PAN271x 开发套件使用手册

发布 0.3.0

Panchip PRF SoC Team

2026 年 03 月 14 日

Table of contents

Chapter 1

快速入门

1.1 SDK 快速入门指南

1.1.1 1 概述

本文是 PAN271x SDK 的快速入门指引，旨在帮助使用者快速入门 PAN271x SoC 的开发。

1.1.2 2 PAN271x EVB 介绍

PAN271x EVB (EValuation Board) 是 Panchip 提供给 PAN271x SoC 用户的一系列开发板的总称，目前包括 1 块 EVB 核心板，1 块 EVB 底板：

开发板名称	SoC 型号	封装	OTP 大小	SRAM 大小
PAN2710U5GA EVB 核心板	PAN2710U5GA	QFN28	16 KB (SRAM 模拟)	3 KB
PAN271x EVB 底板	-	-	-	-

> **注意**：PAN271x EVB 核心板上搭载的 PAN2710U5GA 芯片为预生产芯片，可支持多次烧录，方便用户进行开发调试；而量产版本的 PAN271x 芯片为 OTP 芯片（详见 PAN271x 产品说明书），仅可烧录一次，且烧录后不可擦除。

关于 PAN271x EVB 开发板硬件的详细介绍，请参考PAN271x EVB 硬件资源介绍。

1.1.3 3 SDK 开发环境确认

3.1 搭建 PC 开发环境

在使用 PAN271x SDK 开发之前，请确保您的 PC 上有如下开发环境：

- `Keil MDK`：我们使用 Keil MDK v5 + ARMCC v5.06 作为集成开发环境 IDE
- `JLink`：我们使用 JLink SWD 的方式调试和烧录程序

关于具体的开发环境要求与搭建建议，请参考SDK 开发环境搭建。

3.2 下载 PAN271x DK 开发套件

您可以通过如下几种方式获取到 PAN271x DK 开发套件：

1. 从 PAN271x 芯片的产品介绍 WIKI 网页中的 “**产品开发资料**” 一节中下载到最新版本的开发套件：
   - PAN271x 系列产品介绍

2. 从 PAN271x DK 文档中心（即本文档所在的网站）网页的左下角“**版本选择及下载**”选项卡中，下载到与文档版本相对应的开发套件版本：
   - PAN271x Development Kit 文档中心
3. 直接联系 Panchip 获取

3.3 PAN271x DK 框架概览

```
<PAN271x-DK>
  01_SDK                  // PAN271x SDK 软件开发包，包括 SoC 驱动、例程、及相关脚本等
      build_tools           // Build 工具，包括 JFlash 烧录工具、编译脚本等工具
      components            // Component 组件，包括 PAN_USB 等组件
      drivers               // SoC Drivers 驱动，包括 GPIO、UART、SPI、I2C 等硬件驱动
      platform              // Platform 平台相关代码，包括芯片启动代码、平台初始化代码、Log 机制代
码等
      proprietary_rf        // PRF 2.4G 相关代码，包括 2.4G Lib、2.4G API 接口等
      samples               // Samples 例程，包括组件/驱动/2.4G/低功耗等例程工程
  02_HDK                  // PAN271x 硬件参考设计，开发板的相应图纸等
  03_DOC                  // PAN271x DK 文档（网页）
  04_TOOLS                // PAN271x 相关工具，包括量产烧录工具，RF 测试固件等
```

3.4 快速编译运行一个简单的例程

1. 将 EVB 核心板插到 EVB 底板上
2. 将 EVB 底板 SWD（P00：SWD_CLK，P01：SWD_DAT，GND：SWD_GND）接口通过 JLink 连接至 PC
3. 将 EVB 底板 USB->UART 接口通过 USB Type-C 线连接至 PC，同时将 EVB 底板 J8 排针对 (SoC_P06 & USB-UART_TXD) 和 J9 排针对 (SoC_P05 & USB-UART_RXD) 分别用跳线帽短接起来，然后在 PC 上打开串口终端或串口调试助手（**串口波特率：** 115200）
4. 将 EVB 底板 J17 排针对 (P04 & LED) 使用跳线帽短接起来
5. 打开 PAN271x SDK 中的 GPIO 例程（演示 GPIO 推挽输出控制 LED 灯闪烁）：`<PAN271x-DK>\01_SDK\samples\drivers\gpio`
6. 点击 Keil `Build` 编译按钮，成功后点击 `Download` 按钮进行烧录下载
   - 若无法正常编译，请检查 Keil 版本以及 ARMCC 编译器版本是否正确
   - 若无法正常下载，请确认使用的芯片型号是否为 PAN2710U5GA（预生产版本芯片，SRAM 模拟 OTP）
7. 烧录成功后：
   - 观察串口 Log 打印，可以看到系统成功初始化并闪灯的 Log：

     ```
     CPU @ 48000000Hz
     LED on
     LED off
     LED on
     LED off
     ...
     ```

   - 观察 EVB 底板上的 LED 灯，可以看到其以约 1Hz 的频率闪烁

1.1.4 4 更多相关文档

下面这些文档有助于您进一步了解 PAN271x 系列 SoC 开发的相关知识：

1. SDK 例程汇总：列出了目前 SDK 内置的所有例程的简单介绍及对应文档的跳转链接

2. PRF 2.4G 开发指南：介绍 PRF (Panchip Private RF) 2.4G 开发的基本方法
3. 常见问题（FAQs）：介绍 PAN271x 开发中的常见问题及解决方法

1.2 SDK 开发环境搭建

1.2.1 1. Keil MDK 集成开发环境（IDE）

PAN271x DK 的 SDK 部分基于 Keil MDK v5 + ARMCC v5.06 版本构建，因此我们推荐您也使用相同的 IDE 版本及编译器版本，以确保能够正常开发。

您可以从 Keil 官方网站的历史版本页面中，下载指定版本的 Keil MDK 安装包: https://www.keil.com/update/rvmdk.asp

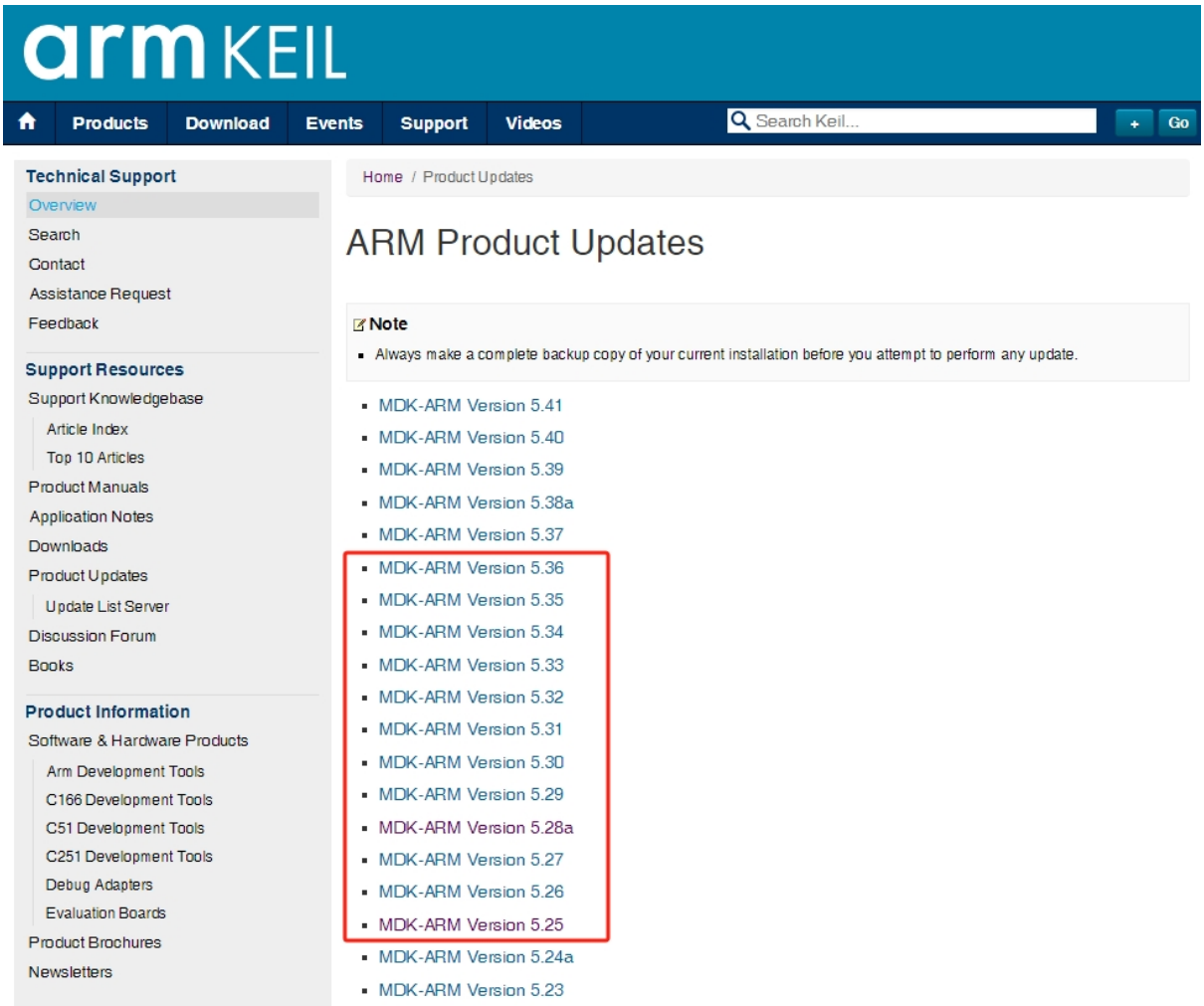

图 1: Keil MDK v5 历史版本下载页面

重要: 我们推荐您使用 MDK v5.25 ~ v5.36 之间的版本, 这些版本在安装完成后，无需额外操作即可直接编译 SDK 中的所有例程。

1. 若您的 PC 已经有 Keil MDK 开发环境，但版本低于 v5.25，则建议您更新 Keil 版本至我们上述的版本
2. 若您的 PC 已经有 Keil MDK 开发环境，但版本高于 v5.36，那么：

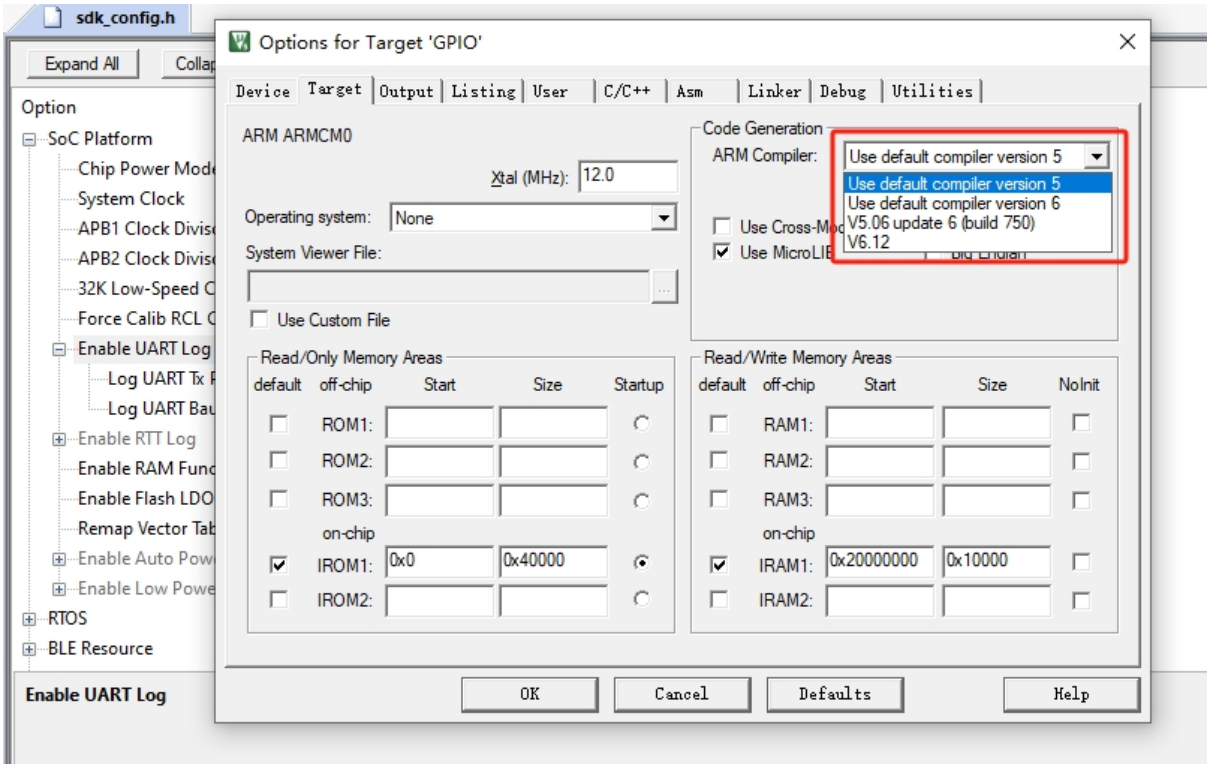

图 2: Keil 编译器版本选择

- 请首先在 Keil 工程配置界面确认是否有 ARMCC v5.06 版本的编译器
- 若没有，则需要从 ARM 官方网站中找到单独的 ARMCC 编译器安装包进行安装: https://developer.arm.com/documentation/ka005198/latest/

1.2.2 2. JLink 烧录器

PAN271x SDK 例程默认使用 JLink 进行程序烧录与调试（针对预生产版本（SRAM 模拟 OTP）的芯片）。

实际上，由于芯片 SRAM 空间较小（3KB），无法直接使用 Keil 自带的 JLink 烧录功能，因此我们在 SDK 中嵌入了一个 JFlash 工具（位于 `<PAN271x-DK>/01_SDK/build_tools/JFlash` 目录），当在 Keil 中点击 Download 按钮的时候，会调用该 JFlash 工具进行烧录。

另外，我们推荐您使用 JLink v9 或更高版本的 JLink 烧录器硬件，以确保能够正常烧录与调试。

> **注**: 对于 OTP 版本的芯片，我们建议您使用 PANLink 量产烧录工具进行烧录，详见量产烧录 文档说明。

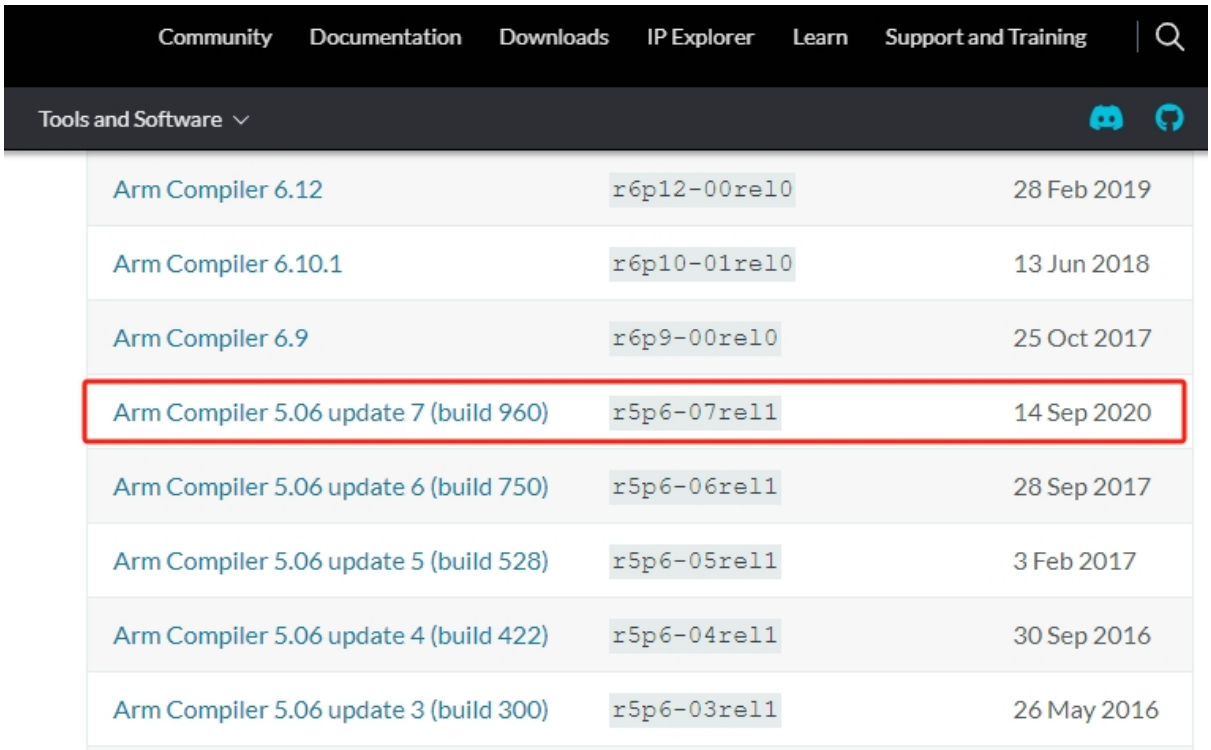

图 3: ARMCC 编译器下载

Chapter 2

硬件资料

文档列表:

2.1 PAN271x EVB 硬件资源介绍

2.1.1 1 概述

本文为 PAN271x Evaluation Board (EVB) 开发板介绍，包括相关板级硬件模块、各模块在 EVB 板上的位置、以及对应电路原理图，旨在帮助开发者快速了解 PAN271x EVB 开发板。

PAN271x EVB 开发板由核心板和底板两大模块组成，其中：

- 核心板提供了 PAN271x SoC 的最小系统，主要包含 PAN271x SoC 芯片、32MHz 高速晶振、32768Hz 低速晶振、复位按钮、板载天线以及一些必要的无源器件

  PAN271x QFN28 核心板还搭载了一个 MIC 电路（支持单端和差分）。

- 底板上提供了 PAN271x SoC 支持的外设模块，其中包含电源管理系统、USB-TypeC 转串口模块、红外模块、LED 灯、独立按键、SWD 烧录口等

2.1.2 2 核心板硬件资源

PAN271x 核心板由芯片最小系统、MIC 电路和转接板组成，核心板以模块的形式嵌入 EVB 底板。

目前 EVB 核心板仅有 PAN2710U5GA（QFN28）这一种 SoC 型号，此型号为预生产版本，其程序存储器为 16KB SRAM 模拟的 OTP，可多次烧录以方便前期调试。

1. 注意 PAN2710U5GA V1.0 核心板 P12/P13/P14 三个引脚的丝印有误，实际丝印请以上述图片为准。
2. 关于芯片最小系统的硬件设计建议，请参考PAN271x 硬件参考设计指南 文档中的相关介绍。

2.1.3 3 底板硬件资源

PAN271x EVB 底板由一系列外设器件组成，包含电源管理系统、USB-TypeC 转串口模块、红外模块、LED 灯、独立按键、SWD 烧录口等。

3.1 电源模块

PAN271x EVB 可选择使用以下两种供电方式之一：

- 5V USB 供电（**推荐**）

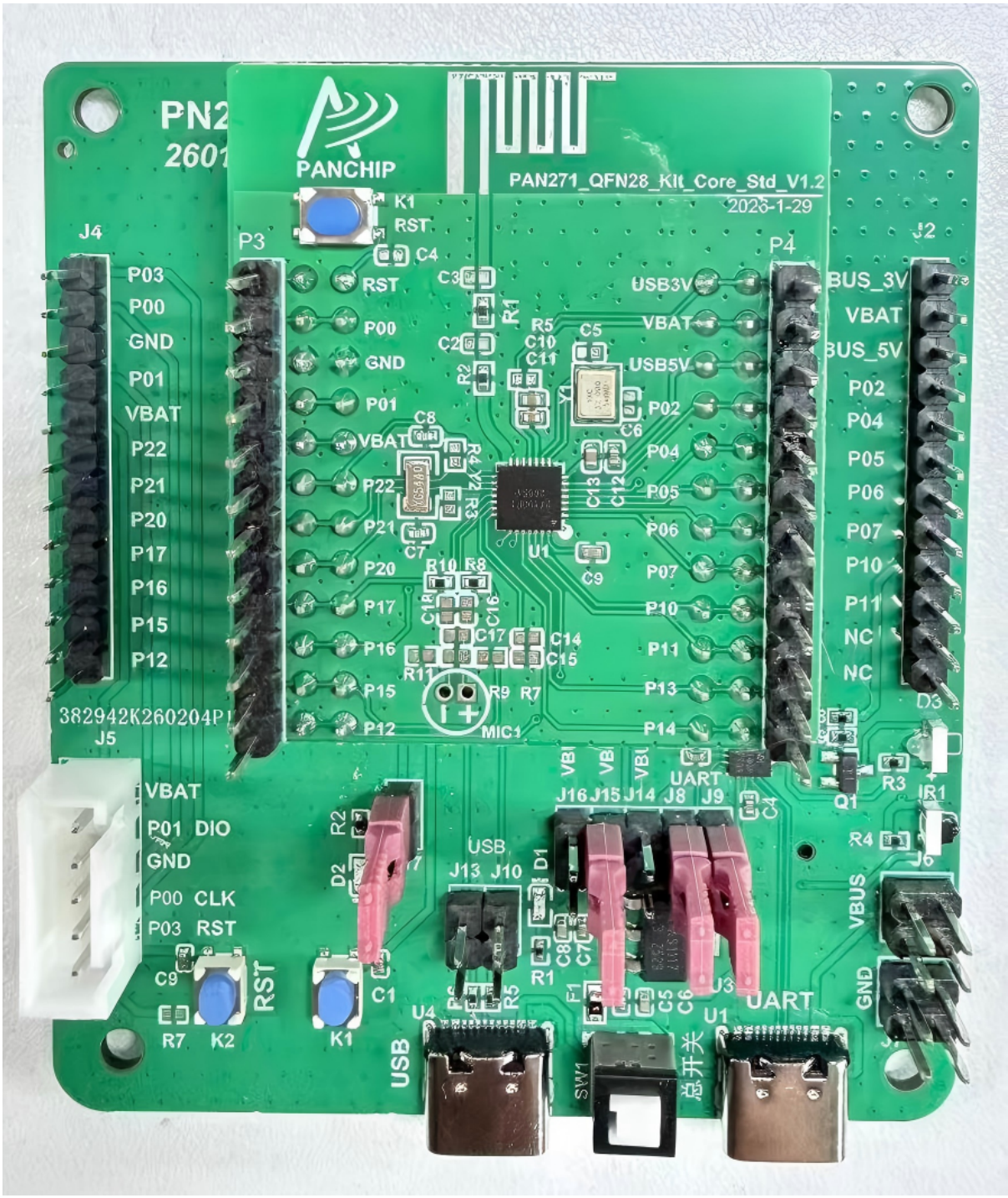

图 1: PAN271x EVB V1.0 实物图（核心版 + 底板）

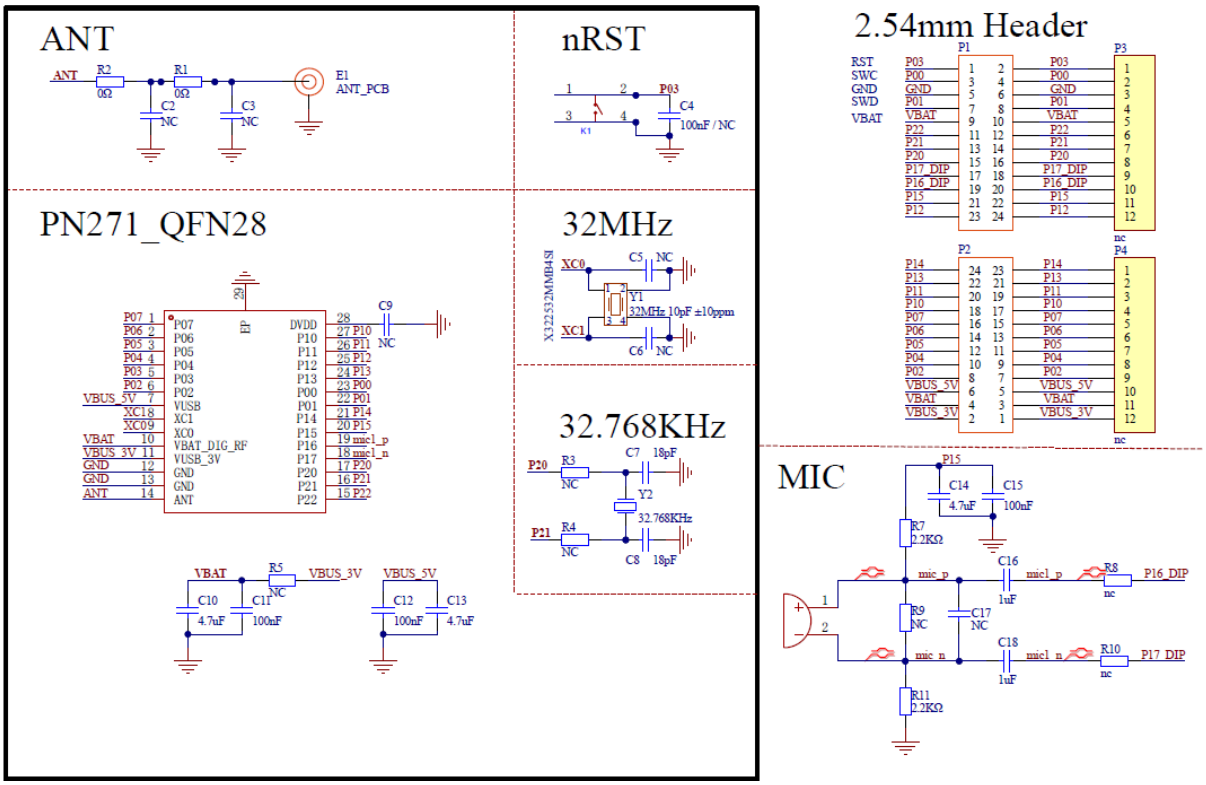

图 2: PAN2710U5GA 核心板原理图

- 3.3V JLink 供电

EVB 底板电源模块原理图如下：

EVB 底板下侧有两个 USB-TypeC 接口 `U1` 与 `U4`，它们的电源引脚均与 EVB 的 5V 电源网络连在一起。二者区别是：

- `U1` 为 USB 转 UART 模块的 USB 端接口，使用跳线帽将此模块与 PAN271x UART 引脚相连，即可实现 PAN271x 与 PC 进行串口通信功能
- `U4` 为普通 USB 接口，使用跳线帽将此接口 PAN271x USB 对应引脚相连，即可使用 PAN271x 的 USB 功能

