
- 低功耗唤醒引脚：PB00 ~ PB07 支持唤醒功能，需设置 AF9 并开启 SYSCFG_WKUP.WKEN，并选择唤醒事件 SYSCFG_WKUP.WKEG 是上升沿或下降沿模式。
- 低功耗模式保持状态：在 STANDBY/SHUTDOWN 模式下，PB00~PB07 会维持进入前的模式设置（输入/输出、PUD、OT、DS），但寄存器值不保留。
- 在进入 STANDBY/SHUTDOWN 模式前，须先完成唤醒引脚（WKUPx）配置，因进入后该引脚会被自动锁定。系统唤醒后，需将 SYSCFG_WKSR.WKCLR 设为 1 以清除唤醒标志，唤醒流程结束后引脚锁定会自动解除。若 WKUPx 被设置为 UART、I2C、SPI 等复用功能，应在唤醒流程完成后再启用通信。
- 部分 I/O 引脚于出厂时已具备固定上拉或下拉设置，用户在使用时须注意设置：

- SWD CLK (PC00) ： 预设下拉
- SWD DATA (PC01) ： 预设上拉
- NRST ： 预设上拉

3.6.5 实际应用建议

进入低功耗模式前，应确认输出引脚不处于浮动状态，以降低功耗与电气干扰。

3.7 UART 模块

3.7.1 概要说明

UART（Universal Asynchronous Receiver/Transmitter）模块用于异步序列通信，支持可调波特率、自动波特率侦测与中断控制。

3.7.2 常用功能描述

- 支持全双工传输。
- 提供自动波特率侦测功能。
- 支持多种中断事件（接收、传送、错误）。

3.7.3 配置说明

1. 启用对应 UART 时钟。
2. 设置波特率、数据位、停止位、奇偶校验。
3. 启用中断（若需）。
4. 开启 UART 传输（UART_CON1.UEN = 1）。

3.7.4 应用限制

- 若发生自动波特率侦测超时（UART_RIF.ABTO），UART_MCON.ABREN 会自动清除。建议同时启用 UART_MCON.ABRREPT 重复侦测功能。
- 若自动波特率成功但接收数据异常，需重新启用 ABREN。
- 超时后无法收发数据的问题与修正方式：

在部分应用中，若发生自动波特率侦测超时（ABTO=1）后，UART 模块可能暂时无法再接收或传送数据。此现象通常是由内部波特率产生逻辑未重载所造成。

修正方式（Workaround）

当侦测到超时情况时，可通过重新更新 UART_BRR 寄存器的内容来触发硬件重载设置，使 UART 模块恢复正常通信。软件中采以下两种方式进行：

1. Polling 模式
   - 程序主循环中轮询 UART_RIF.ABTO 标志。

- 当侦测到 ABTO=1 时，重新写入一次 UART_BRR 寄存器（即使内容相同）。

2. Interrupt 模式
   - 启用 UART_IER.ABTO 中断。
   - 在中断服务程序（ISR）中判断 UART_IFM.ABTO。
   - 重新写入 UART_BRR 寄存器以更新波特率设置。

   重新设置 UART_BRR 前必须确保 uart tx /rx enable 为关闭状态，否则无法成功设置 UART_BRR

3.7.5 中断处理建议

- 传送中断（TXE）发生时，应先清除中断标志再写入新数据至 UART_TXDATA。
- 第一笔传输的空中断与后续中断间隔时间极短，若未先清除中断可能造成数据错送。

3.7.6 实际应用建议

传送流程建议采：

清除中断 → 写入 UART_TXDATA → 等待 TXE 中断清除 → 准备下一笔

可避免数据竞争。

3.8 I2C 模块

3.8.1 概要说明

I2C（Inter-Integrated Circuit）模块提供主从双向通信功能，支持 7-bit 与 10-bit 地址模式，常用于传感器或 EEPROM 通信。

3.8.2 常用功能描述

- 支持主机与从机模式。
- 支持自动停止与重载传输模式。
- 支持多重中断事件（TX、RX、错误、重启）。

3.8.3 配置说明

1. 启用 RCU 中的 I2C 时钟。
2. 设置 I2C_CON1 与 I2C_CON2 控制寄存器。
3. 配置地址模式（7-bit 或 10-bit）。
4. 设置数据长度（NBYTES）与是否启用自动结束（AUTOEND）。
5. 启用 I2C 模块（I2C_CON1.PE = 1）。