  注：在实际使用过程中，选用任意一个 USB 接口供电均可，但需注意电源总开关 SW1 应处于闭合状态。

一种典型的供电方法如下图所示：

1. 使用 USB-TypeC 线连接 EVB 底板的 `U1`（USB 转串口模块）接口与 PC 端 USB 接口
2. 使用跳线帽短接 `J15` 排针对
3. 使用跳线帽分别短接 `J8` 排针对和 `J9` 排针对，用于 USB 转串口模块通信
4. 闭合电源总开关 SW1

这样，USB 5V 电源即可通过板载 LDO `U3` 转换为 3.3V 电压后，再通过 `J15` 排针通路输出到 EVB 核心板的 VBAT 电源引脚，为 PAN271x SoC 供电。

提示

PAN271x SoC 的某些封装型号（如 MSOP10 封装的 PAN2710M5BA），有一个名为 `VUSB` 的引脚，此引脚可以直接输入 USB 5V 电压，通过芯片内部的 LDO 将其转换为 3.3V 后，再提供给芯片供电，这样在 USB 应用中可以省掉一个外部 LDO 器件。

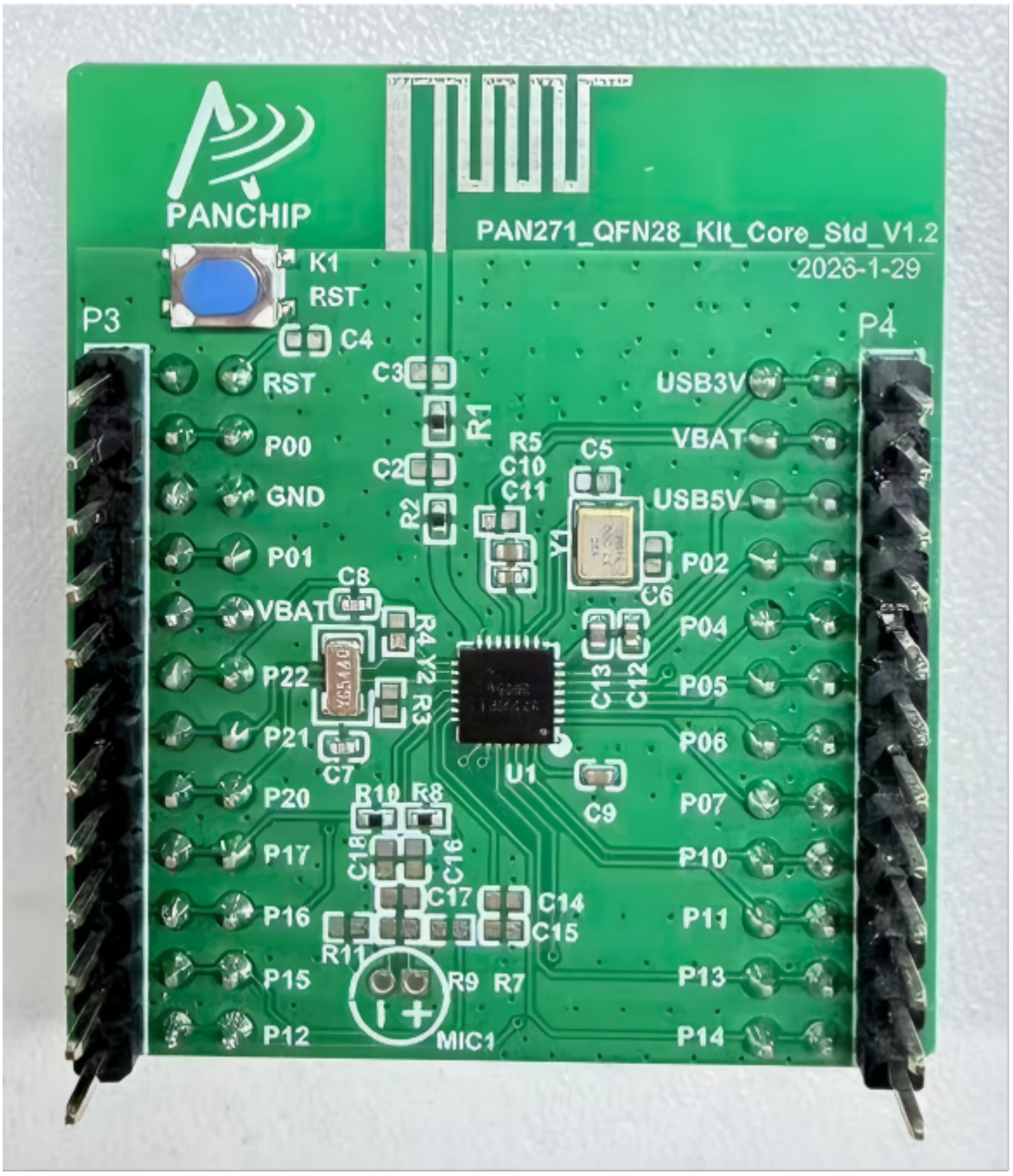

图 3: PAN2710U5GA 核心板

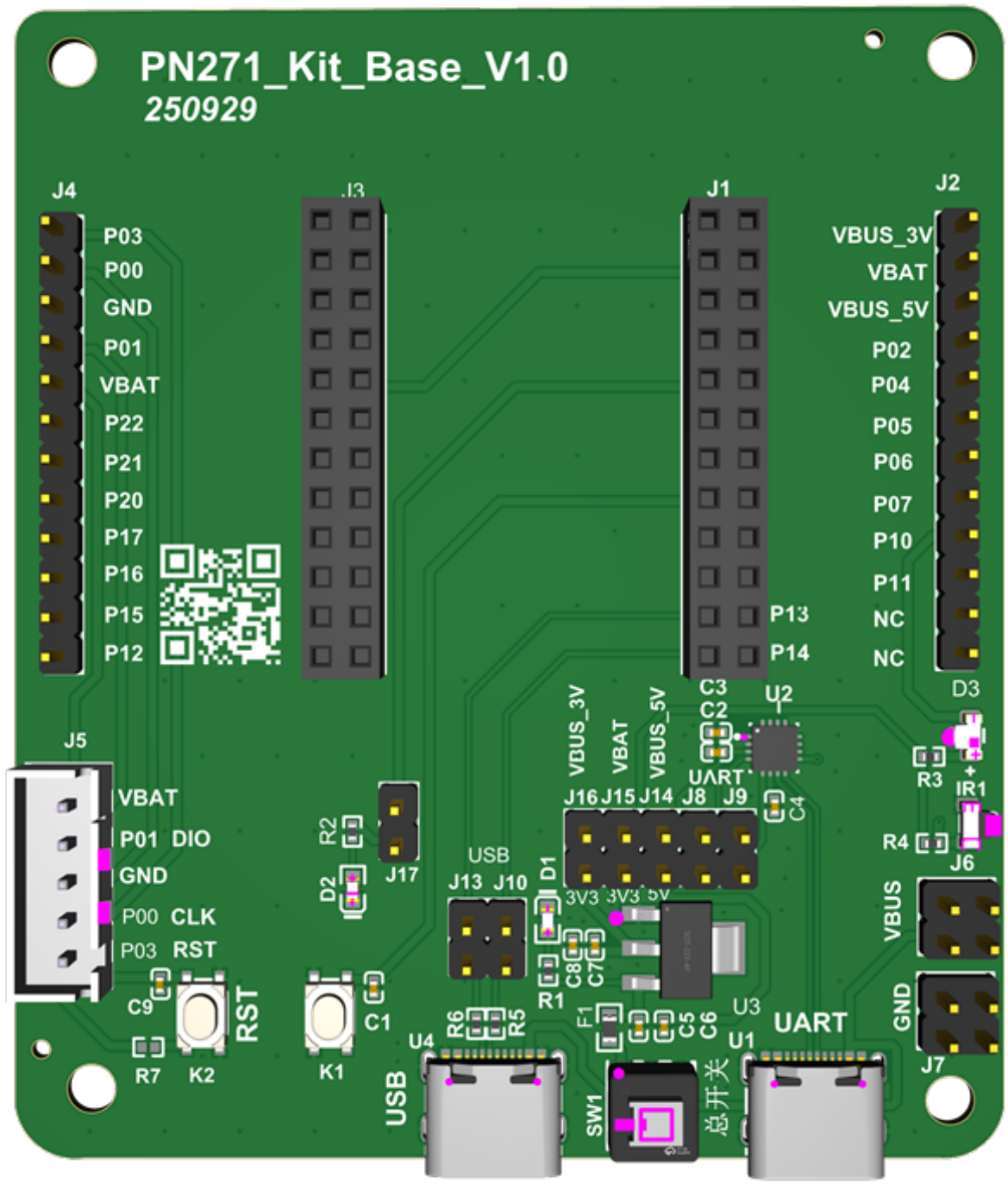

图 4: PAN271x EVB 底板三维图

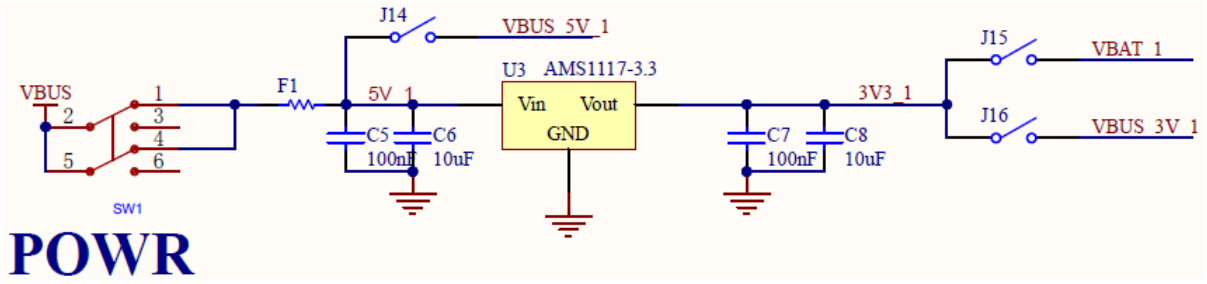

图 5: 电源模块原理图

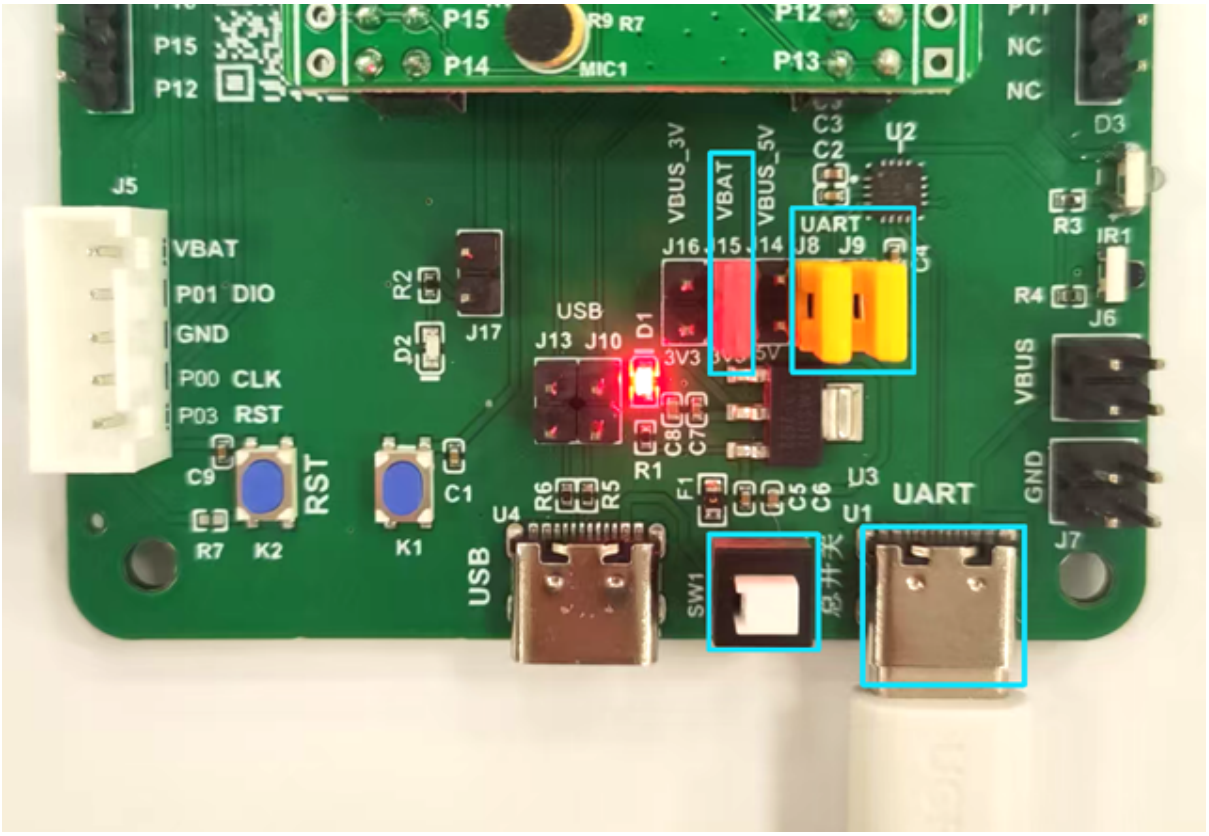

图 6: 典型供电方式示意图

3.2 SWD 调试接口

EVB 底板提供了单排针接口用于连接 JLink 实现 SWD 调试和程序下载功能，该排针接口位于左下方：

1. 将 JLink 下载器插到 SWD 接口 J5
2. 此时供电方式可参考上文电源模块介绍中的典型 USB 供电方式，也可直接使用 JLink 的 3.3V 电源给 EVB 板供电

3.3 USB 转串口模块

PAN271x SoC 的某些引脚可通过软件配置成 UART 串口功能，此时可通过 USB 转 UART 芯片 CH343（U2）转为 USB 接口与 PC 通信。

USB 转 UART 模块使用 U1 USB-TypeC 接口，并且使用跳线帽分别短接 J8 排针对 (SoC_P06 & USB-UART_TXD) 和 J9 排针对 (SoC_P05 & USB-UART_RXD)，如下图所示：

3.4 USB 模块

开发板提供一个 USB-TypeC 接口（U4），其可通过跳线连接到 PAN271x SoC 的 USB 引脚。

为使用此模块功能，需要将跳线帽分别短接 J10 排针对 (SoC_P14 & USB-DP) 和 J13 排针对 (SoC_P13 & USB-DM)，如下图所示：

3.5 独立按键

EVB 底板配备了 2 个独立按键，其中一个为普通 GPIO 按键，另一个为复位按键：

- 按键 K1 直接连接至 SoC 的 P07 引脚，可当做普通按键使用

图 7: JLink 接线示意图

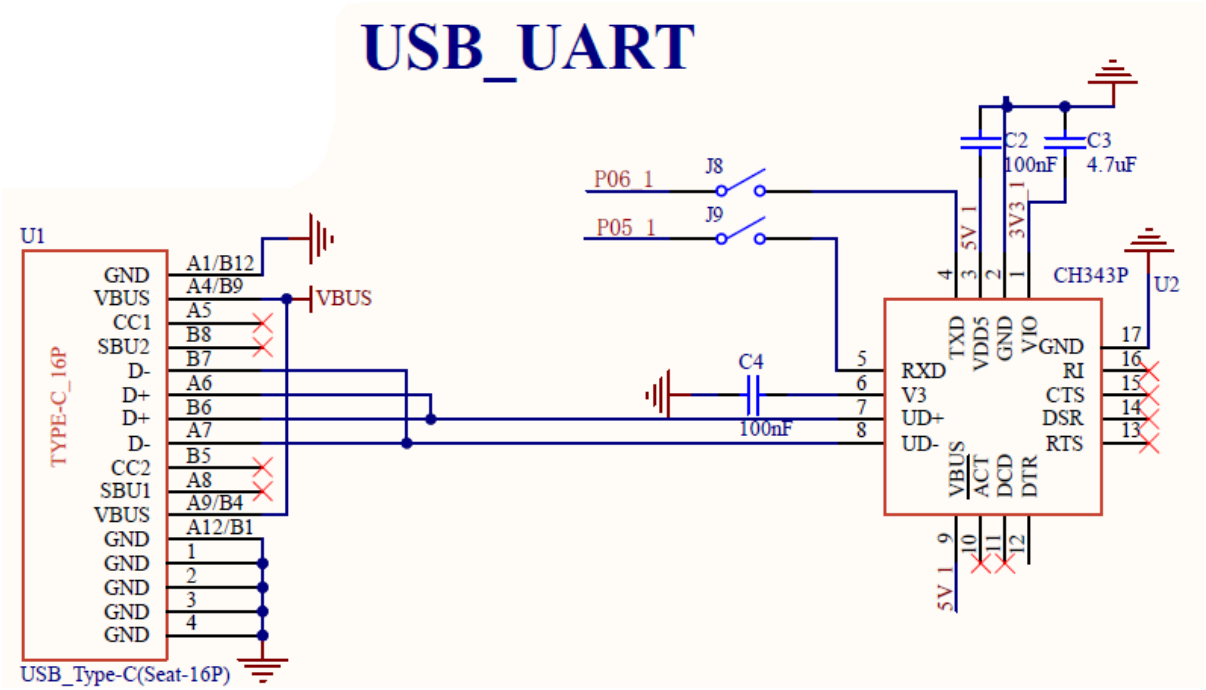

图 8: USB 转 UART 模块原理图

- 按键 K2 直接连接至 SoC 的 P03 引脚，此引脚上电默认为芯片 Reset 引脚，可通过软件将其配置为 GPIO P03 功能

3.6 LED 灯

EVB 底板配备了 2 个 LED 灯，其中一个为 5V 供电指示灯，另一个为普通 IO 控制指示灯：

为使用 IO 控制指示灯 D2，需要使用跳线帽短接 J17 排针对：

3.7 红外模块

EVB 底板搭载了一个红外收发模块，包括一个发射电路和一个接收电路：

2.2 PAN271x 硬件参考设计

2.2.1 1 原理图设计

1.1 参考原理图

1.1.1 PAN2710U5GA（QFN28）最小系统

1.1.2 PAN2710P5DA（SOP16）最小系统参考原理图

1.1.3 PAN2710M5BA(MSOP10)最小系统参考原理图

提醒

- 模组不过认证，且是 PIFA 天线时，需串联一个电容到天线，避免大功率自激。
- 过 FCC/CE 认证，至少需要预留一个 PI 型匹配结构，元件值参考安规设计文档。