3.8.4 应用限制

- 当 SDA 线被外部设备异常拉低，可能导致误入从机模式。

  建议在 START 之后启动软件超时定时器，若超时则执行：

I2C_CON1.PE = 0 → 重新配置 → I2C_CON1.PE = 1

- 关闭重载模式时最大支持 65535 Bytes。
- 启用重载模式后，传输结束时会自动拉低 SCL，可重新设置下一段数据长度。
- 接收模式下最大传输长度为 255 Bytes，超过需使用重载机制。
- I2C_CON2 必须按字（32 位）访问。
- 在 Master 模式下，若设置为非 AUTOEND 且需要连续传输数据，在重新设置 NBYTES 及 START 位后，必须等待 START 位由高变低，再开始进行数据传输，以避免数据遗漏。

3.8.5 中断处理建议

- 发送空中断时（TXE），I2C_TXDATA 写入后会立即被搬移至 TX 移位寄存器。
- 中断流程建议：

  先清除中断 → 再填写 I2C_TXDATA

  可避免中断密集触发导致的竞态。

地址设置

模式	填入方式	示范
7-bit	地址左移一位	0x50 << 1
10-bit	直接填入	0x50

3.9 SPI 模块

3.9.1 概要说明

SPI（Serial Peripheral Interface）模块提供全双工高速同步序列通信，支持主从模式与多种 FIFO 缓冲中断。

3.9.2 常用功能描述

- 支持主机与从机模式。
- 提供发送与接收 FIFO。
- 支持模式故障检测。
- 支援双边沿采样。

3.9.3 配置说明

1. 启用 SPI 时钟并配置 SPI_CON1。
2. 设置时钟极性、相位与数据宽度。
3. 配置主/从模式与 NSS 管理方式。
4. 启用中断（若使用 FIFO）。

SPI 作为从机时，时钟信号由主机提供，所以建议先使能从机，然后主机再发送时钟，否则数据传输可能不正常。

3.9.4 中断处理

- 发送 FIFO 中断
  - 中断事件：
    - FIFO 空中断（TXE）
    - FIFO 低于阀值中断（TXTH）
  - 中断流程： 优先清除中断标志 → 检查 FIFO 状态 → 填写 SPI_DATA
- 接收 FIFO 中断
  - 事件：
    - FIFO 非空（RXNE）
    - FIFO 满（RXF）
    - FIFO 超过阀值（RXTH）
  - 建议优先读取 SPI_DATA，避免 FIFO 满溢导致数据遗失。

3.9.5 模式故障处理

若在主机模式下发生以下任一情况：

- NSS 引脚被外部拉低
- SPI_CON1.SSOUT = 0

则会触发模式故障（SPI_IER.MODF = 1），同时：

- SPI_CON1.SPIEN = 0（关闭输出）
- SPI_CON1.MSTREN = 0（强制进入从机模式）

恢复方式：

1. 将 NSS 拉高。
2. 清除 SPI_RIF.MODF。
3. 重新设置 SPIEN 与 MSTREN。

3.9.6 结束条件

在关闭 SPI 前，应确保：

模式	条件
全双工	RXFLV = 0 且 SPI_STAT.BUSY = 0
单工 / 半双工（发送）	TXE = 1 且 SPI_STAT.BUSY = 0
单工 / 半双工（接收）	等待 RXNE = 1 后再关闭

3.10 ADC 模块

3.10.1 概要说明

ADC（Analog-to-Digital Converter）模块将模拟信号转换为数字信号，用于传感器输入、电压监测与温度测量等应用。

3.10.2 常用功能描述

- 支持多信道、单次或连续转换模式。
- 提供自动注入信道、扫描与中断功能。
- 内置温度传感器与参考电压信道。

3.10.3 配置说明

1. 在 ADCEN = 0 状态下设置 RCU 与低功耗开关。
2. 启用 ADC 时钟并配置采样时间、触发方式。
3. 在 ADCEN = 1 且 ADCDIS = 0 时，设置转换寄存器与触发。
4. 启用中断或 DMA（视应用需求）。