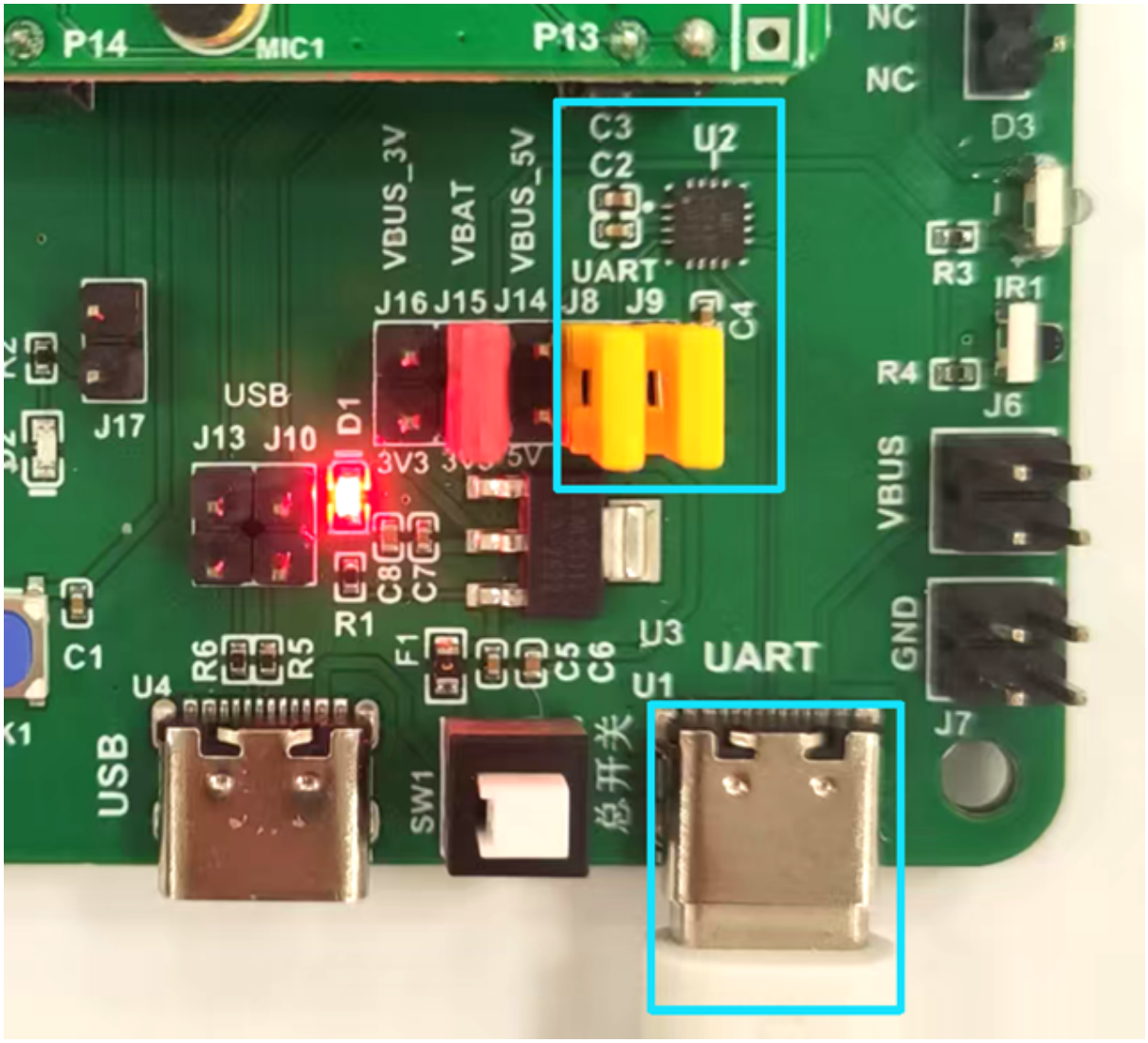

图 9: USB 转 UART 模块实物接线图

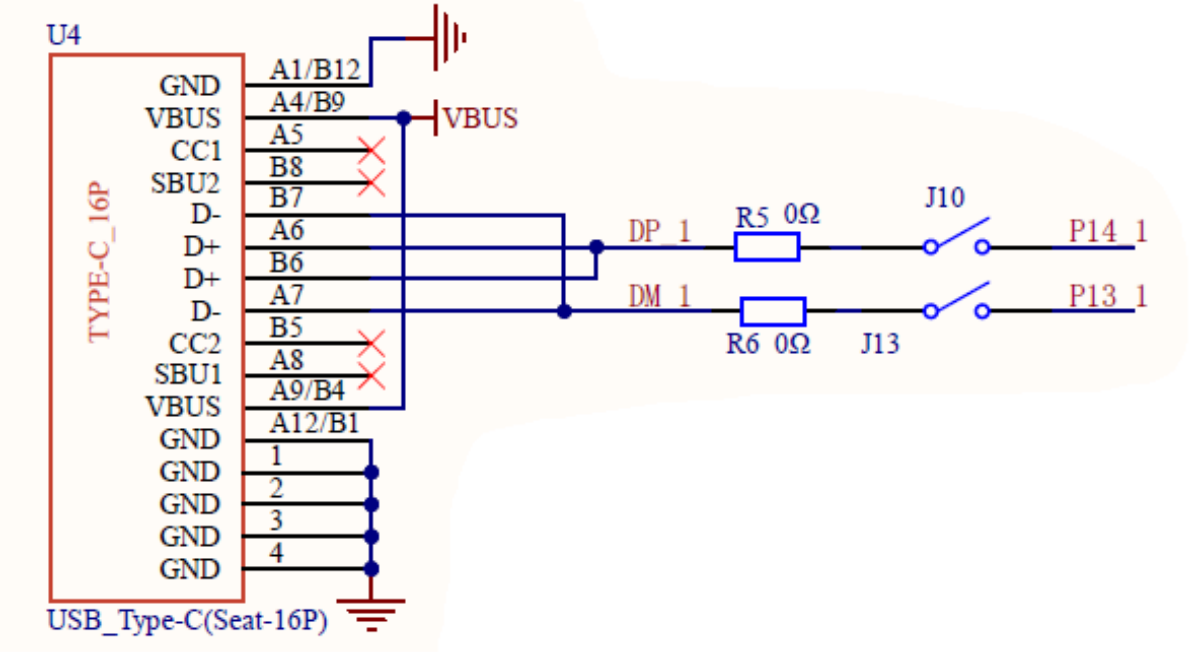

图 10: USB 模块外围电路原理图

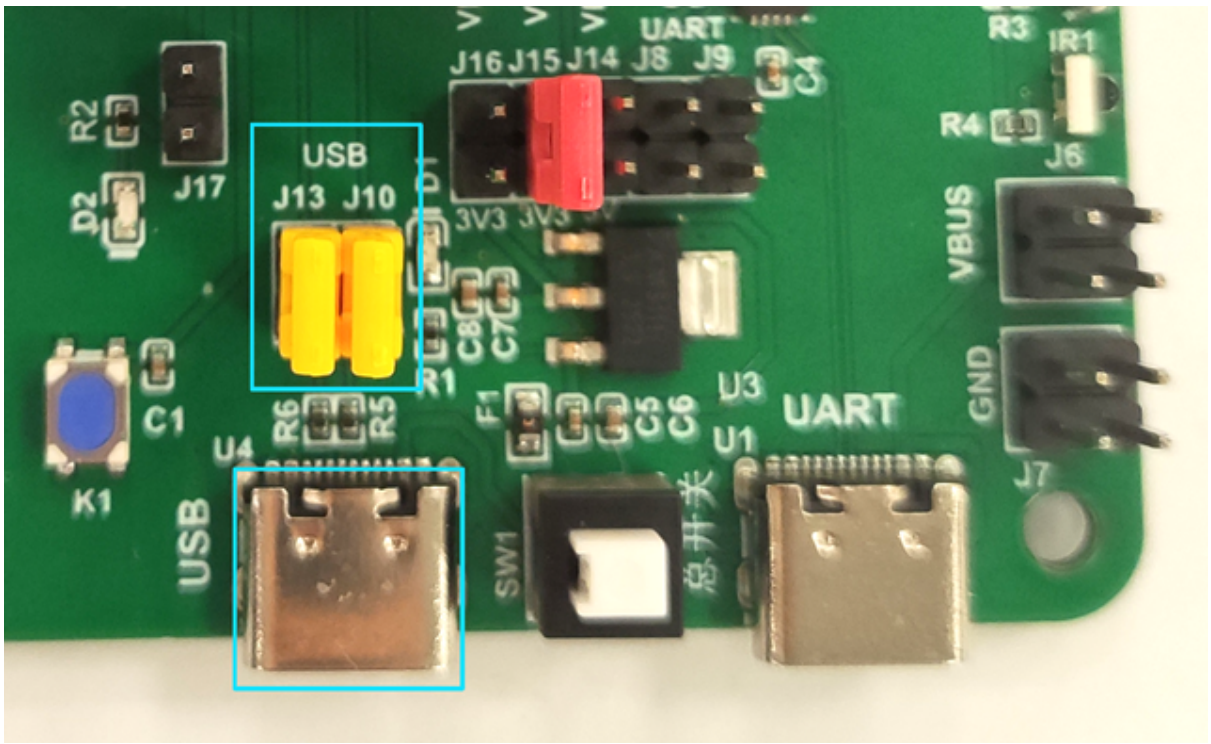

图 11: USB 模块外围电路实物图

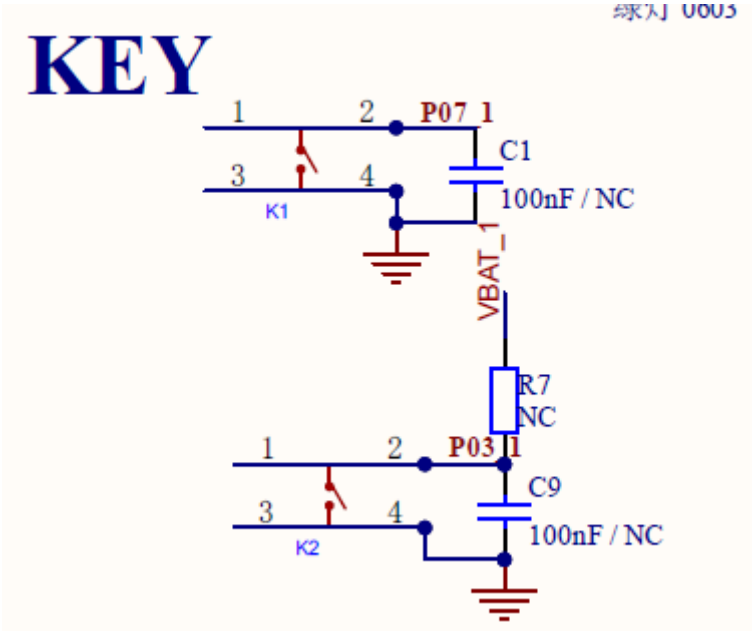

图 12: 独立按键原理图

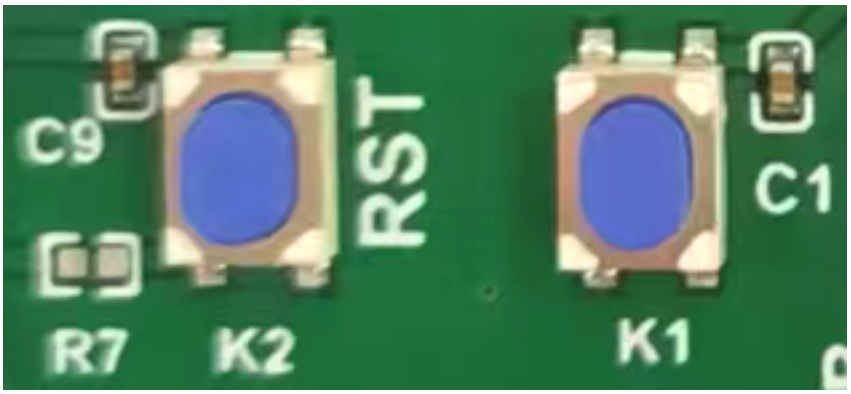

图 13: 独立按键实物图

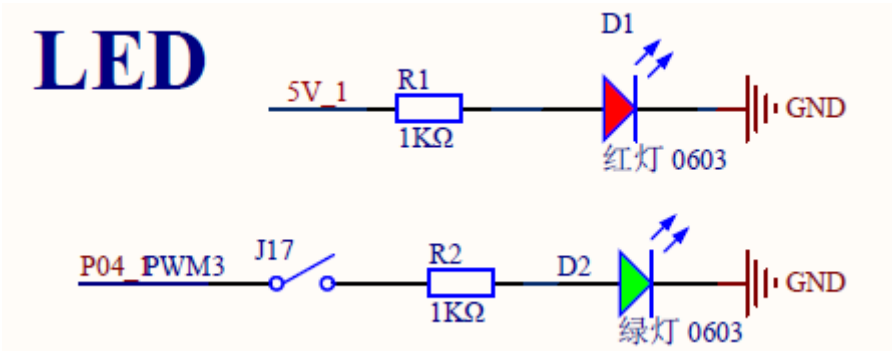

图 14: LED 灯模块原理图

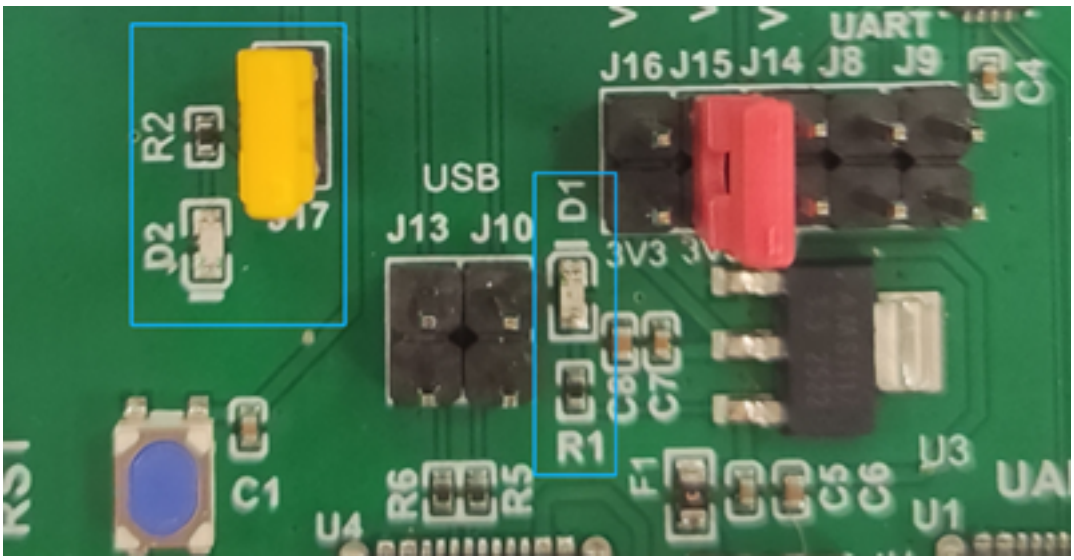

图 15: LED 灯模块实物图

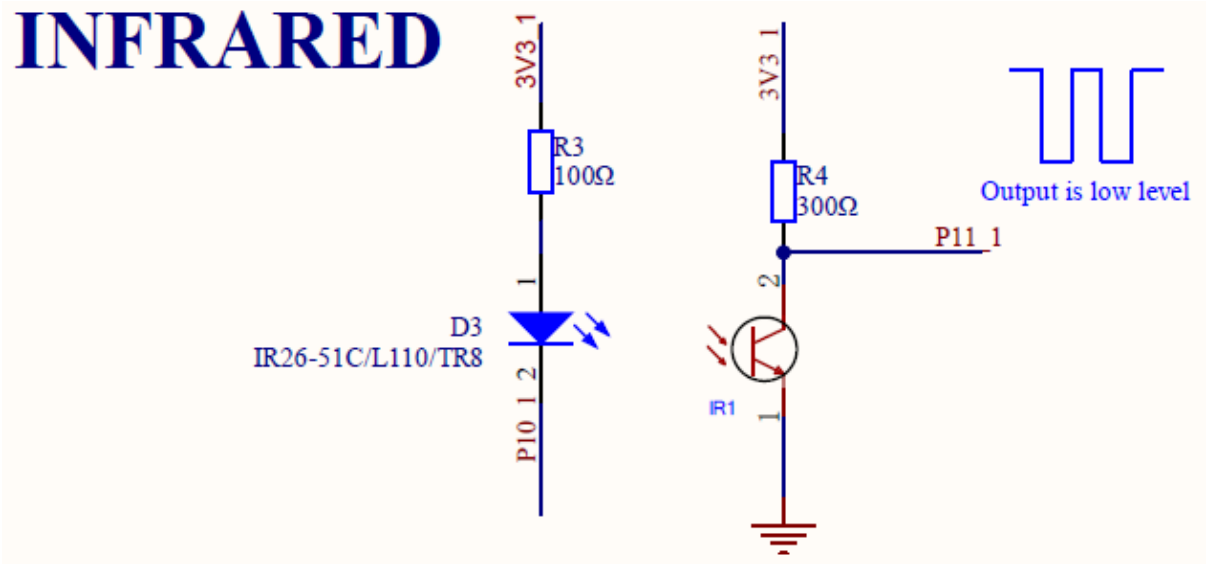

图 16: 红外模块原理图

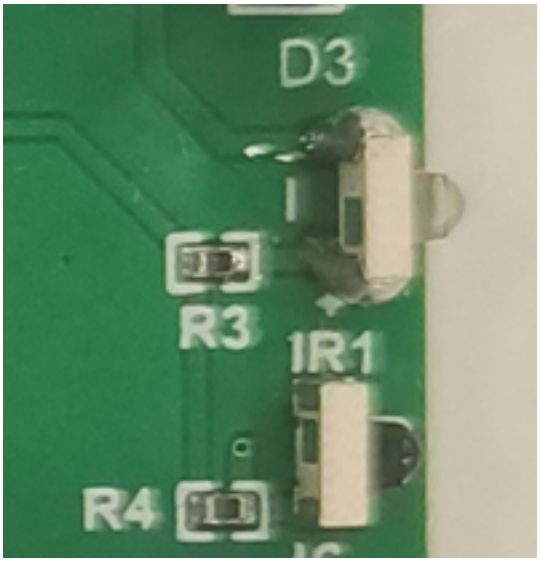

图 17: 红外模块实物接线图

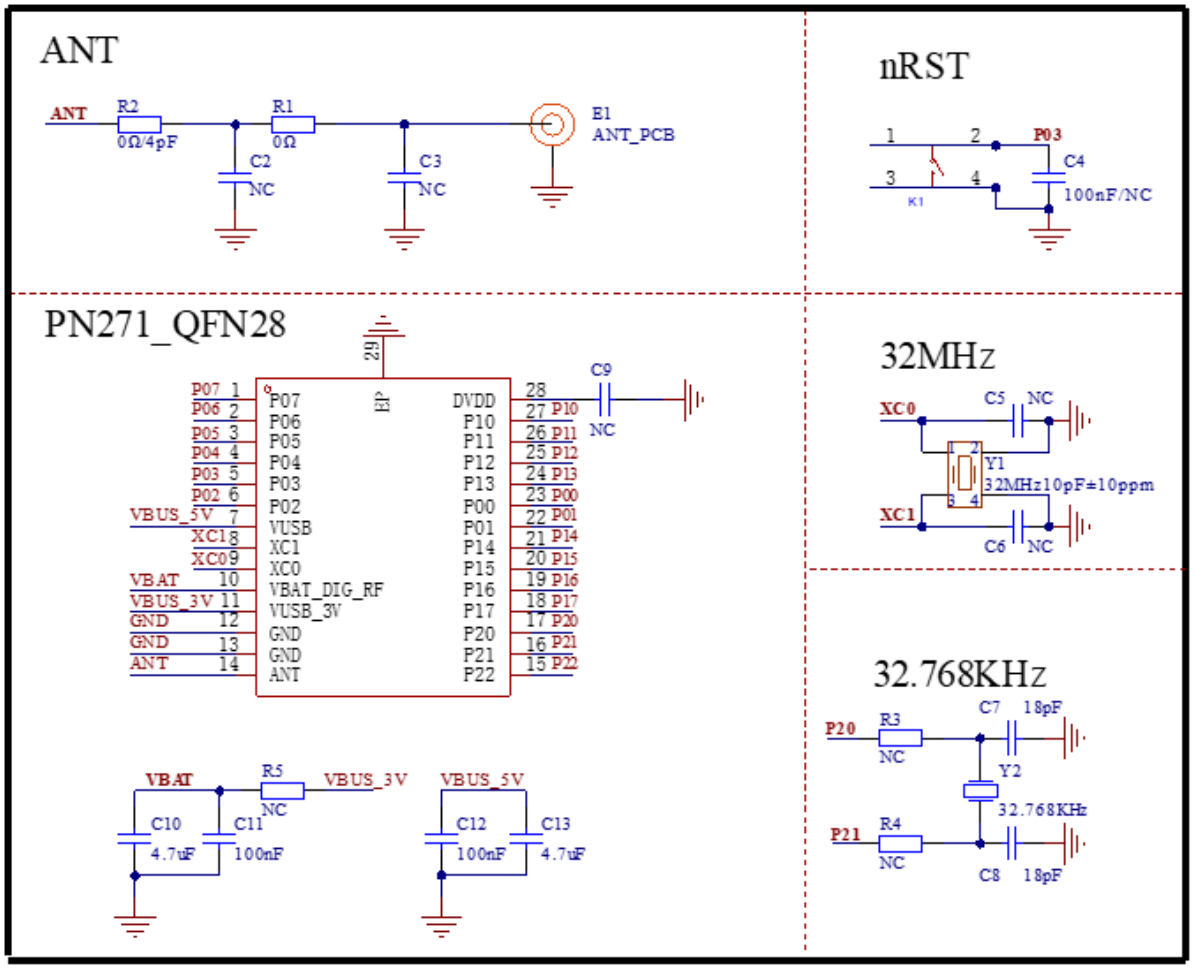

图 18: PAN2710U5GA（QFN28）最小系统参考原理图

位号	参数	封装	描述	数量
C11, C12	100nF	0402	NP0, ±10%, 16V	2
C7, C8	18pF	0402	NP0, ±10%, 16V	2
C10, C13	4.7uF	0402	NP0, ±10%, 16V	2
K1	K2-1808GN-A4SW-04	SMD_4P	4 脚贴片按键	1
R1	0Ω	0402	0Ω 1%	1
R2	0R/4pF	0402	PIFA 天线需串电容	1
U1	PAN271x	QFN28		1
Y1	32MHz	3225	频率偏差±10ppm; 负载电容 10pF	1
Y2	32.768KHz	3215	频率偏差±20ppm; 负载电容 12.5pF	1
R3, R4, R5	NC	0402	--	--
C2, C3, C4, C5, C6, C9	NC	0402	--	--

图 19: BOM 元器件清单

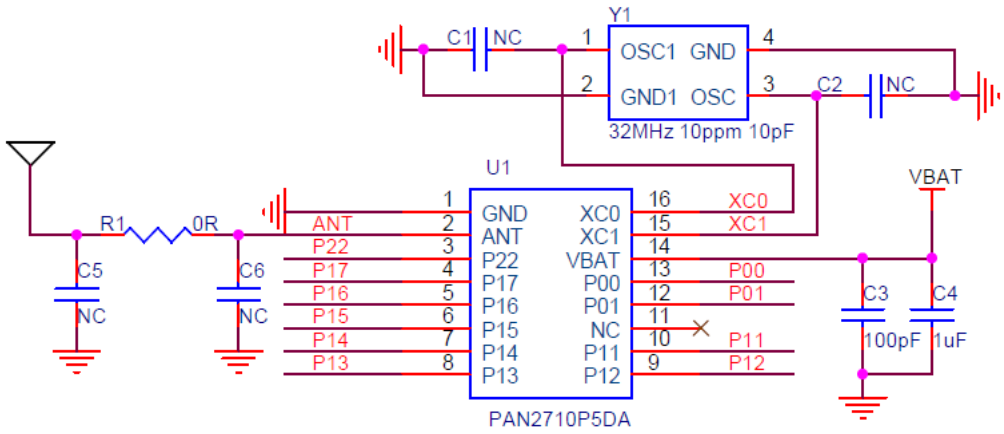

图 20: PAN2710P5DA（SOP16）最小系统参考原理图

位号	参数	封装	描述	数量
C3	100nF	0402	NP0, ±10%, 16V	1
C4	1uF	0402	NP0, ±10%, 16V	1
R1	0R/4pF	0402	0Ω 1%/ PIFA 天线需串电容	1
C1, C2, C5, C6	NC	0402	NP0, ±10%, 16V	--

图 21: BOM 元器件清单

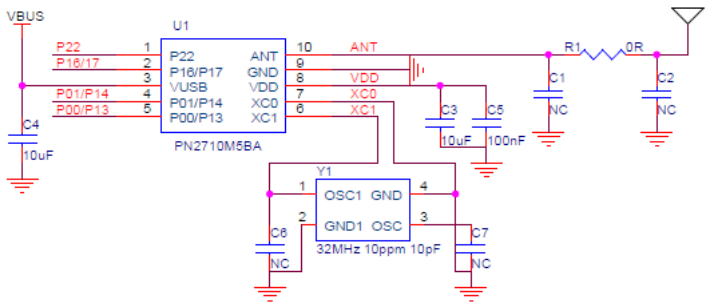

图 22: PAN2710M5BA（MSOP10）最小系统参考原理图

位号	参数	封装	描述	数量
C3, C4	10uF	0402	NP0, ±10%, 16V	2
C5	100nF	0402	NP0, ±10%, 16V	1
R1	0R/4pF	0402	0Ω 1%/ PIFA 天线需串电容	1
C1, C2, C6, C7	NC	0402	NP0, ±10%, 16V	--

图 23: BOM 元器件清单

1.2 电源电路

电源设计的完整性影响芯片性能，好的电源设计更容易发挥无线模块的性能。电压范围 1.8-3.6V，纹波小于 ±100mV，频率小于 1MHz。电源设计需要留有裕量，一般来说，在条件允许的情况下，输出电流能力需要大于峰值电流的 2 倍。如果电流裕量有限，至少也需要 1.5 倍峰值电流以上。在 3.3V 供电系统中，过大的纹波可能通过导线或者地平面耦合到系统容易受到干扰的线路上，例如天线、馈线、时钟线等敏感信号线上，导致模块的射频性能变差。

电源供电有两种方式:

1. VBUS_5V 供电 5V，R5 焊接 0Ω（使 VBUS_3V 和 VBAT 短接）。芯片内部有 LDO，会将 5V 转成 3.3V，从 VBUS_3V 输出，参考 QFN28 原理图。SOP16 没有 VBUS 管脚。
2. VBAT 供电 3.3V，R5 不焊接，参考 QFN28 原理图。MSOP10 封装 VBUS 供电 5V，内部 3.3V 会接到 VDD 管脚，或单独在 VDD 供电 3.3V。

1.3 晶体电路

1.3.1 XTH 晶体推荐参数

1. 晶体频率：32MHz 或 16MHz
2. ESR：小于 80 欧姆
3. 晶体负载电容：10pF
4. 频率偏差：±20ppm 以内

1.3.2 XTL 晶体推荐参数

1. 晶体频率：32.768KHz
2. ESR：小于 40KΩ
3. 晶体负载电容：12pF
4. 频率偏差：±20ppm 以内

1.4 天线匹配电路

天线匹配根据是否需要过 FCC/CE 认证来确定元器件参数。没有安规要求，建议匹配结构也预留（防止模组功率偏低需匹配优化），用 0 欧姆电阻串联到天线，PIFA 天线需串联 4pF 左右隔直电容到天线。

2.2.2 2 PCB 设计

2.1 PCB 板材和叠层设计

板材建议优先选择双面 FR4 板材结构，最终叠层结构根据实际产品来确定。

2.2 电源和地线 Layout

1. 电源线宽度尽量粗，尽量 20mil 以上。电源线必须先经过电容再到芯片电源输入管脚，在靠近芯片电源引脚放两个并联电容对电源低通滤波，其中小容值电容摆放在更靠近芯片引脚的位置，以便较好地滤除高频噪声。滤波电容接地保证较好的回流路径，双面板可以在地 PAD 就近打 VIA 减小回流路径。
2. 建议电源线和地线采用放射状的连接方式，单点接电源/地并且单独走线，RF 芯片的电源/地线走线与其它芯片或器件分开，从总参考电源/地线单独引线，防止受到干扰，铺地推荐使用实心地。
3. 铺地的地线建议与噪声较少的地线或者总参考地线连接，不与强信号或者强干扰器件地线相连，可以有效地减少整个印制板的工作噪声。

2.3 晶体电路 Layout

1. 晶体至芯片管脚的走线尽量加粗，尽量短，不能走过孔
2. 直插晶体的焊盘需要保证外径与内径差值有 0.2mm 以上
3. 印制板上在晶体焊盘和走线的两边有完整地平面，最好不要有任何走线和元器件
4. 为了避免干扰射频信号，晶体尽量远离射频走线
5. 为避免大功率发射时，晶体受到辐射功率干扰，导致 RF 工作异常，晶振电路包括负载电容部分远离天线电路。PCB 天线部分与晶体电路之间尽量有地作为隔离带。

2.4 射频匹配电路 Layout

射频匹配电路对射频性能影响很大，因此需要特别注意。匹配元器件推荐用 0402 封装，匹配结构按参考原理图设计。射频匹配 layout 原则：

1. 防止射频前端能量损耗，从 ANT 管脚到天线匹配电路的走线尽量短，阻抗等于 50 欧。天线 PI 型匹配电路要走顺，并联元件焊盘和走线重合为好，射频走线禁止打过孔换层。
2. 射频走线宽度根据匹配器件是 0402/0603 封装适当调整。线宽控制在 0.5~1mm 之间。避免走线宽度和器件 pad 大小不一致，影响阻抗连续性。
3. 射频走线两边需要良好铺地（多层板多打过孔）。铺地与射频走线间距，根据制板工艺，控制在 0.2~0.4mm 之间，同时满足 50 欧阻抗匹配。射频匹配部分对应背面，需要有完整的参考地平面，避免放置元器件和走线。
4. 芯片有 EPAD，RF 参考地和 EPAD 需要良好连接，多层板需要在 EPAD 上打 4 个以上过孔与底层地连接
5. 为了方便调试天线，ANT Pin 和 PI 型匹配之间可以串联 0R 电阻，电阻旁可以露一块 GND 属性铜皮。

2.5 天线 Layout

天线设计请参考 “2.4G PCB 天线设计指南”，文档请找技术支持获取。

PIFA 天线不能与地线铺铜靠很近，至少留空 1mm。天线部分对应的底层 PCB 严禁铺地。天线与参考地线铺铜间距要大于 1mm。天线周边最好不要有金属结构或元器件及走线，保证在 PCB 上间隔至少 3cm 范围内不摆放较大的带金属材质元器件。导线天线的馈点周围需要净空，净空区域要求 2mm 以上。

2.6 ESD 防护

1. 敏感信号线需添加 ESD 保护管（常见 TVS）。TVS 摆放位置应尽可能靠近 ESD 源头（接头等处），与被保护 IC 的距离要远于 ESD 源。布线时需将 ESD 源直接接到 TVS，减少 TVS 管和回流地之间的寄生电感。
2. 布线时，让敏感信号线远离 PCB 板边。为避免走线与天线间的串扰，走线需远离天线。

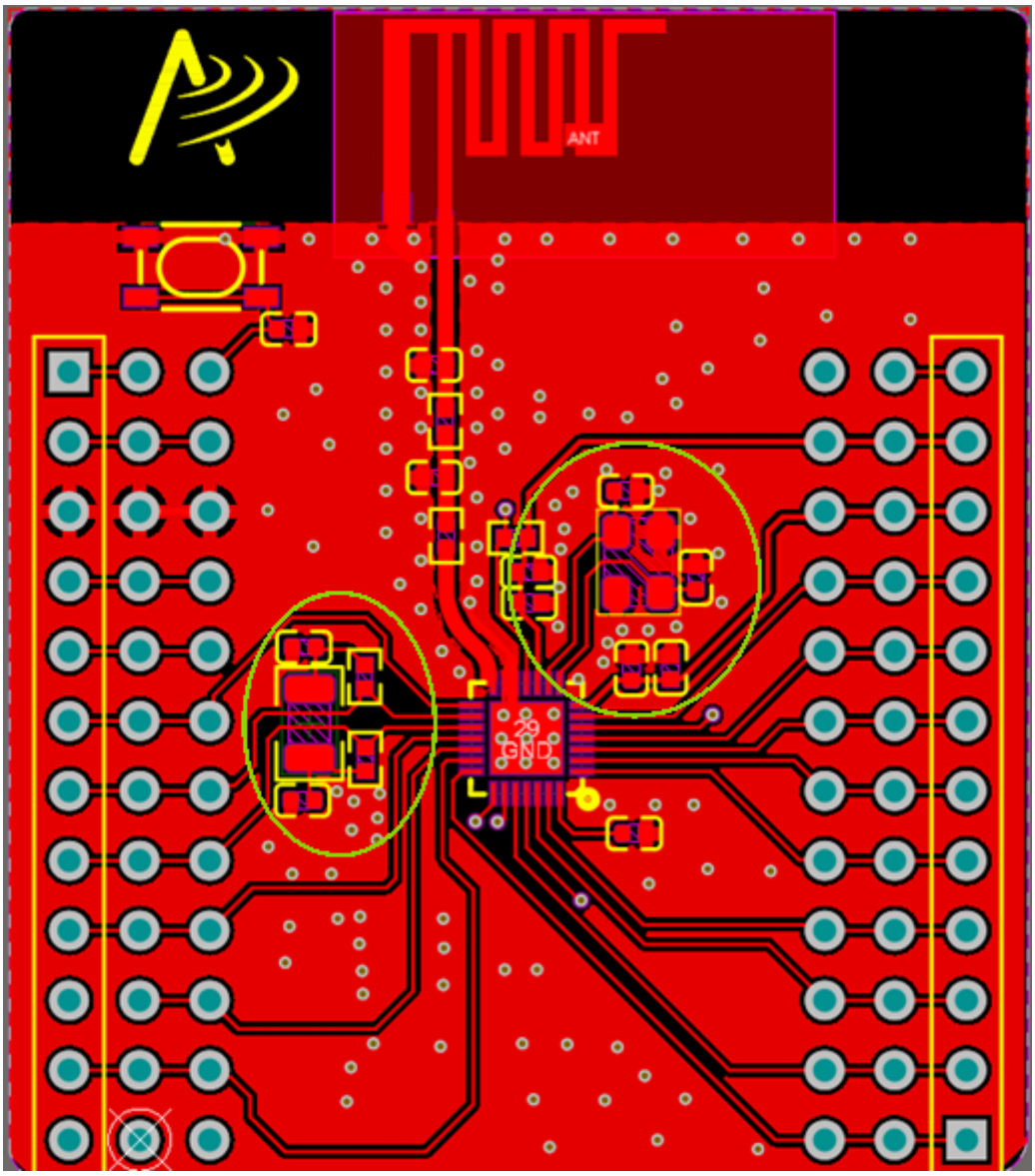

图 24: 晶体布局示意图

3. 删除孤岛铜皮，用地将敏感信号包裹起来，降低干扰信号的辐射影响。
4. 尽量增大过孔的钻孔直径和焊盘直径，减少过孔的寄生电感。
5. 尽量缩短线长以减少寄生电感。因直角走线会产生更大的电磁辐射，避免直角走线连接到器件或走线上。

2.7 PCB Layout 示例

下面是 PAN2710U5GA（QFN28）EVB 核心板布局示意图：

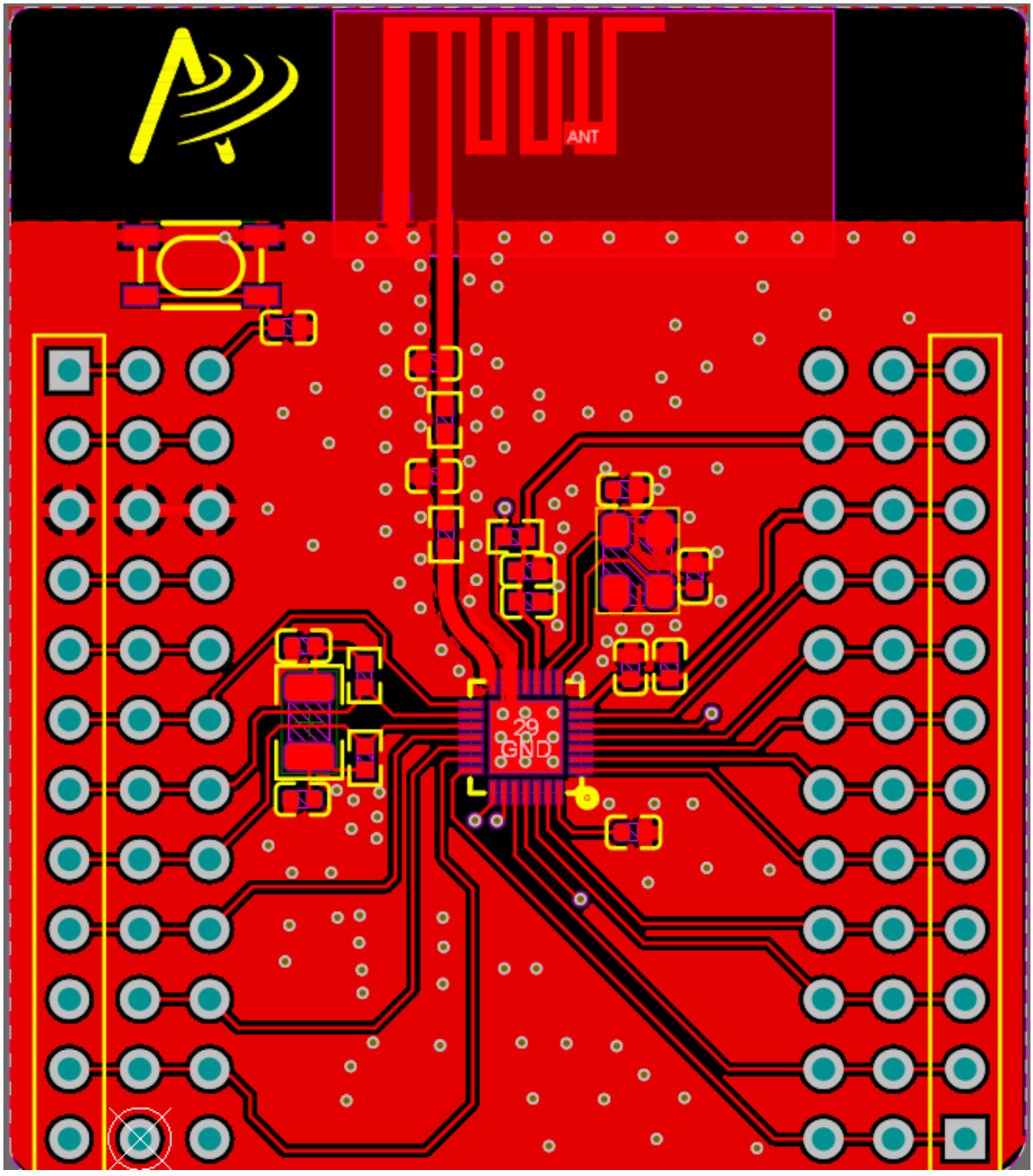

图 25: PAN2710U5GA（QFN28）EVB 核心板双面板 PIFA 天线 Top 层例

HDK 内容：

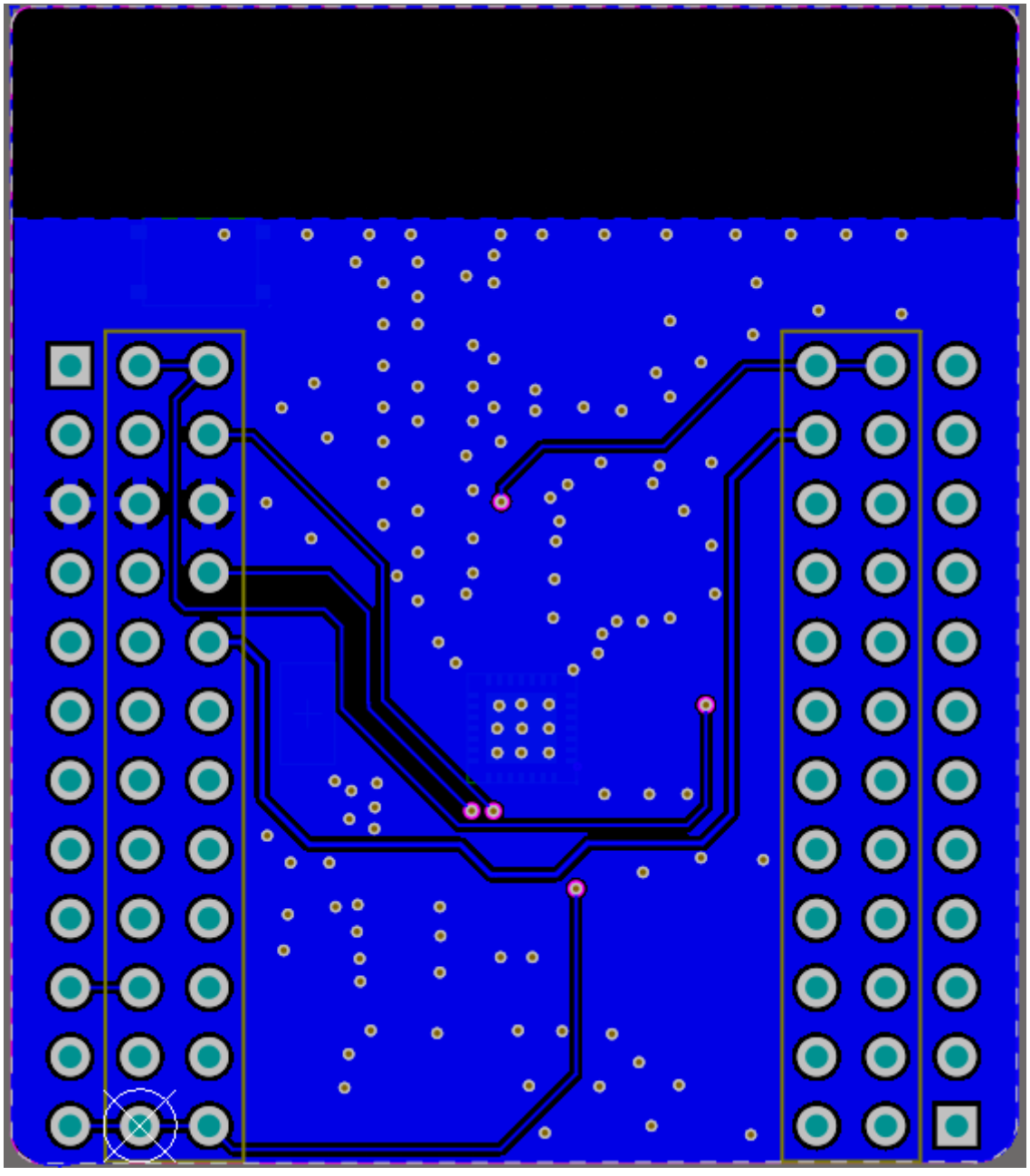

图 26: PAN2710U5GA（QFN28）EVB 核心板双面板 PIFA 天线 Bottom 层例

硬件开发资料（HDK）位于: `<PAN271X-DK>\02_HDK`，其包含如下内容:

02_HDK **子目录**	**包含内容**
PAN271x_BaseBoard_V1.0	PAN271x EVB 底板硬件设计资料（原理图、PCB 文件等）和生产资料（BOM、gerber、坐标等文件）
PAN271x_QFN28_CoreBoard_V1.0	PAN271x QFN28 核心板硬件设计资料（原理图、PCB 文件等）和生产资料（BOM、gerber、坐标等文件）

Chapter 3

演示例程

3.1 外设驱动例程

3.1.1 ADC

1 功能概述

本例程演示演示 ADC Driver 的基本功能与使用方法。

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 连接串口转 USB 调试模块：
    - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
    - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
  - 连接 Jlink，使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连

3 编译和烧录

例程位置：`<PAN271x-DK>\01_SDK\samples\drivers\adc`

双击 Keil Project 文件打开工程进行编译烧录。

本节介绍 ADC 例程的主要流程、核心代码及常用 API 说明，帮助用户快速理解和移植。

4 编译和烧录

4.1 主要流程代码（main.c 节选）

```
int main(void)
{
    platform_init();
    LOG("\nCPU @ %dHz\n", SystemCoreClock);
    ADC_Init();
    while(1)
    {
        systick_delay_ms(300);
        LOG("Vbat Voltage: %d mV\n", ADC_GetVbatVoltMv());        // 获取 Vbat 电压值

        systick_delay_ms(300);
        LOG("External Channel Code: %d\n", ADC_GetCodeByVbat(P1, GPIO_Pin_3));    // 获取外部通
道 Code 值，使用 Vbat 作为参考电压

        systick_delay_ms(300);
        LOG("External Channel Voltage: %d mV\n", ADC_GetVoltByVbat(P1, GPIO_Pin_3));  // 获取外
部通道电压值，使用 Vbat 作为参考电压
    }
}
```

4.2 主要 API 说明

- `ADC_Init()`：初始化 ADC 模块，设置默认值和 ADC 校准值。
- `ADC_GetVbatVoltMv()`：获取 Vbat 电压（单位：mV）。

VDD 档位电压采集说明 这里的“VDD 档位”指 ADC **以芯片供电电压** VDD **（也常被称为** VBAT**）作为参考电压**对外部通道进行采样。相关 API 如下：

- `ADC_GetCodeByVbat(port, pin)`：在 VDD 作为参考电压时，采集指定外部通道并返回 Code。函数内部会将对应 GPIO 临时配置为模拟输入，采样完成后恢复为数字模式；并结合 OTP 校准参数 `adc_vbat_k/b` 对结果做增益/偏移补偿。
- `ADC_GetVoltByVbat(port, pin)`：获取指定外部通道电压（单位：mV）。其内部流程为：先调用 `ADC_GetCodeByVbat()` 获取 Code，再调用 `ADC_GetVbatVoltMv()` 获取当前 VDD，最后按比例换算为 mV。

使用示例：

```
uint16_t code = ADC_GetCodeByVbat(P1, GPIO_Pin_3);
uint16_t volt = ADC_GetVoltByVbat(P1, GPIO_Pin_3);
```

VBG 档位电压采集说明 ADC 支持以内部带隙基准（VBG）作为参考电压进行采集，适用于高精度测量场景。相关 API 如下：

- `ADC_GetVoltByVbg(port, pin)`：VBG 作为参考电压时，获取外部通道的电压值（单位：mV）。

使用示例：

```
uint16_t volt = ADC_GetVoltByVbg(P1, GPIO_Pin_3); // 获取 P1.3 通道的电压值，参考电压为 VBG
```

4.3 串口输出示例

```
CPU @ 48000000Hz
Vbat Voltage: 3300 mV
External Channel Code: 1234
External Channel Voltage: 1100 mV
```

4.4 典型应用说明

- 通过修改 `ADC_GetCodeByVbat` 和 `ADC_GetVoltByVbat` 的引脚参数，可采集不同引脚的模拟信号。
- 若需采集温度或其他内部信号，请参考 SDK 中 ADC 相关 API 文档及头文件注释。

5 ADC 校准值说明（otp_adc_trim_data）

ADC 驱动内部使用 `otp_adc_trim_data` 结构体存储校准参数，包括 ADC 在不同参考电压（Vbat/VBG）下的系数 k、b 等。其作用如下：

- **校准值来源**：
  - 从芯片 OTP 区域读取（由工厂烧录），每块芯片 ADC 校准值都不同。`otp_adc_trim_data` 结构体默认值为全 0，在 debug 阶段，可以给出临时的校准值，在烧录 otp 的时候可以再注释掉：

```
// ADC 临时校准值
static OTP_AdcTrimData_T otp_adc_trim_data = {
    .adc_cal_vbat_k = 11529,
    .adc_cal_vbat_b = 192,
#if ADC_VDD_VOL_ENABLE
    .adc_vbat_k = 9955,
    .adc_vbat_b = 178,
#endif
};
```

- **主要成员含义**：
  - `adc_cal_vbat_k` / `adc_cal_vbat_b`：VBG 作参考电压时，通过 1/4Vdd 计算 Vbat 的 k、b。
  - `adc_vbat_k` / `adc_vbat_b`：Vbat 作参考电压时，外部通道电压的 k、b。
  - `adc_vbg_ref_cal_volt` / `adc_vbg_kb`：VBG 档位下分档的参考电压和 k、b。
- **实际应用建议**：
  - 推荐使用 OTP 烧录的校准值以获得更高精度，若无 OTP 校准则默认值也可满足临时调试需求。

3.1.2 GPIO

1 功能概述

本例程演示演示 GPIO Driver 的基本功能与使用方法。

例程主要完成以下内容：

- 配置 `P0.4` 为 GPIO 推挽输出，周期性输出高/低电平（用于驱动 LED，需要将 P0.4 连接到 J17）
- 初始化串口 `UART0`（115200）并打印日志，方便观察 GPIO 输出变化
- 配置 `P0.7` 为 GPIO 输入并使能上拉（例程中提供了“按键控制 LED”的注释代码，可按需打开）

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）

- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
  - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
  - 使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连
  - GPIO 演示 IO：
    - `P0.4`：GPIO 输出脚（例程用于驱动 LED）
    - `P0.7`：GPIO 输入脚（例程已配置上拉，按键逻辑在注释代码中）
    - `P0.4` 连接到 `J17` 用于驱动 LED
- PC 软件:
  - 串口调试助手（UartAssist）或终端工具（SecureCRT），波特率 115200（用于串口交互）

3 编译和烧录

例程位置：`<PAN271x-DK>\01_SDK\samples\drivers\gpio`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

1. 烧录完成后，连接 P04 和 J17；
2. P04 会循环输出高电平和低电平，可观察到 LED 灯时亮时灭；

4.1 主要流程代码（main.c 节选）

```
void LED_GPIOInit(void)
{
  SYS_SET_MFP(P0, GPIO_Pin_4, GPIO);
  GPIO_SetMode(P0, GPIO_Pin_4, GPIO_MODE_PushPull);

  SYS_SET_MFP(P0, GPIO_Pin_7, GPIO);
  GPIO_SetMode(P0, GPIO_Pin_7, GPIO_MODE_INPUT);
  GPIO_EnablePullupPath(P0, GPIO_Pin_7);
}

int main()
{
  Clock_Init();
  Sys_Init();
  LOG("\nCPU @ %dHz\n", SystemCoreClock);
  LED_GPIOInit();

  while (1)
  {
    SYS_delay_10nop(1000000);
    GPIO_WriteOutputBit(P0, GPIO_Pin_4, 1);
    LOG("LED on\n");
```

(下页继续)

(续上页)

```
    SYS_delay_10nop(1000000);
    GPIO_WriteOutputBit(P0, GPIO_Pin_4, 0);
    LOG("LED off\n");
  }
}
```

4.2 主要 API 说明

- `SYS_SET_MFP(Px, GPIO_Pin_x, GPIO)`：将对应引脚的复用功能切换为 GPIO。
- `GPIO_SetMode(Px, GPIO_Pin_x, mode)`：设置 GPIO 工作模式。
  - 输入：`GPIO_MODE_INPUT`
  - 输出：`GPIO_MODE_PushPull`（推挽输出）
  - 输出：`GPIO_Mode_OpenDrain`（开漏输出）
  - 双向检测：`GPIO_MODE_QUASI`
- `GPIO_WritePin(Px, pin, val)`：以宏方式直接写输出寄存器（例程中给出了替代写法的注释）。
- `GPIO_ReadPin(Px, pin)`：读取输入电平（例程中给出了“按键控制 LED”的注释逻辑）。
- `GPIO_EnablePullupPath(Px, pin)`：为输入引脚使能上拉。
- `GPIO_EnablePulldownPath(Px, pin)`：为输入引脚使能下拉。

4.3 串口输出示例

```
CPU @ 48000000Hz
LED on
LED off
LED on
LED off
...
```

4.4 注意事项

- 使用 GPIO 之前请确保引脚复用已切到 GPIO（`SYS_SET_MFP`），否则可能仍在外设功能下导致读写无效。
- 驱动 LED 时，建议使用推挽输出（`GPIO_MODE_PushPull`）；若外部电路需要开漏方式，请改为 `GPIO_Mode_OpenDrain` 并外接上拉。
- 输入脚如果悬空会导致电平不稳定，建议按需开启上拉/下拉（例程对 `P0.7` 使能了上拉）。

3.1.3 Low Power

1 功能概述

本例程默认演示 Deepsleep **模式**的进入与唤醒：

- 使用 `LP_EnterDeepSleep()` 进入 Deepsleep
- 例程代码中提供了 GPIO **唤醒口**、SleepTimer(32k) **唤醒和** WDT（**看门狗**）**唤醒**的配置示例

2 环境准备

- 硬件设备与线材:
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
  - 逻辑分析仪，用于抓取 debug io 状态
- 硬件接线:
  - 将 EVB 核心板插到底板上
  - 连接串口转 USB 调试模块:
    - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
    - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
  - 连接 Jlink，使用杜邦线将 JLink 仿真器的:
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连
- PC 软件:
  - 串口调试助手（UartAssist）或终端工具（SecureCRT），波特率 115200（用于串口交互）

3 编译和烧录

例程位置: `<PAN271x-DK>\01_SDK\samples\drivers\lowpower`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

4.1 可选唤醒源: GPIO 唤醒口

例程中提供了 `LP_WakeUpPinConfig()`，用于把某个 GPIO 配置成上升沿唤醒源。若要启用 GPIO 唤醒:

1. 取消 main 循环中如下语句的注释:

```
LP_WakeUpPinConfig(P1, GPIO_Pin_6);   /* 配置 P1.6 为唤醒口 */
```

2. 将对应引脚接到可产生上升沿的外部信号。

唤醒后会进入对应端口的 GPIO 中断服务函数（例程实现了 `GPIO1_IRQHandler`），并打印:

```
P1 interrupt
```

4.2 可选唤醒源: SleepTimer(32k) 定时唤醒

例程中提供了 SleepTimer 的配置示例，用于在 Deepsleep 下按 32k 时钟计数定时唤醒:

```
LP_SetUsrSleepTimer(32000 * 3, 1);   /* 设置睡眠时间为 3 秒（约）*/
LP_EnableDeepsleepTimer1();          /* 使能睡眠定时器 1 作为唤醒源 */
```

当 SleepTimer 触发时，会进入 `SLPTMR1_IRQHandler`，并打印:

```
slptmr1 int
```

4.3 可选唤醒源：WDT（看门狗）唤醒

1. 打开看门狗配置：

```
/* 配置 WDT 唤醒功能：关闭看门狗复位，打开唤醒功能 */
SYS_UnlockReg();
WDT_Open(WDT_TIMEOUT_2POW14, WDT_RESET_DELAY_1025CLK, FALSE, TRUE);
WDT_EnableInt();
NVIC_EnableIRQ(WDT_IRQn);
SYS_LockReg();
```

2. 进入 Deepsleep 并等待 WDT 唤醒：

   例程在循环中进入 Deepsleep，WDT 超时后会把芯片从 Deepsleep 唤醒。可以参考如下写法：

```
while (true)
{
    /* 低功耗前可按需关闭 UART RX 数字通路降低漏电（按实际 RX 引脚调整） */
    GPIO_DisableDigitalPath(P0, GPIO_Pin_6);
    LP_EnterDeepSleep();
    GPIO_EnableDigitalPath(P0, GPIO_Pin_6);

    /* 唤醒后刷新 WDT 计数，避免很快再次超时 */
    WDT_ResetCounter();
    SYS_delay_10nop(100);                // 延迟 66us
}
```

3. 唤醒后现象与中断处理：

   - 当 WDT 超时发生，会进入 `WDT_IRQHandler()`，例程会打印：

```
WDT interrupt
```

   - 为避免中断/唤醒标志残留导致重复进入中断，例程在 `WDT_IRQHandler()` 中会清除：
     - Timeout Interrupt Flag
     - Timeout Flag
     - Timeout Wakeup Flag

注意事项：

- `WDT_Open(..., enableReset=FALSE, enableWakeup=TRUE)` 这种配置是“只唤醒不复位”。如果把 `enableReset` 配成 `TRUE`，WDT 超时将导致复位而不是按当前流程继续执行。
- 如果修改了 WDT 超时时间（`WDT_TIMEOUT_xxx`），记得同步评估实际唤醒周期；周期越长，便于观察但响应会更慢。
- WDT_ResetCounter() **喂狗完毕后，至少要等待** 66us **以上，才能再次进入低功耗模式。建议先喂狗再进行其他操作后进入低功耗模式，其他操作要保证在** 66us **以上，这样** CPU **可以手动空转等待** 66us。

4.4 主要 API 说明

- `LP_EnterDeepSleep()`：进入 Deepsleep 模式。
- `WDT_Open(timeout, resetDelay, enableReset, enableWakeup)`：配置 WDT。
  - 例程将 `enableReset` 置为 `FALSE`（不复位），将 `enableWakeup` 置为 `TRUE`（作为唤醒源）。
- `LP_WakeUpPinConfig(GPIO_T *Port, GPIO_PinDef pin)`：配置 GPIO 唤醒源。
- `LP_SetUsrSleepTimer(sleepTime, idx)`：设置用户睡眠定时器比较值（单位为 32k 时钟周期）。

- `LP_EnableDeepsleepTimer1()`：使能 SleepTimer1，并打开中断与唤醒能力。
- `GPIO_DisableDigitalPath()/GPIO_EnableDigitalPath()`：关闭/恢复数字输入通路，常用于降低低功耗漏电。

4.5 低功耗调试建议

- 若目标是评估低功耗电流，建议断开不必要的外设连接（如逻辑分析仪、某些调试接口），并按板卡电路实际情况关闭相关引脚的数字通路/上下拉。
- 进入低功耗前可关闭串口接收数字通路，防止漏电（根据接收串口引脚更改）。例如例程默认 UART0_RX 为 `P0.6`，进入低功耗前调用 `GPIO_DisableDigitalPath(P0, GPIO_Pin_6)`，唤醒后再恢复 `GPIO_EnableDigitalPath(P0, GPIO_Pin_6)`。
- 例程中对下载口相关引脚（`P0.0/P0.1`）关闭数字通路与上拉，是为了降低待机漏电。

3.1.4 PWM

1 功能概述

本例程演示 PAN271x PWM Driver 的基础用法：配置两路 PWM 输出并启动（独立模式 / Independent Mode）。

- `PWM_CH1`：100 kHz，占空比 30%，输出到 `P1.7`（板级常标为 `P17`）
- `PWM_CH4`：200 kHz，占空比 50%，输出到 `P1.2`（板级常标为 `P12`）

例程代码入口：`samples\drivers\pwm\src\main.c`

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
  - 逻辑分析仪或示波器（用于观察 PWM 波形）
- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
  - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
  - 使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连
  - PWM 波形观测：
    - `P17（P1.7 / PWM_CH1）`接逻辑分析仪/示波器
    - `P12（P1.2 / PWM_CH4）`接逻辑分析仪/示波器
    - 观测设备地线接 EVB GND

- PC 软件：
  - 串口调试助手或终端工具，波特率 115200（8N1）

3 编译和烧录

- 例程位置：`<SDK_ROOT>\samples\drivers\pwm`
- Keil 工程：`keil\pwm.uvprojx`

双击 Keil Project 文件打开工程，编译并下载到芯片。

4 例程演示说明

4.1 串口输出示例

烧录完成并复位后，例程会初始化时钟与 UART0，并打印当前 CPU 主频：

```
CPU @ 48000000Hz
```

说明：本例程无交互菜单，不需要串口输入；上电后会持续输出 PWM 波形。

4.2 波形验证（逻辑分析仪/示波器）

- 在 P17（PWM_CH1）观察到约 100 kHz PWM，duty 约 30%
- 在 P12（PWM_CH4）观察到约 200 kHz PWM，duty 约 50%

若频率存在轻微偏差，通常与 PWM 分频/预分频取整有关（驱动会选择“最接近”的可达频率）。

4.3 主要流程代码（main.c 节选）

```
void PWM_GPIOInit(void)
{
    // Enable Clock
    CLK_APB1PeriphClockCmd(CLK_APB1Periph_PWM0_CH01, ENABLE);
    CLK_APB1PeriphClockCmd(CLK_APB1Periph_PWM0_CH23, ENABLE);
    CLK_APB1PeriphClockCmd(CLK_APB1Periph_PWM0_CH45, ENABLE);

    // Config Pinmux
    SYS_SET_MFP(P1, GPIO_Pin_7, PWM_CH1);
    SYS_SET_MFP(P1, GPIO_Pin_2, PWM_CH4);
}

int main(void)
{
    Clock_Init();
    Sys_Init();
    LOG("\nCPU @ %dHz\n", SystemCoreClock);

    PWM_GPIOInit();

    PWM_ConfigOutputChannel(PWM, PWM_CH1, 100000, 30);
    PWM_ConfigOutputChannel(PWM, PWM_CH4, 200000, 50);

    PWM_EnableIndependentMode(PWM);
    PWM_EnableOutput(PWM, BIT(PWM_CH1) | BIT(PWM_CH4));
    PWM_Start(PWM, BIT(PWM_CH1) | BIT(PWM_CH4));

    while (1);
}
```

4.4 主要 API 说明

- `PWM_ConfigOutputChannel(PWM, ch, freq, duty)`：按目标频率/占空比配置通道输出（边沿对齐自动重装载模式）。
- `PWM_EnableIndependentMode(PWM)`：独立模式，各通道按自身配置输出。
- `PWM_EnableOutput(PWM, mask)`：使能对应通道输出引脚。
- `PWM_Start(PWM, mask)`：启动对应通道计数输出。