3.10.4 应用限制

- 仅在 ADC 关闭状态（ADCEN = 0）可更改时钟与低功耗配置。
- 设置自动注入模式时，不可启用非连续模式（DISCEN=0、JDISCEN=0）。
- 硬件并没有强制禁止写入 ADC 寄存器，然而错误的寄存器写入可能导致 ADC 进入未知状态，需重新关闭 ADC 再开启。

3.10.5 VDD（V_{REFP}）计算

由于供应电压可能变动，建议利用内置 V_{REFINT} 通道计算实际 VDD（即 V_{REFP}）：

$$V_{REFP} = FULL_SCALE \times \frac{ADC_{VREFINT_CAL}}{ADC_{VREFINT}}$$

其中:

参数	说明
$ADC_{VREFINT_CAL}$	芯片在生产验证过程中会在 VDDA = 5V 的状态下，利用校准后的 ADC 采样 V_{REFINT} 电压并纪录在配置字中，可利用 ll_syscfg_get_adc_vref(SYSCFG) 读取电压值。
$ADC_{VREFINT}$	当下 ADC 采样 V_{REFINT} 所得 ADC 数值，可借由 ADC_IN[18] 内部通道测量此电压。
FULL_SCALE	ADC 输出最大可表示的数值，12 位分辨率，FULL_SCALE 为 4095。

示例程序

可利用以下函数给入 V_{REFINT} 作为自变量计算 VDD（V_{REFP}）。

```
float ll_adc_vdd_calculation(uint16_t adc_vrefint)
{ return (float)ll_syscfg_get_adc_vref(SYSCFG)/adc_vrefint*4095;
 }
```

Note:

1. 需先执行下列代码打开 V_{REFINT} 开关才可读取到正确电压值。

   ll_adc_enable_vrefinten(ADC12_COMM)

2. ADC Sample Time 建议设置较长的时间以取得较为稳定的电压数值。

3.10.6 真实电压转换

当取得 VDD（V_{REFP}） 之后可根据以下公式对数值换算以取得精确电压值。

$$V_{AIN} = ADC_{AIN}\frac{V_{REFP}}{FULL_SCALE}$$

3.10.7 温度测量

芯片内置温度传感器可通过 V_{TSENSE} 经过进一步计算后取得芯片温度。

$$V_{TSENSE} = ADC_{TSENSE}\frac{V_{REFP}}{FULL_SCALE}$$

当取得 V_{TSENSE} 后代入以下公式即可推算出实际温度。

$$Temparature(℃) = \frac{V_{TSENSE} - V_{30}}{SLOPE} + 30$$

$$SLOPE = \frac{V_{85} - V_{30}}{55}$$

其中:

参数	说明
V_{AIN}	输入模拟电压
ADC_{AIN}	ADC 对输入通道进行采样后输出的数字信号值
ADC_{TSENSE}	ADC 采集温度传感器电压得到的数值，可借由 ADC_IN[17] 内部通道测量电压。
V_{30}	芯片在生产验证过程中会在控温 30°C 的状态下，利用校准后的 ADC 采样 V_{TSENSE} 电压并纪录在配置字中，可利用 ll_syscfg_get_adc_temp_low(SYSCFG) 读取电压值。
V_{85}	芯片在生产验证过程中会在控温 85°C 的状态下，利用校准后的 ADC 采样 V_{TSENSE} 电压并纪录在配置字中，可利用 ll_syscfg_get_adc_temp_high(SYSCFG) 读取电压值。

Note:

1. 需先执行下列代码打开 V_{TSENSE} 开关才可读取到正确电压值。

   ll_adc_enable_vtsen (ADC12_COMM)

2. ADC Sample Time 建议设置较长的时间以取得较为稳定的电压数值。
3. 实际使用示范可参考 ADC_TemperatureSensor 示例程序

3.10.8 ADC 采样的时间限制

ADC 内部电压采样（V_{TSENSE}, V_{REFINT}, V_{RES}）需要一定的采样时间，以下为各内部电压至少需等待的采样时间：

- V_{REFINT}：3 μs
- V_{RES}：0.8 μs
- V_{TSENSE}：3 μs

时间计算方式：

$$t_{sample} = \frac{3.5 + T_{SMP}}{f_{ADC}}$$

其中:

参数	名称	说明
t_{sample}	总采样时间	单次 ADC 采样所需的总时间（单位：秒）。包含固定的前置采样时间与由 SMP 设置决定的可变采样时间。
3.5	固定采样周期	ADC 内部为建立采样电容连接所需的固定周期，单位为 ADC 时钟周期。
T_{SMP}	SMP 设置采样周期	由寄存器 SMPx[7:0] 设置的采样时间对应周期数。