4.5 常见修改

- 修改频率/占空比：直接改 `PWM_ConfigOutputChannel()` 的 `freq` 与 `duty` 参数。
- 增加/更换输出引脚：修改 `PWM_GPIOInit()` 中 `SYS_SET_MFP()` 的引脚复用配置，并同步修改 `PWM_EnableOutput()` / `PWM_Start()` 的通道掩码。

3.1.5 TIMER

1 功能概述

本例程演示 Timer Driver 的基本用法：

- 选择 Timer0 的时钟源
- 配置为周期计数模式（Periodic）
- 设置比较值（Compare Value）产生超时事件
- 使能 NVIC 与 Timer 中断，在中断服务函数中清除标志并打印 Log

例程代码入口：`samples\drivers\timer\src\main.c`

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 连接串口转 USB 调试模块：
    - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
    - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
    - 本例程使用 UART0：P0.5 为 TX，P0.6 为 RX
  - 连接 Jlink，使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连
  - 使用杜邦线或跳线帽将 EVB 底板 `J13` 排针对 (P13) 和 `J10` 排针对 (P14) 分别短接起来

3 编译和烧录

例程位置：<PAN271x-DK>\01_SDK\samples\drivers\timer

双击 Keil Project 文件打开工程，编译并下载到芯片。

4 例程演示说明

4.1 串口输出示例

烧录完成并复位后，例程会启动 Timer0 并打印：

```
Start Timer0.
```

随后进入 Timer0 中断并周期性输出：

```
TIMER0 interrupt.
TIMER0 interrupt.
TIMER0 interrupt.
...
```

说明：本例程无交互菜单，不需要串口输入。

4.2 主要流程代码（main.c 节选）

```
void TMR0_IRQHandler(void)
{
    if (TIMER_GetTFFlag(TIMER0))
    {
        TIMER_ClearTFFlag(TIMER0, TIMER_CTL_TIMER_FLAG_Msk);
        LOG("TIMER0 interrupt.\n");
    }
}

int main(void)
{
    Clock_Init();
    Sys_Init();

    CLK_SetTmrClkSrc(CLK_APB1_TMR0SEL_APB1CLK);
    TIMER_SetWorkMode(TIMER0, TIMER_PERIOD_WORK_MODE);
    TIMER_SetCmpValue(TIMER0, 24000000);

    TIMER_EnableInt(TIMER0);
    NVIC_EnableIRQ(TMR0_IRQn);

    LOG("Start Timer0.\n");
    TIMER_Start(TIMER0);

    while (1);
}
```

4.3 周期配置与计算

- 定时周期近似为：`T  CompareValue / f_timer`（其中 `f_timer` 为 Timer 时钟频率）
- 例程按 `f_timer = 48 MHz` 估算：`T  24000000 / 48000000 = 500ms`

注意：CompareValue 的有效位宽由寄存器字段决定（最大为 0x1FFFFFF）。若设置值超出位宽会被截断，导致周期不符合预期。

4.4 主要 API 说明

- `CLK_SetTmrClkSrc(sel)`：选择 Timer 时钟源（例程选择 APB1CLK）。
- `TIMER_SetWorkMode(timer, mode)`：设置工作模式（Periodic / Continuous）。
- `TIMER_SetCmpValue(timer, value)`：设置比较值，达到比较值触发超时事件。
- `TIMER_EnableInt(timer)` / `NVIC_EnableIRQ(irqn)`：分别使能外设中断与 NVIC。
- `TIMER_GetTFFlag(timer)` / `TIMER_ClearTFFlag(timer, flag)`：查询/清除超时标志；中断里必须清除标志。
- `TIMER_Start(timer)`：启动计数。

4.5 常见修改

- 修改周期：调整 `TIMER_SetCmpValue()` 的 CompareValue（按 4.3 公式换算）。
- 修改时钟源：调整 `CLK_SetTmrClkSrc()` 的选择（降低 `f_timer` 可获得更长周期）。
- 关闭中断：调用 `TIMER_DisableInt()` 并 `NVIC_DisableIRQ()`，再按需 `TIMER_Stop()`。

3.1.6 UART

1 功能概述

本例程演示演示 UART Driver 的基本功能与使用方法。

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 连接串口转 USB 调试模块：
    - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
    - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来（用作 log 输出）
    - 使用杜邦线或跳线帽将串口模块的 RX/TX 分别与 EVB 底板 `J13` 排针 (P13 & TXD) 和 `J10` 排针对 (P14 & RXD) 分别短接起来（用作通信测试及结果输出）
  - 连接 Jlink，使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连

3 编译和烧录

例程位置：`<PAN271x-DK>\01_SDK\samples\drivers\uart`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

1. 烧录完成后，芯片会通过串口打印初始化 Log：

```
CPU @ 48000000Hz
+------------------------------------------------------------------------------+
|                      PAN271x UART Sample Code.                               |
+------------------------------------------------------------------------------+
|    Press key to start specific testcase:                                     |
|    Input '1'    Testcase 1: Baudrate Test.                                   |
|    Input '2'    Testcase 2: Interrupt Test.                                  |
+------------------------------------------------------------------------------+
```

2. 串口输入字符‘1’，uart 波特率测试，uart 波特率固定可设置为 1200、2400、4800、9600、19200、38400、57600、115200、921600 中一种，选定一种波特率后 uart0 发送 8Byte 固定数据，uart1 接收 uart0 发送的数据，同时 uart1 发送任意 8Byte 数据，uart0 正常接收到 uart1 的数据。

```
Send data:0x00 0x01 0x02 0x03 0xCC 0xDD 0xEE 0xFF
Data sent successfully.
Try to receive 8 bytes of data...
Data received: 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

3. 串口输入字符‘2’，uart 中断功能，通过串口输入选择 uart0 的 tx 和 rx fifo trig level，uart0 发送 125Byte 数据 0x00~0x7c，并等待 11Byte 接收数据，uart1 手动发送 11byte 数据，uart0 正常接收到 uart1 的数据。

```
FIFO trigger level setting done, prepare to transmit data (125 Bytes)...
TIMEOUT
Begin to receive data...
Data received (length=11):0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb
UART Test OK, Success case: 1
```

5 RAM/Otp 资源使用情况

- Otp Size: 9.16kB
- RAM Size: 1.05kB

3.1.7 WDT

1 功能概述

本例程演示 WatchDog Timer（WDT）Driver 的基本功能与使用方法。

例程主要完成以下内容：

- 初始化系统时钟、串口（UART0 115200）并打印日志
- 配置并启动 WDT：选择超时周期、复位延时、使能超时中断、使能超时复位
- 通过是否“喂狗”（`WDT_ResetCounter()`）演示两种典型现象：
  - 不喂狗：WDT 周期性超时，触发中断并在复位延时后复位系统
  - 喂狗：系统持续运行，不会因 WDT 超时复位

2 环境准备

- 硬件设备与线材：
  - PAN271x EVB **核心板**与**底板**各一块
  - JLink 仿真器（用于烧录例程程序）
  - USB-TypeC 线一条（用于底板供电和查看串口打印 Log）
  - 杜邦线数根或跳线帽数个（用于连接各个硬件设备）
- PC 软件：
  - 串口调试助手或终端工具，波特率 115200（用于查看串口 Log）
- 硬件接线：
  - 将 EVB 核心板插到底板上
  - 连接串口转 USB 调试模块：
    - 使用 USB-TypeC 线，将 PC USB 插口与 EVB 底板 USB->UART 插口相连
    - 使用杜邦线或跳线帽将 EVB 底板 `J8` 排针对 (P06 & TXD) 和 `J9` 排针对 (P05 & RXD) 分别短接起来
  - 连接 Jlink，使用杜邦线将 JLink 仿真器的：
    - SWD_CLK 引脚与 EVB 底板的 P00 排针相连
    - SWD_DAT 引脚与 EVB 底板的 P01 排针相连
    - SWD_GND 引脚与 EVB 底板的 GND 排针相连
  - P20 接逻辑分析仪

3 编译和烧录

例程位置：`<PAN271x-DK>\01_SDK\samples\drivers\wdt`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

本例程默认行为：启动 WDT 后**不喂狗**（主循环中 `WDT_ResetCounter()` 处于注释状态），因此会周期性触发 WDT 超时并复位。

1. 烧录完成后，芯片会通过串口打印一次初始化 Log（示例）：

```
CPU @ 48000000Hz
```

1. 随后在约“WDT 超时 + 复位延时”的总时间后发生复位，并再次打印 `CPU @ ...`，形成循环。

   总时间满足：`T_reset  T_timeout + T_delay`

   - 本例程配置为：
     - 超时周期：`WDT_TIMEOUT_2POW15`
     - 复位延时：`WDT_RESET_DELAY_1025CLK`
     - 使能超时复位：`TRUE`
     - 使能超时中断：`WDT_EnableInt()`
   - 典型现象（示例）：

```
CPU @ 48000000Hz
CPU @ 48000000Hz
CPU @ 48000000Hz
...
```

2. 如需验证“喂狗不复位”，将 `samples\drivers\wdt\src\main.c` 中主循环的 `WDT_ResetCounter();` 取消注释即可，现象变为：只打印一次 `CPU @ ...`，系统不再周期性复位。

4.1 主要流程代码（main.c 节选）

```
/* Unlock Regs before Enable WDT as several WDT Regs are write-protected */
SYS_UnlockReg();
WDT_Open(WDT_TIMEOUT_2POW15, WDT_RESET_DELAY_1025CLK, TRUE, FALSE);
WDT_EnableInt();
NVIC_EnableIRQ(WDT_IRQn);
SYS_LockReg(); /* Re-lock Regs */

while (1)
{
   /* Feed Watchdog */
   // WDT_ResetCounter();
}
```

4.2 超时与复位延时计算方法

- WDT 超时（近似）：`T_timeout ≈ (2^N) * WDT_CLK` 其中 N 由 `WDT_TIMEOUT_2POWxx` 的“xx”决定。

```
0000 = 2^4 * WDT_CLK.
0001 = 2^6 * WDT_CLK.
0010 = 2^8 * WDT_CLK.
0011 = 2^10 * WDT_CLK.
0100 = 2^12 * WDT_CLK.
0101 = 2^14 * WDT_CLK.
0110 = 2^15 * WDT_CLK.
0111 = 2^16 * WDT_CLK.
1000 = 2^17 * WDT_CLK.
1001 = 2^18 * WDT_CLK.
1010 = 2^19 * WDT_CLK.
1011 = 2^20 * WDT_CLK.
1100 = 2^21 * WDT_CLK.
1101 = 2^22 * WDT_CLK.
1110 = 2^23 * WDT_CLK.
1111 = 2^24 * WDT_CLK.
```

- 复位延时（近似）：`T_delay ≈ D * WDT_CLK` 其中 `WDT_RESET_DELAY_1025CLK/129CLK/17CLK/2CLK` 对应 `D ∈ {1025, 129, 17, 2}`。

4.3 主要 API 说明

- `SYS_UnlockReg()` / `SYS_LockReg()`：WDT 的部分寄存器为写保护寄存器，配置/使能前需解锁。
- `WDT_Open(timeout, resetDelay, enableReset, enableWakeup)`：配置 WDT time-out 和 delay 时间，超时是否 reset，是否启用低功耗唤醒功能。
- `WDT_EnableInt()` / `WDT_DisableInt()`：使能/关闭 WDT 超时中断。
- `NVIC_EnableIRQ(WDT_IRQn)`：在 NVIC 中打开 WDT 中断。
- `WDT_ResetCounter()`：喂狗（清 WDT 计数器）。注意该函数内部会临时解锁/上锁寄存器。

- `WDT_ClearTimeoutIntFlag()`：清除超时中断标志（在自定义 `WDT_IRQHandler()` 中常用）。

4.4 注意事项

- 本例程使能了 `WDT_IRQn`，但未在例程中实现 `WDT_IRQHandler()`。因为使能了 WDT 超时复位，硬件会在复位延时到达后复位系统，因此整体现象通常仍表现为“周期性复位”。
- 若你希望在超时中断时打印日志/做处理，请在工程中实现 `WDT_IRQHandler()`，并在中断中获取中断 `WDT_GetTimeoutIntFlag` 清标志 `WDT_ClearTimeoutIntFlag`（必要时喂狗），避免重复进入中断。
- 常见组合建议：
  - 仅“复位看门狗”：可不使能 `WDT_EnableInt()` / `NVIC_EnableIRQ(WDT_IRQn)`/不实现中断服务函数，只保留 `WDT_Open(..., enableReset=TRUE, ...)`。
  - 仅“中断提醒不复位”：将 `WDT_Open(..., enableReset=FALSE, ...)`，并实现 `WDT_IRQHandler()` 来清中断标志并执行你的超时处理。

3.2 私有 2.4G 例程

3.2.1 PRF 2.4G Normal Trx

1 功能概述

本例程演示 PRF 2.4G **普通（Normal）工作模式**下的 Tx/Rx 通信基本流程：

- Tx 端周期性发送固定长度 Payload
- Rx 端接收并打印 Payload，同时输出 RSSI

例程工程支持通过宏一键切换 Tx/Rx（需要两块板分别烧录 Tx 与 Rx）。

2 例程位置与工程文件

- 例程目录：`<PAN271x-DK>\01_SDK\samples\proprietary_rf\00_normal_trx`
- Keil 工程：`keil\normal_trx.uvprojx`
- 主要代码：`src\main.c`

3 环境准备

3.1 硬件设备 建议准备两套硬件做对测（Tx 一套 + Rx 一套）：

- PAN271x EVB **核心板**与**底板**各 2 套（至少 2 块核心板）
- JLink 仿真器（用于下载程序）
- USB-TypeC 线（用于供电与查看串口 Log）
- 杜邦线/跳线帽（用于连接 SWD 与串口相关跳线）

3.2 硬件接线 对每一套板：

1. 将 EVB 核心板插到底板上
2. 串口（UART0）连接：
   - 用 USB-TypeC 线，将 PC 与底板 USB->UART 口相连
   - 用跳线帽/杜邦线将底板排针：`J8` (P06 & TXD) 与 `J9` (P05 & RXD) 分别短接

  - 例程中 UART0 引脚复用为：P0.5=TX，P0.6=RX（见 `Sys_Init()`）

3. JLink（SWD）连接：
   - SWD_CLK → 底板 P00
   - SWD_DAT → 底板 P01
   - SWD_GND → 底板 GND

3.3 PC 软件

- 串口工具：UartAssist / SecureCRT / MobaXterm 等
- 串口参数：115200, 8N1

4 编译与烧录（Keil）

1. 打开工程 `keil\normal_trx.uvprojx`
2. 选择目标工程 `normal_trx`
3. 编译：Build
4. 下载：Download

说明：该工程在 Keil 的 Flash Download 配置里使用了外部烧录工具（JFlash）与 `keil\settings.jflash`，因此“下载”动作会调用 SDK 自带的 JFlash 执行擦除/烧录/启动。

5 使用方法（Tx/Rx 对测）

5.1 选择 Tx 或 Rx 在 `src\main.c` 顶部通过宏切换：

```
#define TX_MODE (1)   // 1=Tx, 0=Rx
```

- 在 Tx **板**：保持 `TX_MODE=1`，编译并烧录
- 在 Rx **板**：修改为 `TX_MODE=0`，编译并烧录

注意：Tx 与 Rx 的 RF 配置必须一致（频道、速率、地址、地址宽度、Payload 长度等）。

5.2 上电运行 两块板均上电后，分别打开各自串口。

- Tx 端会每隔约 300ms 发送一次 Payload
- Rx 端持续处于接收状态，收到包就打印内容与 RSSI

6 关键配置说明（RFConfig_t）

例程默认配置位于 `src\main.c` 的 `gRfConfig`：

- `WorkMode = PAN271_WORKMODE_NORMAL`：普通模式
- `Channel = 55`：工作信道（Tx/Rx 必须一致）
- `DataRate = PAN271_DR_1Mbps`：速率（Tx/Rx 必须一致）
- `TxPower = PAN271_TXPWR_11dBm`：发射功率
- `AddrWidth = PAN271_WIDTH_5BYTES`：地址宽度（Tx/Rx 必须一致）
- `TxAddr` / `RxAddr[pipe0]`：收发地址（TxAddr 与 Rx pipe0 地址需一致）
- `EnTxNoAck = ENABLE`：不需要 ACK（若要更高可靠性，请按协议需求关闭并配套设置）
- `TxLen` / `RxLen`：Payload 长度

- 例程使用 32 字节
- TxBuf/RxBuf **数组空间需要是** 4 **的整数倍**（例程已有注释）

Rx 超时相关：

- `RxTimeoutUs = 0`：例程默认不启用接收超时
- 若设置为非 0，可配合 `rx_timeout_cb` 在超时时打印/重启接收等

7 串口输出示例

7.1 Tx 端　典型输出：

```
CPU @ 48000000Hz
tx done, count: 1
tx done, count: 2
...
```

7.2 Rx 端　典型输出（内容会随实际 Payload 变化）：

```
CPU @ 48000000Hz
rx len:32
00 01 02 03 04 05 06 07 ...
rssi:-45 dBm
```

8 常见问题排查

1. 串口无打印：
   - 检查波特率是否 115200
   - 检查 UART0 复用引脚（P0.5/P0.6）与底板跳线是否按要求短接
   - 确认 USB->UART 口枚举到正确的 COM 口
2. Rx 收不到包：
   - 确认两块板分别烧录了 Tx 与 Rx（`TX_MODE` 不同）
   - 确认 `Channel/DataRate/AddrWidth/TxAddr/RxAddr/TxLen/RxLen` 完全一致
   - 减小两板距离、避免强干扰环境；必要时调整信道
3. Rx 输出 `crc/pid/len err`：
   - 多为配置不一致或干扰导致，优先检查速率/长度/地址/信道

3.2.2 PRF 2.4G Enhance Trx

1 功能概述

本例程演示 PRF 2.4G **增强（Enhance）工作模式**下的 Tx/Rx 通信基本流程，重点展示：

- 开启 `.WorkMode = PAN271_WORKMODE_ENHANCE`, ACK（`EnTxNoAck = DISABLE`）时的收发过程
- Rx 端在接收中断中封装并发送 ACK Payload
- 多 PIPE 地址配置（示例中启用 Pipe0/1/2）

例程工程支持通过宏一键切换 Tx/Rx（需要两块板分别烧录 Tx 与 Rx）。

2 例程位置与工程文件

- 例程目录：`<PAN271x-DK>\01_SDK\samples\proprietary_rf\01_enhance_trx`
  - 若你的 SDK 外层还有 `01_SDK` 目录，也可能是：`<PAN271x-DK>\01_SDK\samples\proprietary_rf\01_enhance_trx`
- Keil 工程：`keil\enhance_trx.uvprojx`
- 主要代码：`src\main.c`

3 环境准备

3.1 硬件设备

建议准备两套硬件做对测（Tx 一套 + Rx 一套）：

- PAN271x EVB **核心板**与**底板**各 2 套（至少 2 块核心板）
- JLink 仿真器（用于下载程序）
- USB-TypeC 线（用于供电与查看串口 Log）
- 杜邦线/跳线帽（用于连接 SWD 与串口相关跳线）

3.2 硬件接线

对每一套板：

1. 将 EVB 核心板插到底板上
2. 串口（UART0）连接：
   - 用 USB-TypeC 线，将 PC 与底板 USB->UART 口相连
   - 用跳线帽/杜邦线将底板排针：`J8` (P06 & TXD) 与 `J9` (P05 & RXD) 分别短接
     - 例程中 UART0 引脚复用为：P0.5=TX，P0.6=RX（见 `Sys_Init()`）
3. JLink（SWD）连接：
   - SWD_CLK → 底板 P00
   - SWD_DAT → 底板 P01
   - SWD_GND → 底板 GND

3.3 PC 软件

- 串口工具：UartAssist / SecureCRT / MobaXterm 等
- 串口参数：115200, 8N1

4 编译与烧录（Keil）

1. 打开工程 `keil\enhance_trx.uvprojx`
2. 选择目标工程 `enhance_trx`
3. 编译：Build
4. 下载：Download

说明：该工程在 Keil 的 Flash Download 配置里使用了外部烧录工具（JFlash）与 `keil\settings.jflash`，因此“下载”动作会调用 SDK 自带的 JFlash 执行擦除/烧录/启动。

5 使用方法（Tx/Rx 对测）

5.1 选择 Tx 或 Rx

在 src\main.c 顶部通过宏切换：

```
#define TX_MODE (1)   // 1=Tx, 0=Rx
```

- 在 Tx **板**：保持 TX_MODE=1，编译并烧录
- 在 Rx **板**：修改为 TX_MODE=0，编译并烧录

注意：Tx 与 Rx 的 RF 配置必须一致（频道、速率、地址宽度、Payload 长度、是否启用 ACK 等）。

5.2 上电运行

两块板均上电后，分别打开各自串口。

- Tx 端约每 300ms 发送一次 Payload
- Rx 端持续处于接收状态，收到包后打印 Payload + RSSI，并准备 ACK Payload

6 增强模式（ENHANCE）要点

本例程默认：

- WorkMode = PAN271_WORKMODE_ENHANCE
- EnTxNoAck = DISABLE（即 **需要** ACK）

启用 ACK 时的关键点：

- Tx 端需要配置 RxTimeoutUs 用于等待 ACK（例程 Tx 侧设为 300us）
- Rx 端需要在“接收完成中断”中准备 ACK payload，并调用 PAN271_RF_SetTxData()
  - 例程在 event_rx_fun() 中对 TxBuf 做自增并调用 PAN271_RF_SetTxData(TxBuf, gRfConfig.TxLen)

如果将 EnTxNoAck 改为 ENABLE（不需要 ACK），Rx 侧的 ACK 封装逻辑不会执行；此时现象更接近普通模式，但仍处于 ENHANCE workmode。

7 关键配置说明（RFConfig_t）

例程默认配置位于 src\main.c 的 gRfConfig：

- Channel = 68：工作信道（Tx/Rx 必须一致）
- DataRate = PAN271_DR_1Mbps：速率（Tx/Rx 必须一致）
- AddrWidth = PAN271_WIDTH_5BYTES：地址宽度（Tx/Rx 必须一致）
- TxAddr = {12 34 56 78 9A}：Tx 地址
- RxAddr[0..2]：Pipe0/1/2 地址（示例中均启用）
  - 对测最简单做法：让 TxAddr 与 RxAddr[0] 完全一致（例程默认就是这样）
- TxLen / RxLen：Payload 长度（例程为 32 字节）
  - TxBuf/RxBuf **数组空间需要是** 4 **的整数倍**（例程已有注释）
- RxTimeoutUs：等待接收/ACK 的超时（例程 Tx 侧 300us，Rx 侧 0）
- TrxTransTime = 50：收/发切换等待时间（例程注释说明默认固定为 50）

自动重发参数（例程默认关闭）：

- AutoDelayUs：自动重发间隔
- AutoMaxCnt：自动重发最大次数

8 串口输出示例

8.1 Tx 端　典型输出：

```
CPU @ 48000000Hz
tx done cnt=1
tx done cnt=2
...
```

若开启 ACK 且对端未工作/地址不匹配，可能出现超时打印：

```
rx timeout
```

8.2 Rx 端　典型输出（内容会随实际 Payload 变化）：

```
CPU @ 48000000Hz
rx len:32
00 01 02 03 04 05 06 07 ...
rssi:-45 dBm Cnt:1
```

3.2.3 PRF 2.4G 16M Crystal Trx

1 功能概述

本例程演示 PRF 2.4G 在 16MHz **外部晶振（**XTH=16MHz**）**条件下的 Tx/Rx 通信基本流程，重点展示：

- 16MHz 外晶振条件下的系统时钟初始化（`Clock_Init()`）
- 使用 16MHz 专用 PRF 库 `prf_lib_16m.lib`
- Tx 端周期性发送 Payload，Rx 端接收并打印 Payload 与 RSSI

注意：本例程仅适用于 16MHz **外部晶振**的芯片/模组；若硬件为 32MHz 外晶振，系统时钟会不正确，导致例程表现异常。

2 例程位置与工程文件

- 例程目录：`<PAN271x-DK>\01_SDK\samples\proprietary_rf\02_clock_16m_trx`
- Keil 工程：`keil\clock_16m_trx.uvprojx`
- 主要代码：`src\main.c`
- 依赖库：`proprietary_rf\prf_lib_16m.lib`（与普通晶振例程使用的 `prf_lib.lib` 不同）

3 环境准备

3.1 **硬件设备**　建议准备两套硬件做对测（Tx 一套 + Rx 一套），并确保两套硬件均为 16MHz **外晶振版本**：

- PAN271x EVB **核心板**与**底板**各 2 套（至少 2 块核心板）
- JLink 仿真器（用于下载程序）
- USB-TypeC 线（用于供电与查看串口 Log）
- 杜邦线/跳线帽（用于连接 SWD 与串口相关跳线）

3.2 硬件接线

对每一套板：

1. 将 EVB 核心板插到底板上
2. 串口（UART0）连接：
   - 用 USB-TypeC 线，将 PC 与底板 USB->UART 口相连
   - 用跳线帽/杜邦线将底板排针：`J8` (P06 & TXD) 与 `J9` (P05 & RXD) 分别短接
     - 例程中 UART0 引脚复用为：P0.5=TX，P0.6=RX（见 `Sys_Init()`）
3. JLink（SWD）连接：
   - SWD_CLK → 底板 P00
   - SWD_DAT → 底板 P01
   - SWD_GND → 底板 GND

3.3 PC 软件

- 串口工具：UartAssist / SecureCRT / MobaXterm 等
- 串口参数：115200, 8N1

4 编译与烧录（Keil）

1. 打开工程 `keil\clock_16m_trx.uvprojx`
2. 选择目标工程 `clock_16m_tx`
3. 编译：Build
4. 下载：Download

说明：该工程在 Keil 的 Flash Download 配置里使用了外部烧录工具（JFlash）与 `keil\settings.jflash`，因此“下载”动作会调用 SDK 自带的 JFlash 执行擦除/烧录/启动。

5 使用方法（Tx/Rx 对测）

5.1 选择 Tx 或 Rx

在 `src\main.c` 顶部通过宏切换：