3.10.9 ADC 校准

当系统重新上电后，ADC 的测量精度可能因 V_{REFP} 电压变动而受到影响。为了提升测量准确度，可通过下列线性补偿公式计算补偿参数 ：

$$y = Ax + B$$

其中，A 为增益（Gain），B 为偏移量（Offset）。

为了推导出 A 与 B，需要以两组高、低电压值{x1,x2}、{y1,y2}代入上述公式运算，求得 Gain(A) 与 Offset(B)。后续使用时将 ADC 转换结果(y)减去 Offset 之后除以 Gain 即可回推准确值(x)。

◆ 由于硬件支持此补偿机制，所以在算出 Gain 与 Offset 可填入相关寄存器中并开启硬件补偿模式，在使用 ADC 时即可取得补偿后的转换值。

◆ SDK 所提供的驱动程序(ll_adc.c)内的 adc_calibration() 函数有进行此校正补偿运算。

在 adc_calibration() 程序中，

利用芯片出厂时测量 $\frac{1}{32} \times V_{RES1}$与 $\frac{31}{32} \times V_{RES1}$ 两点（做为 {x1, x2}），

与目前系统状态下测量的 $\frac{1}{32} \times V_{RES2}$与 $\frac{31}{32} \times V_{RES2}$ 数值(做为 {y1, y2})，

代入上述公式运算，求得 Gain(A) 与 Offset(B)。

算出 Gain(A) 与 Offset(B) 后，程序会将数值填入硬件相关寄存器并开启硬件补偿。

参数	说明
$\frac{1}{32} \times V_{RES1}$	芯片出厂时测测量量 5V 供电下 $\frac{1}{32} \times V_{DDA}$ 的电压值，并且记录于配置字中。
$\frac{31}{32} \times V_{RES1}$	芯片出厂时测测量量 5V 供电下 $\frac{31}{32} \times V_{DDA}$ 的电压值，并且记录于配置字中。

参数	说明
$\frac{1}{32} \times V_{RES2}$	目前系统状态下测量的 $\frac{1}{32} \times V_{DDA}$ 的电压值。
$\frac{31}{32} \times V_{RES2}$	目前系统状态下测量的 $\frac{31}{32} \times V_{DDA}$ 的电压值。

3.10.10 校准注意事项

- 由于硬件会对于 Offset 数值饱和进行判断，当测量电压值接近 V_{REFP} 时可能会导致误判，因此补偿方式改为硬件与软件配合进行。

  在使用上仍然借由 adc_calibration() 程序中求得 Gain 与 Offset，只填入 Gain 值借由硬件协助运算，而 Offset 补偿则由取得 ADC 转换数值后借由 SDK 所提供的 Driver(ll_adc.c)内 ll_adc_read() 进行。

- 因为此 Offset 补偿数值极小，为了效率考虑，SDK 示范中默认只靠硬件进行 Gain 补偿，若用户需要精确的测量值则使用 ll_adc_read() 额外补偿 Offset。

  软件 Offset 运算后的数值可能产生 OverFlow 与 UnderFlow 情况，使用时需特别注意。

- 补偿时在下列应用中须特别注意:
  - OverSampling: 根据 OverSampling(ADC_CFG2.OVSRATIO) 的次数与 OverSamplingShift(ADC_CFG2.OVSSHIFT) 的配置，进行 offset 数值倍率的计算。
    - ex1:
      OverSampling = 8x
      OverSamplingShift = No Shifts
      则需要加上 8 倍 Offset
    - ex2:
      OverSampling = 8x
      OverSamplingShift = Shift 2-bits
      则需要加上 2 倍 Offset
  - Compensation：各通道可独立设置 Offset Compensation，若再搭配 ADC Calibration 的 Offset 设置一同使用，则转换结果已由硬件进行 Offset Compensation，只需另外进行 ADC Calibration Offset
    - 正常情况下只有在开启 Compensation 时才会有转换值为负值的情况。用户在单独使用 ADC Calibration 的 Offset 时若发现转换值与其相减后为负值可以自行根据应用判断是否为 0
  - ADC Watch Dog：此看门狗的门坎数值是直接对 ADC 转换值进行判定，所以在靠近门坎边界时可能造成误判，可借由将门坎值加上 Offset 避免误判