```
#define TX_MODE (1)   // 1=Tx, 0=Rx
```

- 在 Tx **板**：保持 `TX_MODE=1`，编译并烧录
- 在 Rx **板**：修改为 `TX_MODE=0`，编译并烧录

注意：Keil 工程只有一个目标 `clock_16m_tx`，Rx 侧同样通过修改 `TX_MODE` 来编译烧录。

5.2 上电运行

两块板均上电后，分别打开各自串口：

- Tx 端会每隔约 300ms 发送一次 Payload
- Rx 端持续处于接收状态，收到包就打印内容与 RSSI

6 16MHz 专用配置说明

6.1 系统时钟初始化（Clock_Init）

例程在 `main()` 的 `Clock_Init()` 中包含 16MHz 外晶振相关的关键配置（摘取要点）：

- 使能 DPLL 16MHz 模式：`PMU->DPLL_CTRL |= DPLL_CTRL_CLK_DPLL_16M_MODE_AON_Msk`

串口打印会输出 `SystemCoreClock`，用于快速验证时钟是否正确：

- 正常情况下应看到类似：`CPU @ 48000000Hz`

6.2 PRF 库选择（prf_lib_16m.lib） 本例程 Keil 工程已默认链接 16MHz 专用库：`proprietary_rf\prf_lib_16m.lib`。

如果你将该例程移植到其他工程或手动改工程配置，请务必确认链接库仍为 `prf_lib_16m.lib`；否则在 16MHz 外晶振条件下，可能出现时钟/参数不匹配导致收发异常。

7 关键配置说明（RFConfig_t）

例程默认 RF 配置位于 `src\main.c` 的 `gRfConfig`，Tx/Rx 需要保持一致：

- `WorkMode = PAN271_WORKMODE_NORMAL`：普通模式
- `Channel = 55`：工作信道
- `DataRate = PAN271_DR_1Mbps`：速率
- `TxPower = PAN271_TXPWR_11dBm`：发射功率
- `AddrWidth = PAN271_WIDTH_5BYTES`：地址宽度
- `TxAddr` / `RxAddr[pipe0]`：收发地址（TxAddr 与 Rx pipe0 地址需一致）
- `EnTxNoAck = ENABLE`：例程默认不需要 ACK
- `TxLen` / `RxLen`：Payload 长度（例程使用 32 字节）
  - TxBuf/RxBuf **数组空间需要是** 4 **的整数倍**（例程已有注释）

8 串口输出示例

8.1 Tx 端 典型输出：

```
CPU @ 48000000Hz
tx done, count: 1
tx done, count: 2
...
```

8.2 Rx 端 典型输出（内容会随实际 Payload 变化）：

```
CPU @ 48000000Hz
rx len:32
00 01 02 03 04 05 06 07 ...
rssi:-45 dBm
```

9 注意事项与排查建议

1. **硬件晶振必须为** 16MHz **外晶振**：若为 32MHz 外晶振，`SystemCoreClock` 打印值通常会异常，进而导致 RF 收发异常。
2. Rx 收不到包时优先核对：两板 `Channel/DataRate/AddrWidth/TxAddr/RxAddr/TxLen/RxLen` 是否完全一致。
3. 串口无打印时检查：波特率 115200、UART0 复用引脚（P0.5/P0.6）与底板跳线是否按要求短接、COM 口是否正确。

3.2.4 PRF 2.4G LP RF

1 功能概述

本例程演示 PRF 2.4G **发送（Tx）**与 MCU Deepsleep **低功耗**结合的基本流程：

- 初始化 RF（PRF 库）并准备 32 字节 Payload
- 进入 Deepsleep
- 通过 SleepTimer(32k) **定时**或 GPIO **上升沿**唤醒
- 唤醒后发送一次 Payload，并在发送完成回调里打印计数
- 循环执行，实现“休眠-唤醒-发送”的低功耗间歇发射

2 例程位置与工程文件

- 例程目录：`<PAN271x-DK>\01_SDK\samples\proprietary_rf\03_lp_rf`
- Keil 工程：`keil\lp_rf.uvprojx`
- 主要代码：`src\main.c`

3 环境准备

3.1 硬件设备

- PAN271x EVB **核心板**与**底板** 1 套（仅演示“低功耗 + 发射”时 1 套即可）
- 若要做对测：再准备 1 套板做 Rx
- JLink 仿真器（用于下载程序）
- USB-TypeC 线（用于供电与查看串口 Log）
- 杜邦线/跳线帽（用于连接 SWD 与串口相关跳线）

3.2 硬件接线 对每一套板：

1. 将 EVB 核心板插到底板上
2. 串口（UART0）连接：
   - 用 USB-TypeC 线，将 PC 与底板 USB->UART 口相连
   - 用跳线帽/杜邦线将底板排针：`J8` (P06 & TXD) 与 `J9` (P05 & RXD) 分别短接
     - 例程中 UART0 引脚复用为：P0.5=TX，P0.6=RX（见 `Sys_Init()`）
3. JLink（SWD）连接：
   - SWD_CLK → 底板 P00
   - SWD_DAT → 底板 P01
   - SWD_GND → 底板 GND

可选：GPIO 唤醒口测试

- 例程默认把 P1.3 (已扩展到 `J13`) 配为上升沿唤醒（见 `LP_WakeUpPinConfig(P1, GPIO_Pin_3)`）。
- 由于函数开启下拉（pulldown），如需外部唤醒，请让 P1.3 从低电平产生一次上升沿（例如接按键到高电平/或外部信号源）。

3.3 PC 软件

- 串口工具：UartAssist / SecureCRT / MobaXterm 等
- 串口参数：115200, 8N1

4 编译与烧录（Keil）

1. 打开工程 `keil\lp_rf.uvprojx`
2. 选择目标工程 `lp_rf`
3. 编译：Build
4. 下载：Download

说明：该工程在 Keil 的 Flash Download 配置里使用了外部烧录工具（JFlash）与 `keil\settings.jflash`，因此“下载”动作会调用 SDK 自带的 JFlash 执行擦除/烧录/启动。

5 使用方法

5.1 选择 Tx 或 Rx

在 `src\main.c` 顶部通过宏切换：

```
#define TX_MODE (1)   // 1=Tx(默认，带 Deepsleep 演示)，0=Rx
```

- Tx（**默认**）：编译烧录后会周期性“睡眠 → 唤醒 → 发送一次”
- Rx：修改为 `TX_MODE=0` 后编译烧录，设备会持续接收并打印 Payload + RSSI

5.2 Tx（Deepsleep 间歇发送）流程说明

Tx 模式下主循环核心逻辑如下（摘取要点）：

```
LOG("Enter Deepsleep mode.\r\n");

LP_WakeUpPinConfig(P1, GPIO_Pin_3);     // GPIO 上升沿唤醒

LP_SetUsrSleepTimer(32000*1, 1);        // SleepTimer 比较值，单位 =32k 时钟周期
LP_EnableDeepsleepTimer1();             // 使能 SleepTimer1 中断 + 唤醒

GPIO_DisableDigitalPath(P0,GPIO_Pin_6); // 关 UART0_RX 数字通路，降低漏电（按实际 RX 引脚调整）
LP_EnterDeepSleep();                    // 进入 Deepsleep
GPIO_EnableDigitalPath(P0,GPIO_Pin_6);

LOG("Exit Deepsleep mode.\r\n");

TxBuf[0]++;
PAN271_RF_SetTxData(TxBuf, sizeof(TxBuf));
PAN271_RF_TrxStart();                   // 发送一次
```

要点说明：

- `LP_SetUsrSleepTimer(sleepTime, idx)` 的 `sleepTime` 单位是 32k **时钟周期数**。
  - 例如 1 秒约为 `32000*1`；3 秒可改为 `32000*3`（参见 `drivers/pan_lp.h` 的说明）。
- `LP_EnableDeepsleepTimer1()` 使能的是 SleepTimer1（对应该例程的 `SLPTMR1_IRQHandler`）。
- 进入低功耗前关闭 UART0_RX 的数字通路，常用于降低低功耗漏电；唤醒后再恢复。

5.3 Rx（对测/调试）使用说明

Rx 模式下会执行：

```
PAN271_RF_TrxStart();
while(1) { }
```

收到包后会在接收回调中打印：Payload 长度、十六进制内容、RSSI。

对测建议：

- Tx 端：使用本例程（`TX_MODE=1`）
- Rx 端：可使用本例程编译成 `TX_MODE=0`，或直接使用 normal_trx 的 Rx 端

注意：Tx 与 Rx 的 RF 配置必须一致（频道、速率、地址、地址宽度、Payload 长度等）。

6 关键配置说明

6.1 RFConfig_t（结构体参数）

例程默认 RF 配置位于 `src\main.c` 的 `gRfConfig`：

- `WorkMode = PAN271_WORKMODE_NORMAL`：普通模式
- `Channel = 55`：工作信道（Tx/Rx 必须一致）
- `DataRate = PAN271_DR_1Mbps`：速率（Tx/Rx 必须一致）
- `TxPower = PAN271_TXPWR_11dBm`：发射功率
- `AddrWidth = PAN271_WIDTH_5BYTES`：地址宽度（Tx/Rx 必须一致）
- `TxAddr = {12 34 56 78 9A}`：Tx 地址
- `RxAddr[0]`：Pipe0 地址（示例启用且与 TxAddr 一致）
- `EnTxNoAck = ENABLE`：不需要 ACK
- `TxLen/RxLen = 32`：Payload 长度
  - TxBuf/RxBuf **数组空间需要是 4 的整数倍**（例程已有注释）

6.2 RF 中断回调（发送完成打印）

例程在 `src\main.c` 实现了 `PAN271_RF_ISR_Init()`，用于注册 RF 中断回调（如 `tx_cb`）。

- 发送完成后会进入 `event_tx_fun()` 并打印 `tx done, count: N`
- 该函数由 RF 驱动在 `PAN271_RF_Init()` 内部调用（驱动里提供了弱定义，用户实现会覆盖）

6.3 低功耗相关配置

进入 Deepsleep 前，例程还做了两类常见的“降漏电”处理：

1. 关闭下载口相关引脚数字通路/上拉（避免 SWD 引脚漏电）：

```
GPIO_DisableDigitalPath(P0, 0x03);
GPIO_DisablePullupPath(P0, 0x03);
```

1. 进入低功耗前关闭 UART0_RX 数字通路（P0.6），唤醒后恢复。

7 串口输出示例

7.1 Tx（Deepsleep + 发送）

典型输出：

```
CPU @ 48000000Hz
Enter Deepsleep mode.
slptmr1 int
Exit Deepsleep mode.
tx done, count: 1
Enter Deepsleep mode.
...
```

当 GPIO 唤醒触发时，可能看到：

```
P1 interrupt
```

7.2 Rx 典型输出（内容会随实际 Payload 变化）：

```
CPU @ 48000000Hz
rx len:32
00 01 02 03 04 05 06 07 ...
rssi:-45 dBm
```

8 常见问题排查

1. 串口无打印：
   - 检查波特率是否 115200
   - 检查 UART0 复用引脚（P0.5/P0.6）与底板跳线是否按要求短接
   - 确认 USB->UART 口枚举到正确的 COM 口
2. Rx 收不到包：
   - 确认两块板分别烧录了 Tx 与 Rx（`TX_MODE` 不同）
   - 确认 `Channel/DataRate/AddrWidth/TxAddr/RxAddr/TxLen/RxLen` 完全一致
   - 缩短两板距离、避开强干扰；必要时调整信道
3. 低功耗电流不理想：
   - 确认进入 Deepsleep 前已按需关闭 UART RX 数字通路、下载口相关引脚数字通路/上拉
   - 断开不必要的外设连接（例如某些调试接口），并结合板级电路评估漏电来源

外设驱动例程

源码路径: `<PAN271x-DK>\01_SDK\samples\drivers`

例程	说明
ADC	演示 ADC Driver 的基本功能与使用方法
GPIO	演示 GPIO Driver 的基本功能与使用方法
PWM	演示 PWM Driver 的基本功能与使用方法
TIMER	演示 Timer Driver 的基本功能与使用方法
UART	演示 UART Driver 的基本功能与使用方法
WDT	演示 WatchDog Driver 的基本功能与使用方法
LowPower	演示 deepsleep 的基本功能与使用方法

私有 2.4G 例程

源码路径: `<PAN271x-DK>\01_SDK\samples\proprietary_rf`

例程	说明
PRF 2.4G Normal Trx	演示 PRF 2.4G 普通型通信的基本流程
PRF 2.4G Enhance Trx	演示 PRF 2.4G 增强型通信的基本流程
PRF 2.4G 16M Crystal	演示 PRF 2.4G 16M 晶振通信的基本流程
PRF 2.4G lp_rf	演示 PRF 2.4G 通信和低功耗模式切换的基本流程

Chapter 4

开发指南

4.1 PRF 2.4G 开发指南

RF 2.4G 驱动可直接复用 SDK 内提供的例程，或通过代码导出工具生成对应的 RF 驱动库。推荐优先采用代码导出工具生成专用 RF 库，该方式生成的固件体积更小，能够有效节省 OTP 资源，为客户预留更多空间用于实现自身业务逻辑代码。

4.1.1 1 代码导出工具使用

代码导出工具为 PC **端配置工具**，用于把常用 PRF（2.4G）参数以 ** “LIB 或者工程” ** 的形式导出，减少手工填充 `RFConfig_t` 时的遗漏与不一致。

1.1 适用场景

- 需要快速生成 `RFConfig_t` 初始化代码（普通/增强模式、TX/RX、地址宽度、多 Pipe、动态长度等）。
- 需要在多个工程/多个设备间复用同一套 RF 参数，避免手工拷贝出错。
- 需要把 “工具界面上的参数” 与 “驱动初始化代码” 形成一一对应，便于评审与版本管理。
- 需要更多的 OTP 资源，容纳更多的业务代码。

1.2 基本使用流程

1. 打开代码导出工具。
2. 选择业务模式/协议模式（例如 XN297 、普通/增强等），并选择收发模式（TX 或 RX）。
3. 按需求配置关键参数（建议按下列顺序）：
   - **射频基础参数**：`Channel`、`DataRate`、`TxPower`、`WorkMode`。
   - **地址相关**：`AddrWidth`、`TxAddr`、接收端 `RxAddr[0..2]`（多 Pipe 场景）。
   - Payload **长度**：固定长度时配置 `TxLen`/`RxLen`；动态长度场景需同时确认工具导出的动态长度使能项与最大长度限制。
   - **增强型模式**：`EnTxNoAck`、自动重传（`AutoDelayUs`/`AutoMaxCnt`）、接收超时（`RxTimeoutUs`）、收发切换等待（`TrxTransTime`）。
4. 点击 “Export Lib File”，选择导出到文件或复制到剪贴板。
5. 将导出的代码粘贴到工程中（通常为 `RFConfig_t` 的初始化 + 相关驱动调用），并与例程的逻辑保持一致。

1.3 导出代码如何落地到例程

导出内容通常会包含以下两类之一（以工具实际输出为准）：

- 直接导出 RF LIB 库
- 导出 RF 工程方式

导出 RF LIB 库：

1. 以增强型为例，在例程（如 `samples/proprietary_rf/01_enhance_trx`）中替换 RF LIB 库，删除 `RFConfig_t` 结构体，并且 `PAN271_RF_Init` 的参数删除。
2. 确保在代码导出工具中 TX 与 RX 两端以下配置一致，否则可能表现为 “收不到包/偶发丢包/误包”：
   - `Channel`、`DataRate`、`WorkMode`（普通/增强）
   - `AddrWidth` 与对应的地址内容（TXAddr 与 RX pipe 地址）
   - Payload 长度策略（固定长度还是动态长度）
3. 若接收端使用多 Pipe：
   - 发送端仍只使用单一地址；
   - 接收端按需求开启 1~3 个 Pipe，并确保 `EnPipe` 使能位与地址匹配。

4.1.2 2 例程使用

1. TRX 例程功能：
   - 发射端例程的功能：每隔 300ms 发送一次 2.4G 数据包，长度 32 个字节。
   - 接收端例程的功能：接收发送端的 2.4G 信号，并将接收到的数据通过串口打印出来。
2. 开发说明：
   - 2.4g 初始化配置
   - API 介绍
   - SAMPLE 运行流程
   - 帧结构介绍

4.1.3 3 环境配置

1. 环境要求
   - board：pan271x_evb
   - uart：显示串口输出 log，uart 端口 P06（UART_RX）、P05（UART_TX），波特率 115200
2. 编译和烧录：
   1. 项目位置：
   - TX 端：” samples\proprietary_rf\01_enhance_trx” TX_MODE 置为 1。
   - RX 端：” samples\proprietary_rf\01_enhance_trx” TX_MODE 置为 0。

   工程默认配置是 XN297 模式，增强型。
3. 选择好后编译程序，用 j-link 烧录编译后的 hex 文件到 pan271x_evb 板子中。

4.1.4 4 演示说明

1. 步骤
   - 将接收端串口和发射端串口分别接到 PC 的 USB 端口上。
   - 配置接收端和发送端。
   - 观察 PC 串口工具的输出结果。
2. 结果

   发射端输出结果：

```
CPU @ 48000000Hz
rx len:32
63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-28 dBm Cnt:1
rx len:32
64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-28 dBm Cnt:2
rx len:32
65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-28 dBm Cnt:3
rx len:32
66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-28 dBm Cnt:4
rx len:32
67 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-27 dBm Cnt:5
rx len:32
68 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-24 dBm Cnt:6
rx len:32
69 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-23 dBm Cnt:7
rx len:32
6A 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-25 dBm Cnt:8
rx len:32
6B 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rssi:-25 dBm Cnt:9
```

   接收端输出结果：

```
CPU @ 48000000Hz
rx len:32
1 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-30 dBm Cnt:1
tx done cnt=1
rx len:32
2 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-31 dBm Cnt:2
tx done cnt=2
rx len:32
3 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-30 dBm Cnt:3
tx done cnt=3
rx len:32
4 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-28 dBm Cnt:4
tx done cnt=4
rx len:32
5 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-26 dBm Cnt:5
tx done cnt=5
rx len:32
6 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-27 dBm Cnt:6
tx done cnt=6
rx len:32
7 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
rssi:-30 dBm Cnt:7
```

4.1.5 5 开发说明

5.1 2.4G 初始化配置

配置结构体“RFConfig_t”（定义在 pan271_rf_def.h），各成员介绍如下：

Type	name	Description
PAN271_Trx_Mode_t	trx_mode	设置发送/接收模式：PAN271_TX_MODE / PAN271_RX_MODE
uint8_t	Channel	RF 频点配置（驱动接口注释为 0~127）
PAN271_TXPWR_t	TxPower	发射功率配置：-10/0/3/7/11 dBm
PAN271_DataRate_t	DataRate	数据速率配置：1Mbps / 250Kbps
PAN271_WorkMode_t	WorkMode	工作模式：普通/增强
PAN271_AddrWidth_t	AddrWidth	地址宽度设置：2~5 bytes
uint8_t	TxAddr[5]	发送地址（按 AddrWidth 指定的长度有效）
RxPipeAddr_t	RxAddr[3]	接收多管道地址（最多 3 路）；每路包含 EnPipe 使能位和 Addr[5] 地址
uint8_t	TxLen	发送数据长度（字节）
uint8_t	RxLen	接收数据长度（字节）
bool	EnTxNoAck	是否使能发送无应答（增强模式下常用）
uint16_t	RxTimeoutUs	接收超时时间，单位 us
uint16_t	AutoDelayUs	自动重传延迟时间，单位 us
uint8_t	AutoMaxCnt	自动重传最大次数
uint8_t	TrxTransTime	收/发切换等待时间，单位 us（例程中常用默认值为 50）

PAN271_Trx_Mode_t：

Type	Value	Description
PAN271_TX_MODE	0	发射模式
PAN271_RX_MODE	1	接收模式

PAN271_DataRate_t：

Type	Value	Description
PAN271_DR_250Kbps	1	250kbps 数据速率
PAN271_DR_1Mbps	3	1Mbps 数据速率

PAN271_TXPWR_t：

Type	Value	Description
PAN271_TXPWR_NEGATIVE_10dBm	0	-10 dBm
PAN271_TXPWR_0dBm	1	0 dBm
PAN271_TXPWR_3dBm	2	3 dBm
PAN271_TXPWR_7dBm	3	7 dBm
PAN271_TXPWR_11dBm	4	11 dBm

PAN271_WorkMode_t：

Type	Value	Description
PAN271_WORKMODE_NORMAL	0	普通模式
PAN271_WORKMODE_ENHANCE	1	增强模式

PAN271_AddrWidth_t：

Type	Value	Description
PAN271_WIDTH_2BYTES	2	2 bytes 地址宽度
PAN271_WIDTH_3BYTES	3	3 bytes 地址宽度
PAN271_WIDTH_4BYTES	4	4 bytes 地址宽度
PAN271_WIDTH_5BYTES	5	5 bytes 地址宽度

PAN271_RxPipeIndex_t：

Type	Value	Description
PAN271_RX_PIPE0	0	接收管道 0
PAN271_RX_PIPE1	1	接收管道 1
PAN271_RX_PIPE2	2	接收管道 2

5.2 API 介绍

本节接口以 `pan271_rf_driver.h` 提供的驱动 API 为准。以下按“每个接口单独描述”的方式列出。

PAN271_RF_Init

```
void PAN271_RF_Init(RFConfig_t* config);
```

初始化 PAN271 芯片（参数为 RF 配置结构体 `RFConfig_t` 指针）。

PAN271_SYS_LDO_Enable

```
void PAN271_SYS_LDO_Enable(void);
```

使能 RF LDO。

PAN271_SYS_LDO_Disable

```
void PAN271_SYS_LDO_Disable(void);
```

禁用 RF LDO。

PAN271_RF_ISR_Init

```
void PAN271_RF_ISR_Init(void);
```

初始化 PAN271 射频模块中断回调函数（用户可在工程中重写/实现）。

PAN271_RF_RFModuleReset

```
void PAN271_RF_RFModuleReset(void);
```

重启 RF 射频模块。

PAN271_RF_SetDataRate

```
void PAN271_RF_SetDataRate(PAN271_DataRate_t DataRate);
```

设置 PAN271 芯片数据速率。

PAN271_RF_EnableWhiten

```
void PAN271_RF_EnableWhiten(bool EnWhite);
```

设置 PAN271 是否使能白化。

PAN271_RF_SetWorkMode

```
void PAN271_RF_SetWorkMode(PAN271_WorkMode_t WorkMode);
```

设置 PAN271 工作模式（普通/增强）。

PAN271_RF_SetAddrWidth

```
void PAN271_RF_SetAddrWidth(uint8_t AddrWidth);
```

设置 PAN271 地址宽度（2~5 bytes）。

PAN271_RF_SetTrxMode

```
void PAN271_RF_SetTrxMode(PAN271_Trx_Mode_t TrxMode);
```

设置收发器工作模式（TX/RX）。

PAN271_RF_SetTrxRamReady

```
void PAN271_RF_SetTrxRamReady(PAN271_Trx_Mode_t TrxMode, bool Ready);
```

设置收发器 RAM 准备状态。

PAN271_RF_SetWhiteInitVal

```
void PAN271_RF_SetWhiteInitVal(uint8_t initVal);
```

设置白化初始值。

PAN271_EnableManualPid

```
void PAN271_EnableManualPid(bool EnManuPid);
```

设置手动 PID 使能。

PAN271_RF_SetTxPower

```
void PAN271_RF_SetTxPower(PAN271_TXPWR_t tx_pwr);
```

设置 PAN271 发射功率。

PAN271_RF_SetTxAddr

```
void PAN271_RF_SetTxAddr(uint8_t *Addr);
```

设置收发器的静态发送地址。

PAN271_RF_EnableTxNoAck

```
void PAN271_RF_EnableTxNoAck(const PAN271_WorkMode_t WorkMode, bool EnNoAck);
```

设置 PAN271 Tx No Ack 功能（与工作模式相关）。

PAN271_RF_SetAutoRetrans

```
void PAN271_RF_SetAutoRetrans(uint16_t AutoDelayUs, uint8_t AutoMaxCnt);
```

设置自动重传参数（延迟与最大次数）。

PAN271_RF_SetChannel

```
void PAN271_RF_SetChannel(uint8_t Channel);
```

设置工作频点（驱动注释：0~127 对应 2400MHz~2527MHz）。

PAN271_RF_SetTxData

```
void PAN271_RF_SetTxData(uint8_t *TxPayload, uint8_t PayloadLen);
```

设置发送有效数据及长度。

PAN271_RF_TrxStart

```
void PAN271_RF_TrxStart(void);
```

启动 PAN271 发送/接收。

PAN271_RF_StartCarrierWave

```
void PAN271_RF_StartCarrierWave(uint8_t Channel);
```

启动载波发射。

PAN271_RF_StopCarrierWave

```
void PAN271_RF_StopCarrierWave(void);
```

停止载波发射。

PAN271_RF_SetCrcScheme

```
void PAN271_RF_SetCrcScheme(void);
```

设置 CRC 校验模式。

PAN271_RF_RX_GOON_Enable

```
void PAN271_RF_RX_GOON_Enable(bool EnRxGoon);
```

当 CRC 错误时，设置是否继续接收。

PAN271_RF_SetRxPayloadLen

```
void PAN271_RF_SetRxPayloadLen(uint8_t PayloadLen);
```

设置接收有效数据长度。

PAN271_RF_GetRxPayload

```
uint8_t PAN271_RF_GetRxPayload(uint8_t* RxPayload);
```

获取接收数据，返回接收数据长度。

PAN271_RF_EnableDynamicPL

```
void PAN271_RF_EnableDynamicPL(bool Dpy_En);
```

设置是否使能动态包长度解析。

PAN271_RF_SetRxSingleAddr

```
void PAN271_RF_SetRxSingleAddr(RxPipeAddr_t* RxAddr, PAN271_RxPipeIndex_t PipeIndex);
```

设置某个管道接收地址。

PAN271_RF_SetRxAddr

```
void PAN271_RF_SetRxAddr(RxPipeAddr_t* RxAddr, uint8_t NumPipes);
```

设置多管道接收地址（NumPipes 最大为 RF_RX_MAX_PIPE_NUM）。

PAN271_RF_GetPipesNum

```
uint8_t PAN271_RF_GetPipesNum(void);
```

获取已启用的接收管道数量。

PAN271_RF_SetWaitAckTimeout

```
void PAN271_RF_SetWaitAckTimeout(uint16_t AckTimeoutUs);
```

设置等待 ACK 超时时间（单位 us）。

PAN271_RF_GetRxPacketRssi

```
int16_t PAN271_RF_GetRxPacketRssi(void);
```

获取接收数据包 RSSI（缓存值）。

PAN271_RF_GetRealTimeRssi

```
int16_t PAN271_RF_GetRealTimeRssi(void);
```

读取实时 RSSI。

PAN271_RF_ExitContinueRxMode

```
void PAN271_RF_ExitContinueRxMode(void);
```

退出连续接收模式。

PAN271_RF_SetMaxRxLength

```
void PAN271_RF_SetMaxRxLength(uint8_t MaxLen);
```

设置最大接收数据长度。

PAN271_RF_VCO_Cal

```
void PAN271_RF_VCO_Cal(void);
```

VCO 校准。

PAN271_RF_TwoPointCal

```
void PAN271_RF_TwoPointCal(PAN271_DataRate_t DataRate);
```

两点式校准。

PAN271_RF_BW_DCOC_Cal