3.11 OPAMP 模块

3.11.1 概要说明

OPAMP（Operational Amplifier）模块提供可程序化的运算放大器，适用于信号缓冲、放大与滤波等应用。

3.11.2 常用功能描述

- 支持单端、差动与缓冲模式。
- 可与 ADC 模块直接连接以采集放大后信号。
- 支持低功耗运行与寄存器保持。

3.11.3 配置说明

- 启用对应的 OPAMP 时钟。
- 设置输入端与增益模式（外部分压或内部回授）。
- 设置输出端链接至目标模块（如 ADC）。

3.11.4 应用限制

- 输入阻抗应尽量小以避免失真。
- 使用独立模式时需配置外部分压电阻。

3.11.5 低功耗行为

模式	寄存器行为
SLEEP	无影响
STOP0 / STOP1	无影响，保持原设置
STANDBY / SHUTDOWN	OPAMP 寄存器断电，退出后需重新配置

3.12 TIMER 模块

3.12.1 概要说明

TIMER 模块提供计数、PWM 产生、输入捕获与输出比较等功能，是 MCU 时序控制的核心组件。

3.12.2 常用功能描述

- 支持基本计数、比较、捕获、PWM、编码器模式。
- 支持主从触发、死区时间、互补输出。
- 具备刹车（Brake）与输出安全保护功能。

3.12.3 配置说明

1. 设置计数方向、预分频器与自动重载值。
2. 选择触发来源与更新事件（UERSEL）。

3. 设置输出比较模式（含极性与死区时间）。
4. 若启用刹车功能，配置 BKIN 引脚与保护参数。

3.12.4 应用限制

- 当 UERSEL = 1 时，仅上溢或下溢才产生更新事件。

  例: UERSEL=1 时，Slave Mode Reset 仅初始化计数器，不产生更新事件
- CHn 与 CHnN 互补输出不得同时为有效电平。
- 若死区时间大于脉冲宽度，对应输出脉冲不会产生。

3.12.5 刹车功能

- 刹车输入为电平有效，当保持有效时 GOEN 不能设为 1，此外，BRK 中断状态也无法被清除。
- 当电平配置错误时（例如配置成高电平有效），为了安全目的无法支持双向刹车输入模式。
- 双向刹车模式需设置 E16C6T_BDCFG.BKDID = 1 并设 BKIN 为开漏。
- 刹车事件发生后，GOEN 会自动清除为 0。
  - OFF 状态设置：
  - OFFSSI = 0 输出高阻态，不受控制。
  - OFFSSI = 1
    - 单通道输出时：煞车事件后，通道输出脚会依 OISSn 设置维持于预设的空闲状态。
    - 双通道输出（主输出与互补输出）时：通道输出会先输出无效状态（取决于极性设置），经过死区时间后回到空闲电平。系统会确保主、互补两脚不会同时为有效，以防止功率元件误导通。

3.12.6 通道设置

当信道配置为输入模式时：

CCnNPOL	CCnPOL	模式	说明
0	0	非反相 / 上升沿	在门控或编码器模式下，输入信号不反相；在复位、外部时钟或触发模式下，侦测上升沿有效。
0	1	反相 / 下降沿	在门控或编码器模式下，输入信号先反相；在复位、外部时钟或触发模式下，侦测下降沿有效。
1	0	保留	不可使用。
1	1	非反相 / 上升沿 + 下降沿	在门控、编码器、复位、外部时钟或触发模式下与 00 设置相同，但在输入捕获模式下，侦测双边沿有效。

3.12.7 锁定等级（写保护功能）

表 9. LOCKLVL 层级说明

LOCKLVL	描述
00	无保护
01	锁定部分寄存器（例：E16C6T_CON2 中之 OISS、DTPEN、BRKEN...等）

10	锁定级别 1 + OFFSSR/ OFFSSI/ CCnPOL/ CCnNPOL
11	锁定级别 2 + CHnMOD/ CHnPEN（信道配置为输出模式）

3.12.8 蜂鸣器控制

首先借由软件控制 2 组 Timer 产生具有高频载波之 PWM 调制信号，接着利用 IR Out 将两个信号进行逻辑运算 AND 进而驱动蜂鸣器，并且其中一个 Timer (A32C2T2) 由另外一个 Timer (A32C2T1) 的更新事件触发 Counter 计数，借此达成信号同步，提升声音输出质量。

可由示例程序 **TIMER_sound_generator_two_pwm** 所提供函数控制蜂鸣器

- **BuzzerConfig(uint16_t frequency, uint8_t volume, uint8_t carrier_frequency)**

  使用时依次给予音调频率(Hz)、音量比例、载波频率(kHz)作为自变量，函数内会自动进行运算并设置 Timer

  Note: 若调整了音调频率或是载波频率，需要同步对音量比例进行调整

- **BuzzerStop(void)**

  关闭蜂鸣器输出

- **BuzzerVoiceFadeAway(void)**

  控制蜂鸣器音量平滑缩减至无声

3.13 IWDT 模块

3.13.1 概要说明

IWDT（Independent Watchdog Timer）模块独立于系统主时钟，用于侦测软件失控并触发系统重启。

3.13.2 常用功能描述

- 采用 LRC 作为时钟来源。
- 具可调预分频与重载值设置。
- 超时未喂狗将导致 MCU 复位。
- 提供「窗口时间」机制，监测喂狗时机。
- IWDT 启用后便无法停止。

3.13.3 配置说明

1. 对 IWDT_BKKR 写入 0x0000CCCC 启用 IWDT
2. 配置写入解锁，参考 IWDT 写保护
3. 设置预分频值（PR）与重载值（RL[11:0]）。
4. 配置 IWDT 重载计数器数值。
5. 根据启用窗口与否进行不同设置：

- 不启用窗口设置：将 IWDT_BKRLR 的值刷新到看门狗定时器 (IWDT_BKKR=0x0000AAAA)。
- 启用窗口设置：配置窗口寄存器 IWDT_BKWINR。此时会将 IWDT_BKRLR 的值刷新到看门狗定时器。

每次解锁及写入设置都需要等待 IWDT_BKFR.BUSY 数值更新为 0

3.13.4 超时时间计算

超时计算公式如下：

$$t_{IWDT} = t_{LRC} \times PR \times RL$$

其中：

参数	说明
t_{IWDT}	IWDT 超时时间
t_{LRC}	LRC 时钟周期，固定为 $\frac{1}{32768}$≈30.5 μs
PR	预分频值，由分频器设置决定
RL	重载计数值（0x000 ~ 0xFFF），预设为 0x0FFF

Note:
1. 详细设置可参考 IWDT 示例程序
2. 当系统因 Watchdog 复位循环重启，可使用 NRST 引脚拉低一段时间进行系统复位。

3.14 WWDT 模块

3.14.1 概要说明

WWDT（Window Watchdog Timer）模块监控程序运行时间窗口，防止过早或过晚喂狗，确保系统稳定运行。

3.14.2 常用功能描述

- 提供「窗口时间」机制，监测喂狗时机。
- 喂狗过早或过晚皆会触发复位。
- 支持中断模式，用于数据保存或安全处理。

3.14.3 配置说明

1. 设置分频器（WDGTB）与窗口值（WWDT_CFG.W）。
2. 启用 WWDT 并写入初始计数值。
3. 在允许窗口内喂狗，若超过则触发复位。

3.14.4 超时时间计算

超时计算公式如下：

$$t_{WWDT} = t_{PCLK} \times 4096 \times 2^{WDGTB} \times (t[5:0] + 1)\ (ms)$$

其中：

参数	说明
t_{WWDT}	WWDT 超时时间
t_{PCLK}	APB1 以 ms 为单位的时钟周期
t[5:0]	计数值

3.14.5 中断与复位行为

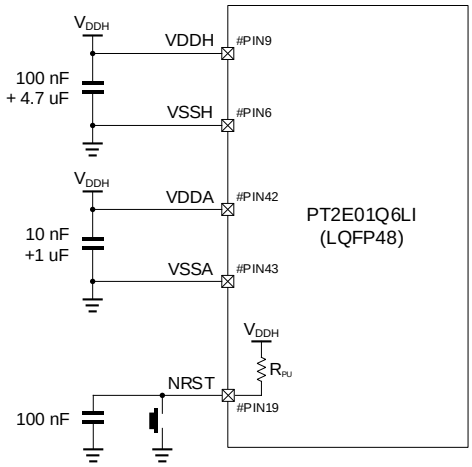
- 计数从 0x7F 递减至 0x3F 时触发复位。
- 若启用中断，当计数至 0x40 会产生中断事件，可于中断中保存数据或执行安全操作。

4 系统电路与版图注意事项

4.1 最小系统电路

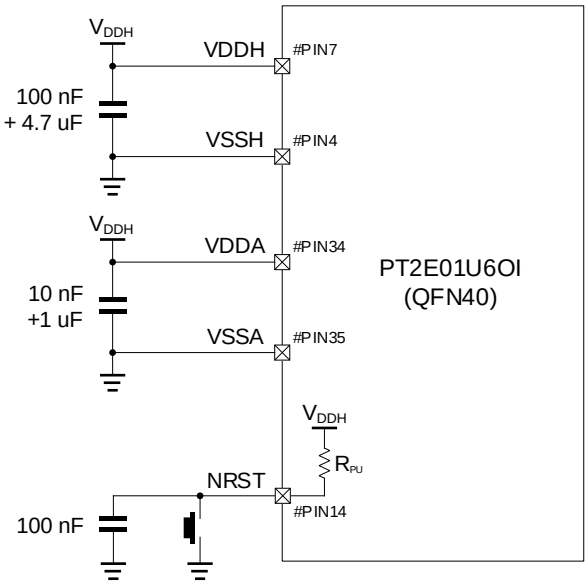
4.1.1 LQFP48 封装芯片最小系统电路

图 2. LQFP48 封装芯片最小系统电路



4.1.2 QFN40 封装芯片最小系统电路

图 3. QFN40 封装芯片最小系统电路



Note:

1. 电源设计：芯片电源方案必须依照上图建议，以确保在各种应用场景下皆能稳定运行。
2. 去耦电容配置：每组电源引脚需连接陶瓷耦合电容（如图所示），并应尽可能靠近对应管脚，以降低电源噪声、提高芯片稳定性。
3. 复位电路设计：NRST 引脚采用 RC 复位，内部支持上拉电阻（上拉值请参考数据手册电气特性章节），电容建议使用 100nF。

4.2 外围电路

4.2.1 外部晶振 HOSC

注意事项	说明
外部陶瓷电容	电容建议数值，请参阅 PT2E01 数据手册，电气特性章节。
版图参考原则	外部晶振应尽量靠近 IC，提高稳定度。

4.2.2 PTEM / PTEM-V1LITE 与调适接口

注意事项	说明
电路建议	• PC00 与 PC01 为 PTEM/PTEM-V1LITE 调适接口。 ○ PC00（SWCLK）：默认为输入下拉模式。 ○ PC01（SWDIO）：默认为输入上拉模式。 • 若应用程序将此两脚配置为一般 IO 功能，需预留 NRST 引脚。 当 NRST = 0 时，系统可自动恢复 SWCLK 与 SWDIO 功能，用于刻录或除错。
Socket 刻录时的电源方案	使用 Socket 进行脱机刻录时，请确保芯片电源依照 图 2. LQFP48 封装芯片最小系统电路 的电源方案建议，否则可能造成刻录异常或启动失败。

4.2.3 UART-BOOT 刻录接口

注意事项	说明
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，PT2E01 预设为 Main Flash。此配置可通过修改 配置字 更换默认值。
UART-BOOT 接口选用	当配置字 BOOT BYPASS 选项为 BootRom 时： • 系统开机从 BootRom 开始执行。 • PB00 与 PB01 引脚默认为 UART1 通信接口： ○ PB01 → UART1_TX（输出） ○ PB00 → UART1_RX（输入） • 可通过 UART 通信接口搭配 UART-BOOT 刻录工具进行程序下载或更新。
	当配置字 BOOT BYPASS 选项为 Main Flash 时： • 系统开机将直接跳转至 Main Flash 区域执行。 • PB00 与 PB01 引脚预设为 GPIO。 • 不具备 UART-BOOT 刻录功能。

4.2.4 上下电斜率

上电斜率需小于 1.2s/V，下电斜率需大于 140μs/V，才能正常启动

5 文件版本纪录

表 10. 文件修订版本记录

日期	版本	修订内容
2025-12-03	PRE1.0	初版
2026-07-08	PRE2.0	1. 3.8 I2C 章节新增 Master 模式下连续传输注意事项。 2. 4.1.2 封装信息更新为 QFN40。