```
void PAN271_RF_BW_DCOC_Cal(PAN271_DataRate_t DataRate);
```

带宽/直流偏移校准。

5.3 MULTI PIPE 使用介绍

1. 多 pipe 的使用主要在于接收端，发送端同时只有一个 pipe，接收端同时有 3 个地址。
2. 驱动侧多 pipe 地址可通过初始化配置结构体 `RFConfig_t` 中的 `RxAddr[3]` 一次性配置（每路包含 `EnPipe` 使能位与 `Addr[5]` 地址），随后调用 `PAN271_RF_Init()` 生效。
3. 若需要运行过程中动态修改，可使用 `PAN271_RF_SetRxSingleAddr()` 或 `PAN271_RF_SetRxAddr()` 更新接收端多管道地址。

5.4 中断介绍

本 SDK 的 PRF（2.4G）中断采用 "**软件分发回调**" 的形式：

- 硬件中断源为 `LL_IRQn`（向量函数名为 `LL_IRQHandler`）。
- `LL_IRQHandler()` 内部调用 `PAN271_RF_IRQ_Handler()`，由后者根据 R11_CFG **的中断** flag 分发到用户注册的回调。
- `PAN271_RF_Init()` 会在初始化过程中自动完成：
  - 使能 `LL_IRQn`（内部调用 `PAN271_RF_IRQ_Enable()`）
  - 调用 `PAN271_RF_ISR_Init()` 初始化/注册回调（该函数在驱动中为弱定义，用户工程实现同名函数即可覆盖）

5.4.1 中断入口（LL_IRQHandler）与使能（LL_IRQn）

中断入口在 [proprietary_rf/src/pan271_rf_port.c] 中实现：

```
void PAN271_RF_IRQ_Enable(void)
{
   NVIC_SetPriority(LL_IRQn, 0);
   NVIC_EnableIRQ(LL_IRQn);
}

void LL_IRQHandler(void)
{
   PAN271_RF_IRQ_Handler();
}
```

一般情况下不需要用户手动 `NVIC_EnableIRQ()`；只要调用 `PAN271_RF_Init()`，驱动会自动使能 RF 中断。

5.4.2 回调注册（以 enhance_trx 为例）

回调结构体类型为 `PAN271_Callback_t`（定义在 `pan271_rf_def.h`），全局实例 `isr_cb` 位于 [proprietary_rf/src/pan271_rf_port.c]。

enhance_trx 例程在 [samples/proprietary_rf/01_enhance_trx/src/main.c] 中实现 `PAN271_RF_ISR_Init()`，用于注册回调：

```
void PAN271_RF_ISR_Init(void)
{
   isr_cb.tx_cb = event_tx_fun;
   isr_cb.rx_cb = event_rx_fun;
   isr_cb.rx_timeout_cb = event_rx_timeout_fun;
   isr_cb.rx_crc_err_cb = event_crc_err_fun;
```

（下页继续）

(续上页)

```
    isr_cb.rx_pid_err_cb = event_pid_err_fun;
    isr_cb.rx_acc_adr_err_cb = event_acc_adr_err_fun;
    isr_cb.rx_len_err_cb = event_len_err_fun;
}
```

说明:

- `PAN271_RF_ISR_Init()` 在驱动中提供弱定义，应用侧实现同名函数即可“覆盖”驱动默认实现。
- 该函数由 `PAN271_RF_Init()` 内部调用，典型工程无需额外调用。

5.4.3 **中断 flag 与回调分发关系** `PAN271_RF_IRQ_Handler()` 会依次检查 R11_CFG 中的各中断 flag (`PAN271_RF_IntFlag()` 本质是判断 `PRI_RF->R11_CFG & mask`)，若回调非空则调用，最后清除中断:

中断事件	flag（寄存器位）	回调指针 (PAN271_Callback_t)	典型含义
接收完成	R11_CFG: RX_IRQ_FLAG	`rx_cb`	Rx FIFO 有效数据可读
接收超时	R11_CFG: TIMEOUT_IRQ_FLAG	`rx_timeout_cb`	等待接收超时（与 `RxTimeoutUs`/掩码配置相关）
CRC 错误	R11_CFG: CRC_ERR_IRQ_FLAG	`rx_crc_err_cb`	收到包但 CRC 校验失败
PID 错误	R11_CFG: PID_ERR_IRQ_FLAG	`rx_pid_err_cb`	PID 校验失败（增强/重传相关）
长度错误	R11_CFG: LENGTH_ERR_IRQ_FLAG	`rx_len_err_cb`	length 字段非法或不匹配
地址错误	R11_CFG: ACC_ADDR_ERR_IRQ_FLAG	`rx_acc_adr_err_cb`	Access Address 不匹配
发送完成	R11_CFG: TX_DONE_IRQ_FLAG	`tx_cb`	Tx 流程结束（与工作模式/掩码配置相关）

5.4.4 **增强型 ACK Payload 的关键点 (enhance_trx)** 在增强模式且需要 ACK 的接收端场景中 (`EnTxNoAck == DISABLE` 且设备处于 `PAN271_RX_MODE`)，接收端需要在“接收完成回调”中快速准备 ACK payload。enhance_trx 的示例是在 `event_rx_fun()` 中直接调用:

```
if ((gRfConfig.EnTxNoAck == DISABLE) && (gRfConfig.trx_mode == PAN271_RX_MODE))
{
    TxBuf[0]++;
    PAN271_RF_SetTxData(TxBuf, gRfConfig.TxLen);
}
```

该段代码运行在中断上下文中，目的是赶在硬件发送 ACK 之前修改 BUFFER，并把 BUFFER 复制到 FIFO 中。

注意：在高性能的 2.4g 应用中尽量不要使用打印，会影响射频收发性能。增强型 RX 中断中不能有延时或者打印或者执行时间很长的接口，不然可能会影响发送中断从而导致时序错误。

4.1.6 6 Sample 运行流程

enhance_rx 例程运行流程

如下图所示:

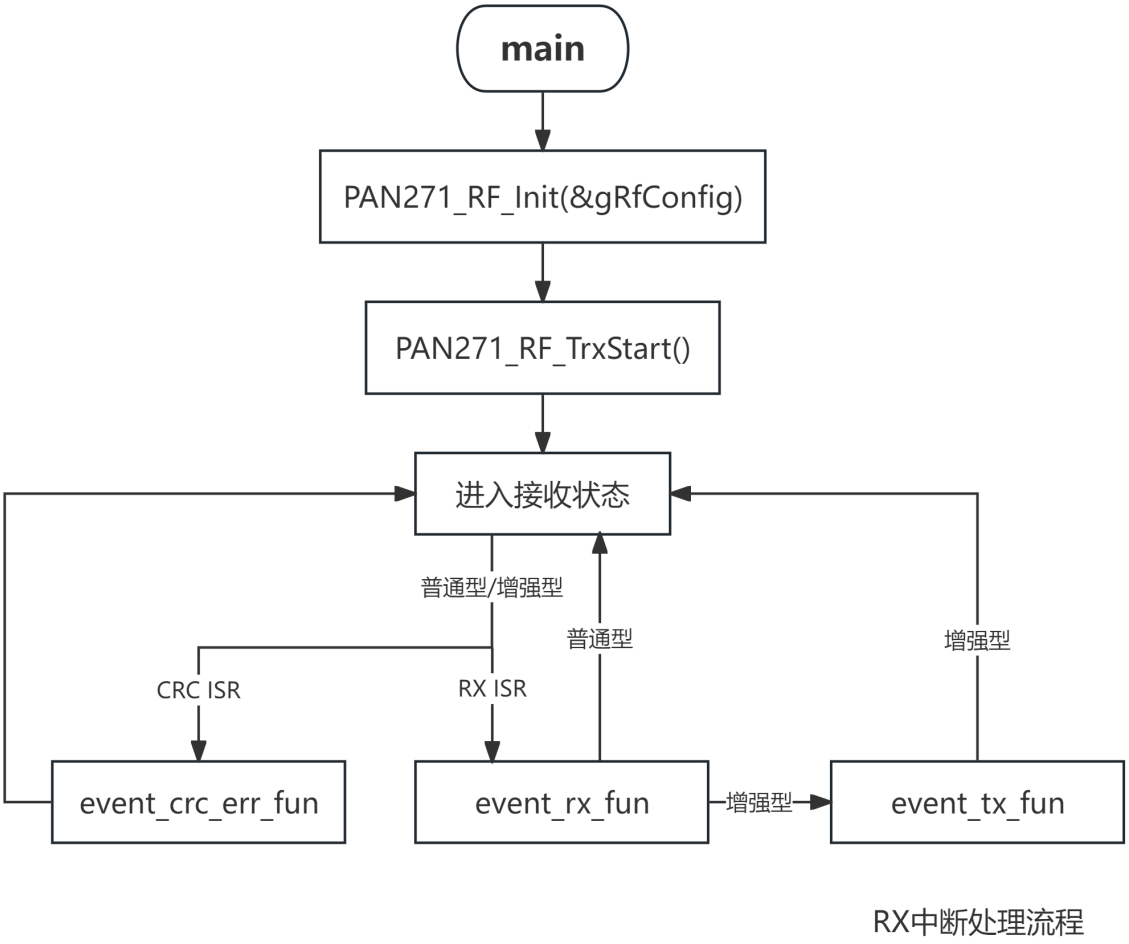
main
PAN271_RF_Init(&gRfConfig)
PAN271_RF_TrxStart()
进入接收状态
普通型/增强型
普通型
增强型
CRC ISR
RX ISR
event_crc_err_fun
event_rx_fun
增强型
event_tx_fun
RX中断处理流程

图 1: RX 例程流程图

enhance_tx 例程运行流程

如下图所示：

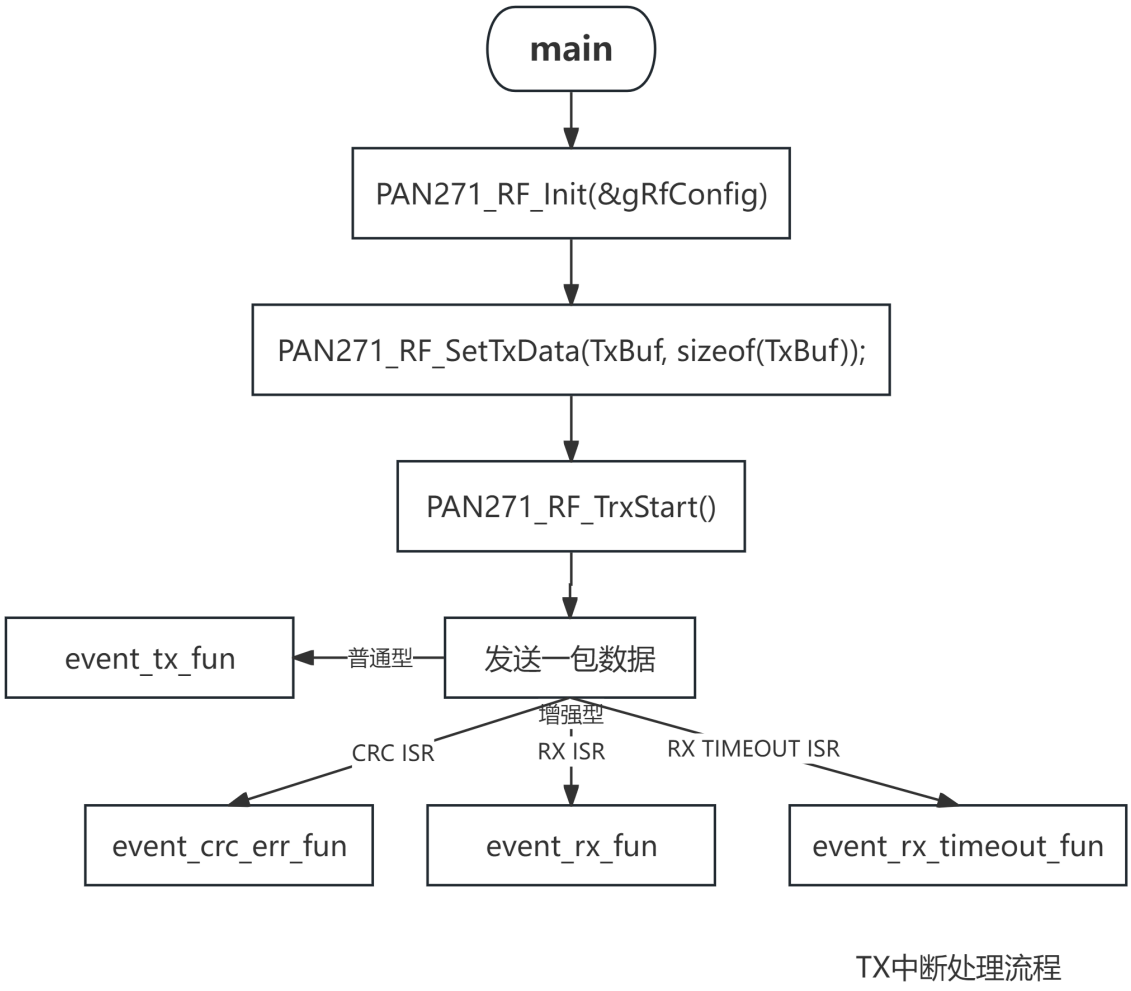

图 2: TX 例程流程图

4.1.7 7 2.4G 帧结构介绍

7.1 xn297 兼容帧结构

普通型：

3 byte	2~5 byte	0~255 byte	2 byte
preamble(0x710f55)	addr	payload	crc(包含 addr 和 payload)

空中 bit 序：大端模式

PAN271 普通型的 payload 长度范围为：0~255 Byte

增强型：在地址和 payload 之间插入 signal(10bit) 数据。

3 byte	2~5 byte	10bit	0~127 byte	2 byte
preamble(0x710f55)	addr	signal	payload	crc(包含 addr 和 payload)

signal 结构：

7bit	2bit	1bit
数据长度标识 (动态 payload)	PID 标识 (判断是否是重发包)	NO_ACK 标识 (tx 完是否需要 ack)

PAN271 普通型的 payload 长度范围为：0~127 Byte

白化和 crc 的作用域包括 addr+payload+signal。

4.1.8 7 2.4G PID 流程

tx 和 rx pid 处理逻辑如下：

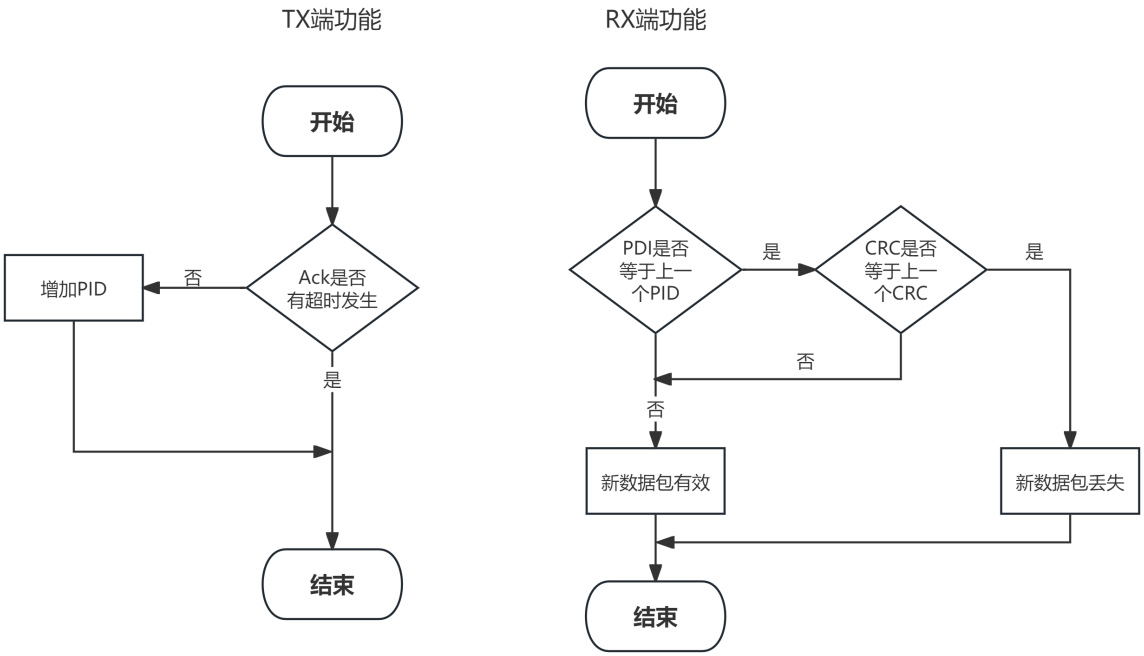

图 3: PID 生成和检测

每一包数据都包括两位的 PID（数据包标志位），来帮助接收端识别该数据是新数据包还是重发的数据包，防止多次存入相同的数据包，PID 的生成和检测如上图所示。发送端的 ack 包未发生超时 PID 值加一。支持软件配置 tx pid 和 rx pid。

4.2 常见问题（FAQs）

4.2.1 Q1：PAN271x 芯片烧录方式是什么？

我们推荐使用 Panchip 提供的量产烧录工具 PANLink 烧录 OTP 版本的芯片（详见量产烧录 文档说明）；

而 JLink 用于调试烧录预生产版本（SRAM 模拟 OTP）的芯片。

4.2.2 Q2：PAN271x 收发两端为什么不能正常通信？

RF 收发通信需要多个参数匹配才能正常通信，具体包括以下几点：

1. 工作模式：保证收发两端处于相同的工作模式
2. 通信速率：保证收发两端处于相同的通信速率
3. 通信地址：保证收发两端通信地址长度和内容都相同
4. 数据长度：普通模式时（固定包长）保证收发两端通信数据长度要相同
5. CRC 校验：CRC 校验的开启/关闭两端要匹配
6. 距离较远时，选择高功率，低速率方式通信

4.2.3 Q3：16M 晶振与 32M 晶振使用的区别是什么？

1. 16M、32M 在硬件上使用对应的 16M/32M 晶振；
2. 使用 16M 晶振时，修改 Clock_Init 函数，将 DPLL 设置为 16M 晶振模式；
3. 使用 16M 晶振时，导入 `prf_lib_16m.lib` 的 RF 库；

具体可以查看 `samples/proprietary_rf/02_clock_16m_trx` 例程；

4.2.4 Q4：普通模式（NORMAL）与增强模式（ENHANCE）有什么区别？

- NORMAL：常用于固定包长的简单通信。
- ENHANCE：常见能力包括 ACK、自动重传、多 PIPE 等。

增强模式常用参数：

- `EnTxNoAck`：是否需要 ACK（`DISABLE` 表示需要 ACK，`ENABLE` 表示不需要 ACK）
- `RxTimeoutUs`：等待接收/ACK 的超时（单位 us）
- `AutoDelayUs` / `AutoMaxCnt`：自动重传间隔与次数

4.2.5 Q5：增强模式下 Tx 端一直打印 `rx timeout`，怎么排查？

`rx timeout` 通常表示 Tx 侧等待 ACK（或等待接收）超时。建议按以下顺序排查：

1. **两块板是否分别烧录了** Tx **与** Rx（`TX_MODE` 是否一 Tx 一 Rx）。
2. Tx/Rx **配置是否完全一致**：`Channel/DataRate/WorkMode/AddrWidth/TxAddr/RxAddr/TxLen/RxLen`。
3. 若确实要使用 ACK：
   - Tx 和 Rx 侧 `EnTxNoAck = DISABLE`（需要 ACK），并设置合理的 `RxTimeoutUs`。
   - Rx 侧需要在“接收完成回调”里尽量在第一时间准备 ACK payload（例程会在 `rx_cb` 中调用 `PAN271_RF_SetTxData()` 封装 ACK 数据）。
4. 缩短距离、避开强干扰环境，必要时更换信道。

4.2.6 Q6：串口没有打印或出现乱码，怎么排查？

按例程默认配置，常见排查点：

1. 串口参数是否为 `115200，8N1`。
2. UART0 引脚复用是否正确：例程通常使用 `P0.5=TX`、`P0.6=RX`（见 `Sys_Init()`）。
3. 底板跳线/杜邦线是否按要求短接（部分例程文档提到 `J8/J9` 对应引脚的短接）。
4. PC 侧是否打开了正确的 COM 口（USB->UART 是否正常枚举）。

4.2.7 Q7：如何做单频载波发射用于射频测试？

驱动提供了载波相关接口：

- `PAN271_RF_StartCarrierWave(Channel)`：开始载波发射
- `PAN271_RF_StopCarrierWave()`：停止载波发射

一般流程是先完成 RF 初始化并设置信道/功率，再启动载波。注意在合规的测试环境下进行。

Chapter 5

量产测试

5.1 量产烧录

5.1.1 1 芯片烧录口硬件连接

对于遵循PAN271x 硬件参考设计 文档规范的方案硬件板，OTP 芯片可以通过 Panlink 方式进行烧录。

5.1.2 2 量产烧录工具

为配合 Panlink 2.0 烧录器进行量产烧录，我们提供了对应的 PC 上位机工具。也可以在 `04_TOOLS\量产烧录工具\PAN271x Download Tool\PAN271x Download Tool.exe` 找到。

下载

2.1 硬件准备

预先将 Panlink 2.0 通过 MiniUSB 线连接到 PC 电脑。

图 1: 图 2-1 Panlink 2.0 烧录器

如果 Panlink 2.0 固件程序不支持 PAN271 芯片烧录，则需要根据提示自动更新升级。

图 2: 图 2-2 MiniUSB 连接线

或按照帮助文档方法更新 Panlink 2.0 固件程序。

PAN-LINK2.0 硬件需要特殊处理才能够烧录 PAN271x，否则无法连接。

PAN-LINK2.0 **特殊处理**：如果已经做了特殊处理则无需再做特殊处理。

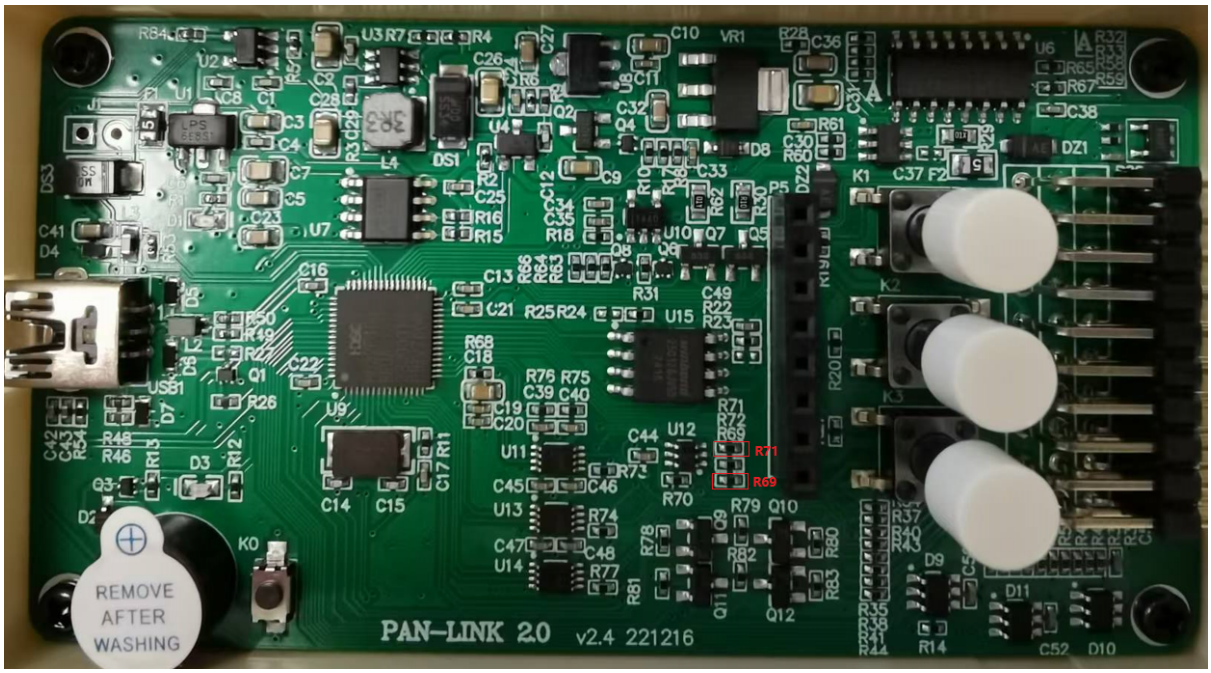

图 3: 图 2-3 Panlink 2.0 烧录器硬件需要特殊处理区域红色标注

PAN-LINK2.0 **硬件**	**处理说明**
R71 电阻	需要将原先的 220Ω 电阻更换为 1KΩ 电阻
R69 电阻	需要将原先的 10KΩ 电阻更换为 100 KΩ 以上阻值电阻

2.1.1 PAN271x 芯片烧录接线

Panlink 2.0	连接	PAN271x SoC
VDD	<—>	VBAT
(6V)	<—>	P22
GND	<—>	GND
A3	<—>	P01
A4	<—>	P00

注：(6V) PAN-LINK 的 6.5V 高压输出，提供烧录 PAN271x 芯片 OTP 用。默认不输出，只有在执行烧录时才会自动控制输出。

2.2 上位机工具界面

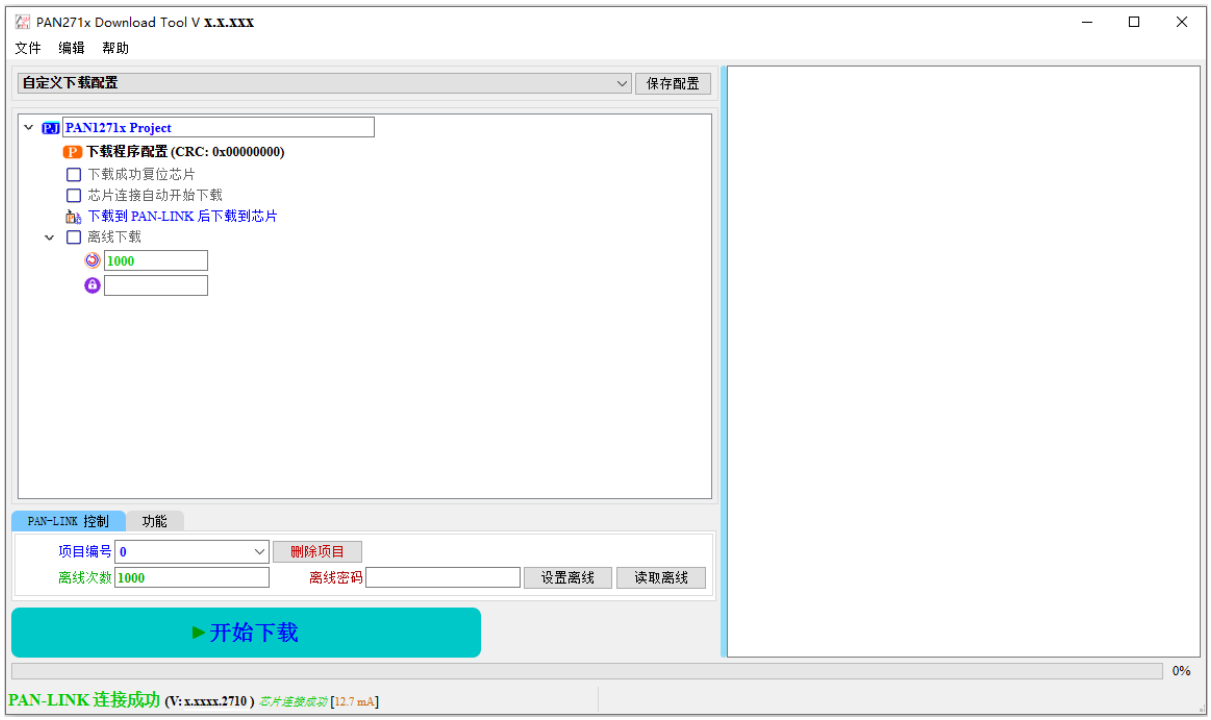

图 4: 图 2-3 烧录工具界面

如上图 2-3 所示为烧录工具界面。

1. 点击 **功能-> 获取芯片信息**，右侧会显示芯片信息和**芯片型号**
2. 根据右侧显示的**芯片型号**，在 **编辑-> 芯片**中选择对应的型号，防止芯片误烧
3. 在下载程序配置中的下载程序配置项右键点击加载程序

   PANLINK 上位机工具支持 .hex 和 .bin 两种固件格式。
4. 根据需求选择设置其他下载配置
5. 选择下载模式，或直接默认下载到 PAN-LINK 后下载到芯片模式
6. 点击下载开始下载程序到芯片

2.3 查看帮助文档

通过烧录工具的**帮助-> 查看帮助文档**或直接通过快捷键 F1, 打开查看帮助文档。

PAN-LINK2.0 程序更新方法、以及烧录工具的详细使用说明都在帮助文档中有详述。

图 5: 图 2-4 查看帮助文档

5.2 射频测试

5.2.1 1 功能概述

本文主要介绍 PAN271x RF 测试固件的使用。

5.2.2 2 环境要求

- PAN271x EVB 若干块
- USB 转串口工具若干块
- 硬件接线：
  - 使用**杜邦线**连接 EVB 和 USB 转串口工具:
    - UART1_Rx (P00) 与 USB 转串口工具 TX 连接
    - UART1_Tx (P01) 与 USB 转串口工具 RX 连接
- Panchip ToolBox 下载
- USB TYPE-C 线

5.2.3 3 RF 测试固件说明

NO	固件说明	下载链接	更新日期
1	RF 性能测试 (发射功率、频偏、EVM、对测收包率等)	PAN271x RF 测试固件	2025-10-16

5.2.4 4 演示说明

PAN271x EVB **板** PIN **脚接线说明：**

PAN271x EVB 板 GPIO	USB 转串口工具
UART1_Rx (P00)	UART_TX
UART1_Tx (P01)	UART_RX
VBAT (VDD)	VCC(3.3V)
GND	GND

RF 性能测试需通过 PAN271x Toolbox 工具箱工具进行测试：

1. 用 PANLink 或者 JLink 烧录固件 “PAN271x RF 测试固件”。

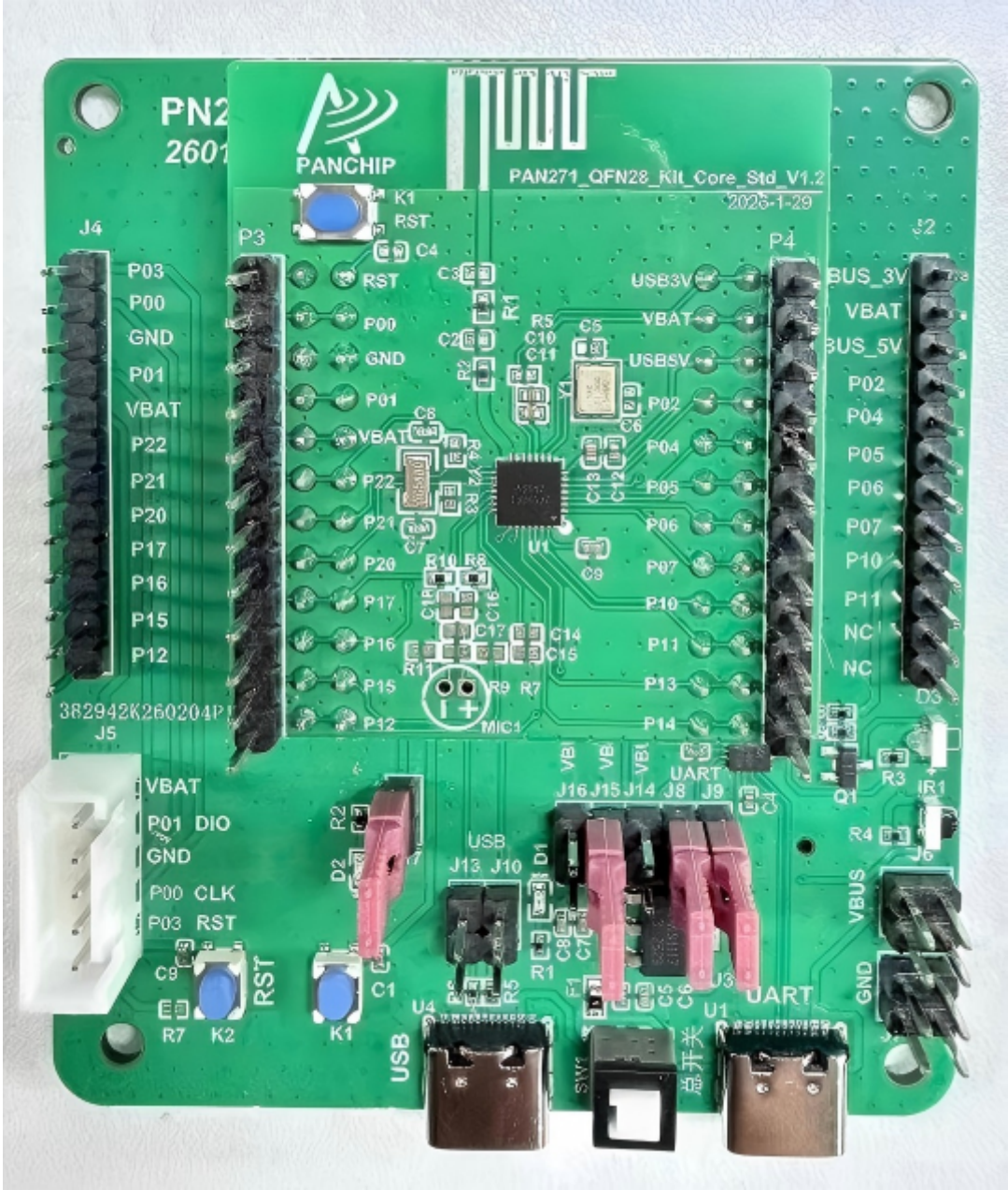

图 6: 图 1 PAN271 SRAM EVB 板

PAN271x RF 测试固件可从前面列出的 WiKi 链接下载，也可直接从 PAN271x DK 开发套件中找到（位于：`<PAN271x-DK>/04_TOOLS/RF` 测试固件）。

2. 测试流程可参考Panchip Toolbox 工具箱 文档第一章 RF 测试说明。

   UART 通信波特率: 115200

典型的测试场景

- TX 测试，可以通过软件发送单载波和 dtm 数据包（下图是一个单载波的典型配置界面），通过频谱仪观察射频状况

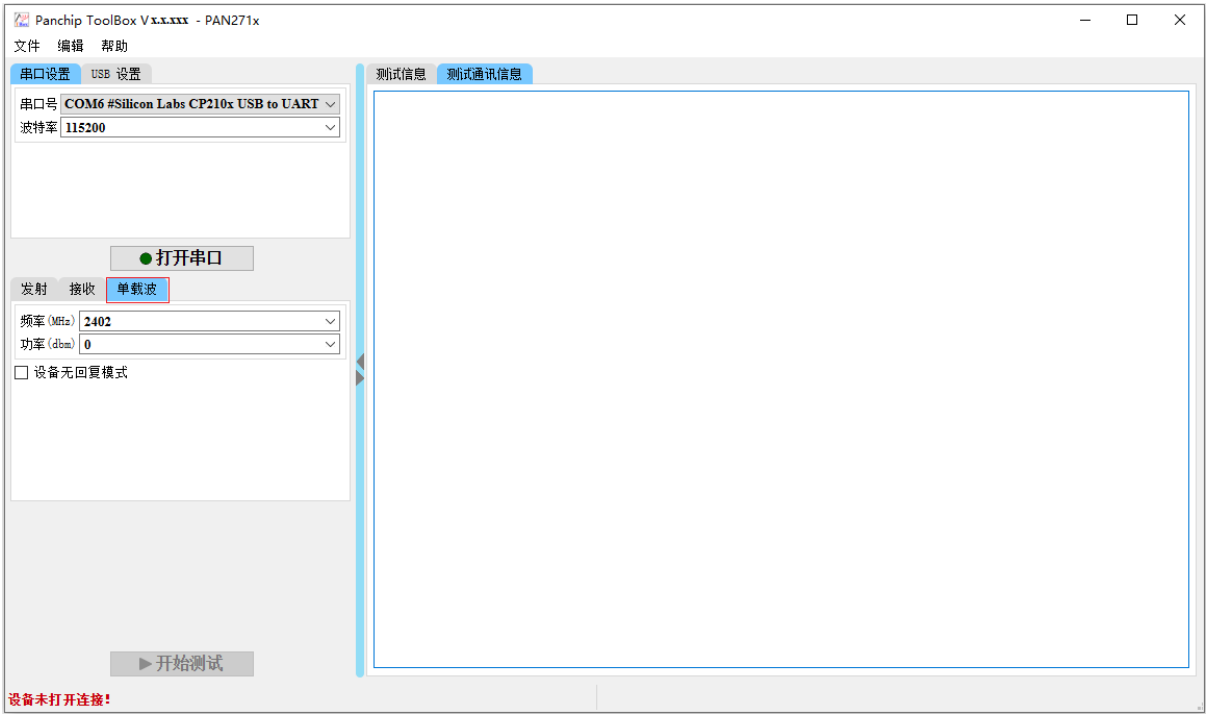

图 7: 图-2 单载波配置界面

- RX 测试可以打开 2 个软件，控制 2 个 PAN271x 芯片，一个设置为 tx 模式，一个设置为 rx 模式，tx 端设置 tx counts 发送，rx 端打印收到的 counts，进而判断 rx 的质量和灵敏度。

Tx 端：

Tx 端打开串口，配置好发送选项卡参数后，点击开始测试，上位机会发送 1000 包数据；

Rx 端：

Rx 端打开串口，配置好接收选项卡参数后，点击开始测试会进入接收状态。点击 `Read Rx Cnt` 或者停止测试后，接收端会打印出收到的包数和 RSSI；

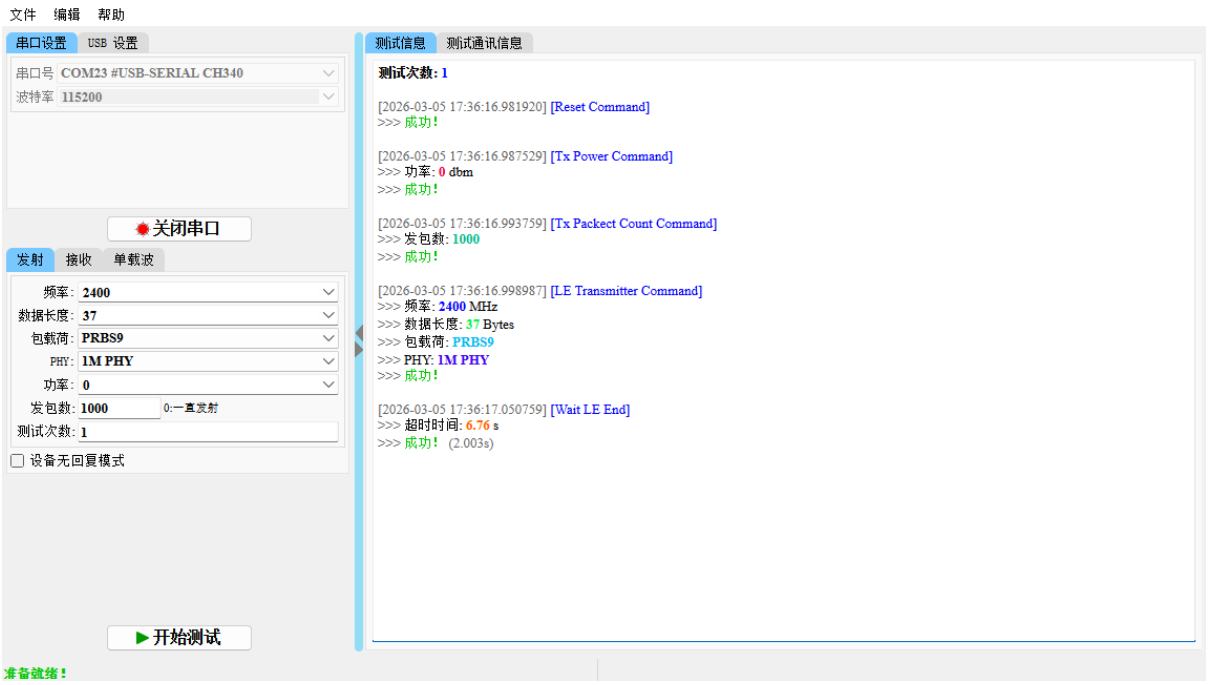

图 8: 图-3 Toolbox 发送端

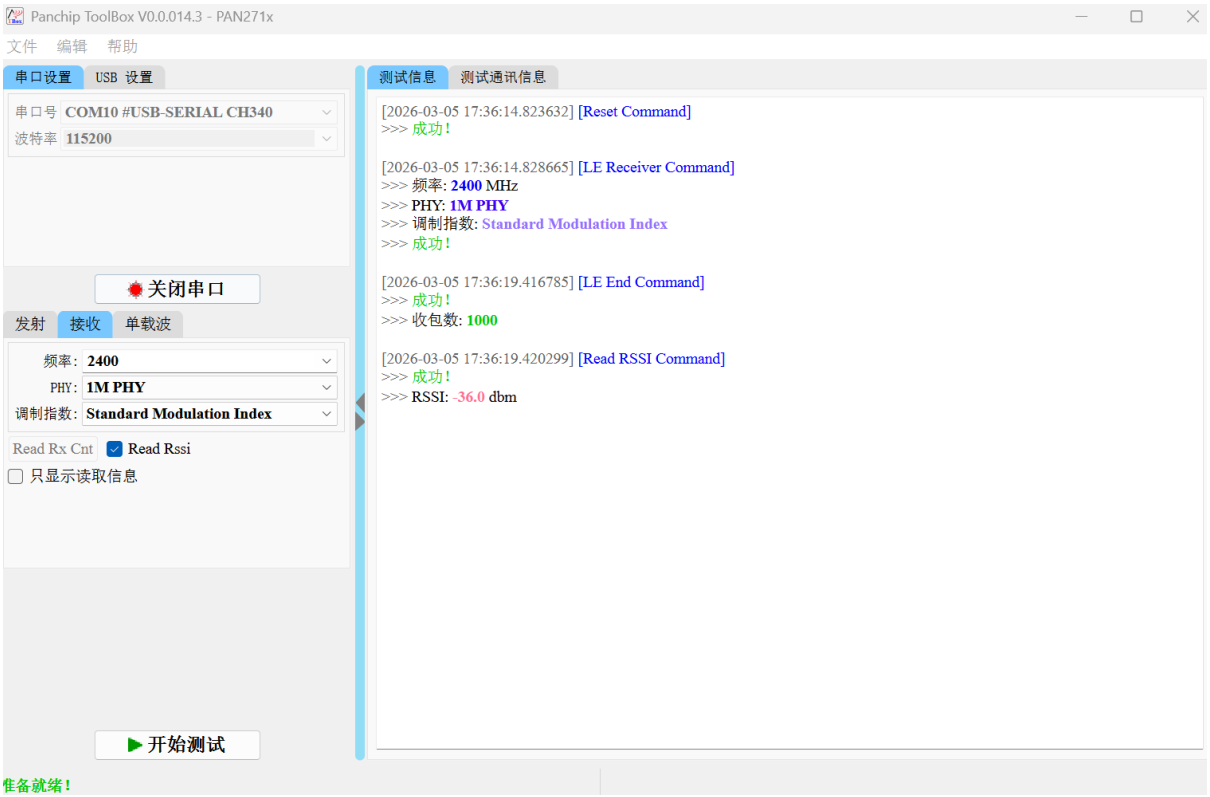

图 9: 图-4 Toolbox 接收端

Chapter 6

开发工具

6.1 Panchip Toolbox 工具箱

Panchip Toolbox 是 Panchip SoC 开发工具集合，目前包含如下功能：

- RF 测试
- 芯片引脚配置（TBD）

下载工具。

6.1.1 启动界面选择

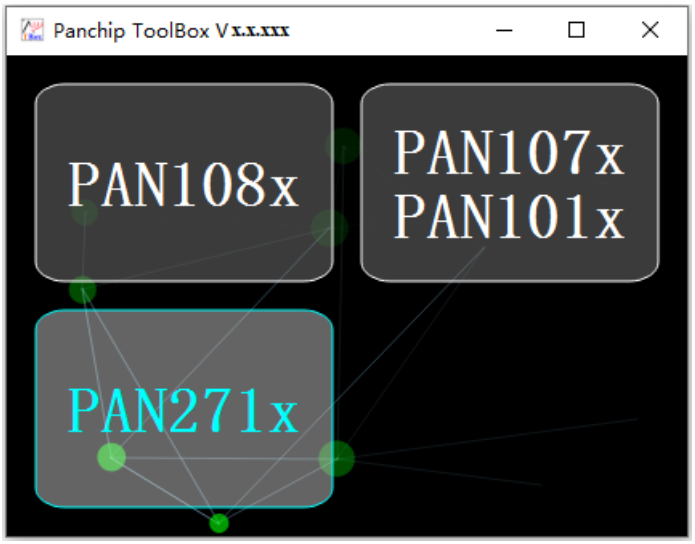

图 1: 启动界面选择 PAN271xx 进入 ToolBox 功能界面

6.1.2 功能界面选择

6.1.3 1. RF 测试

此功能需要配合 PAN271x 芯片的 RF 测试固件（位于：`<PAN271x-DK>/04_TOOLS/RF 测试固件`）使用。

支持通过串口通讯或 USB 通讯测试 2.4G 的**发射**、**接收收包数**、**单载波发射**等功能。

支持通过 USB 通讯测试 2.4G 的**单载波发射**、**跳频发射**、**发射**、**接收收包数**等功能。

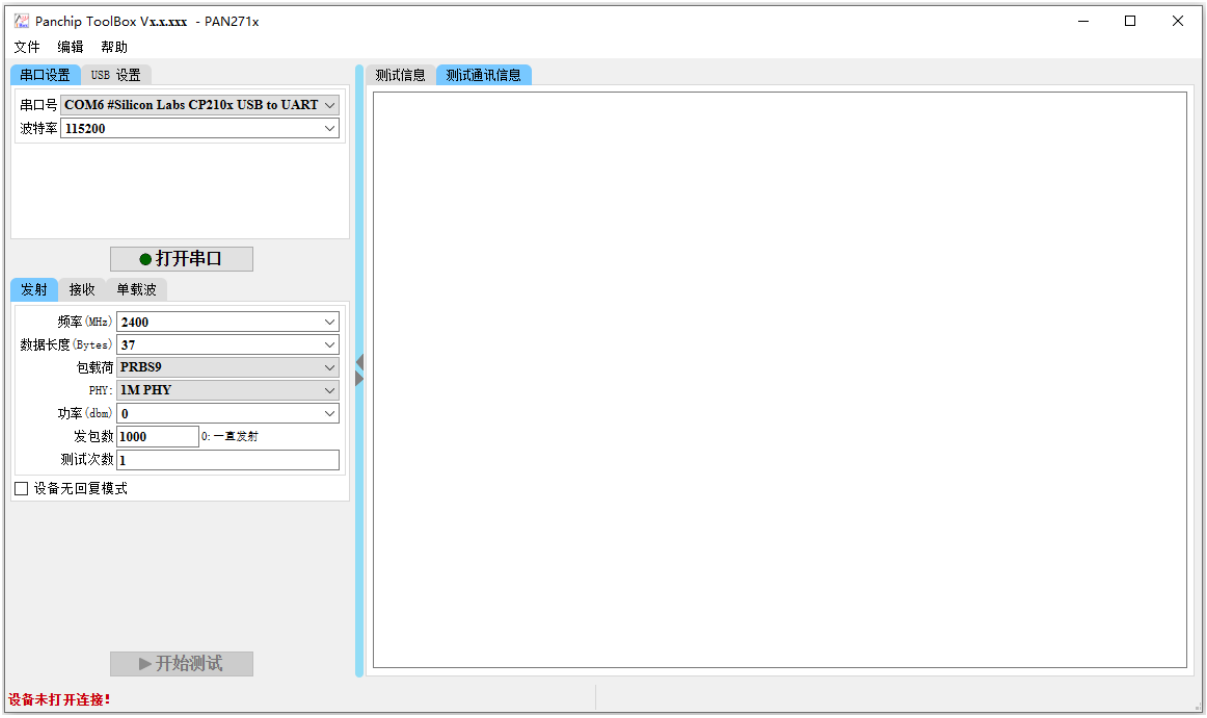

图 2: PAN271x ToolBox 功能界面

图 3: 显示切换功能界面

1.1. RF 测试界面

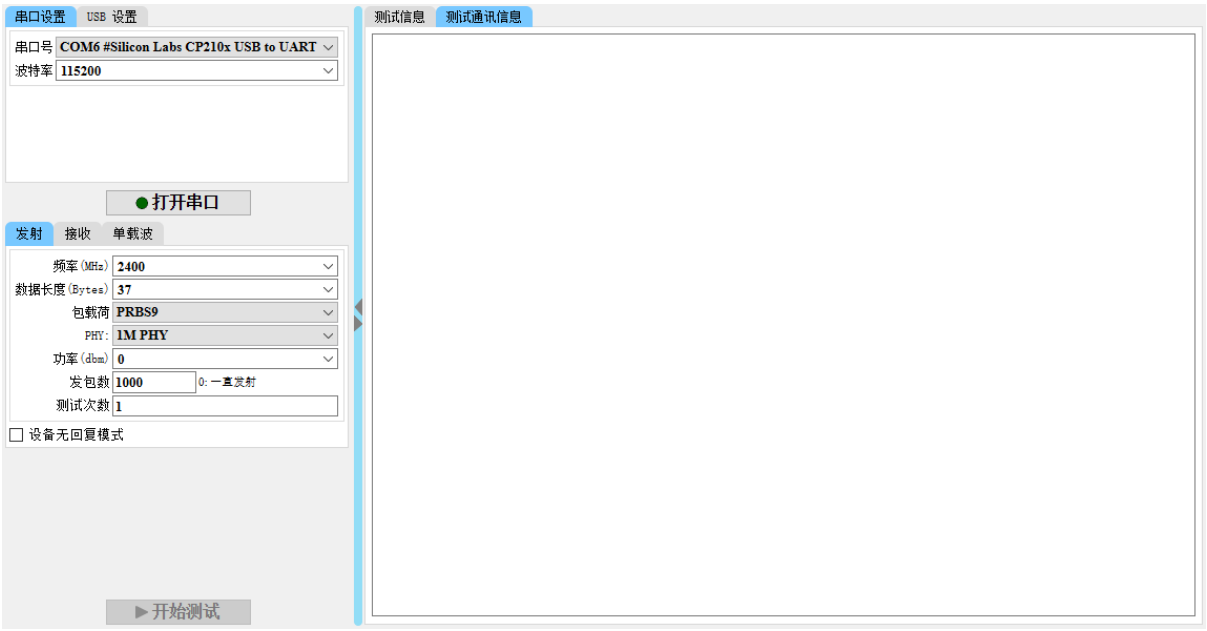

图 4: RF 测试界面

6.1.4 2. 芯片引脚配置

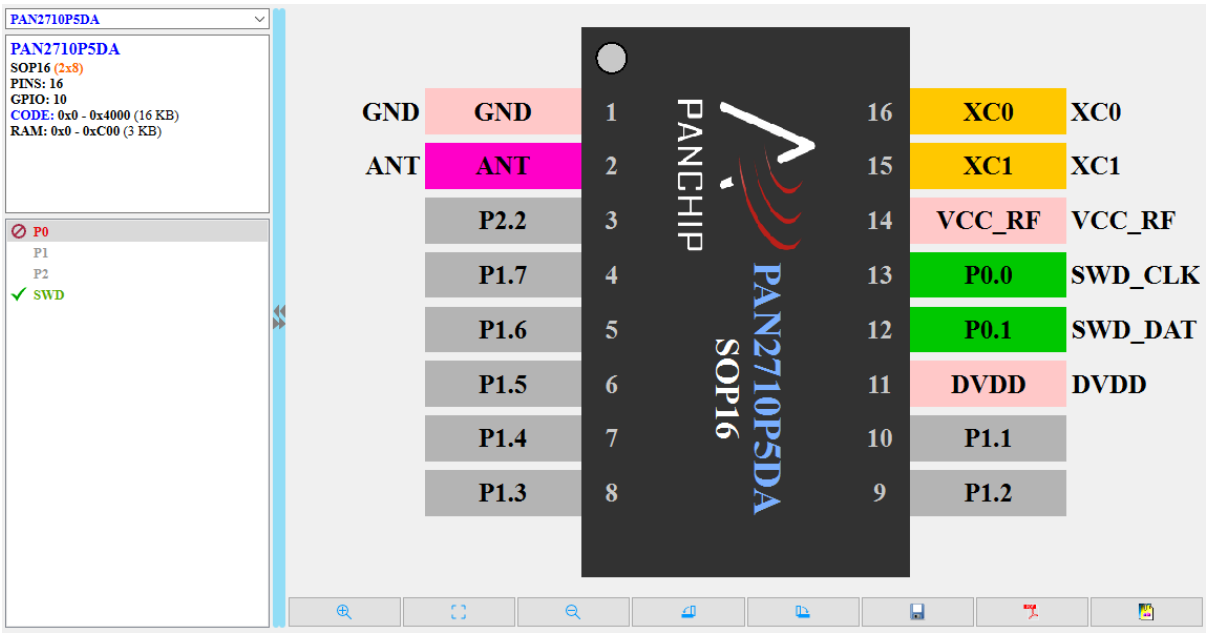

图 5: 芯片引脚配置界面

Chapter 7

其他文档

Chapter 8

更新日志

8.1 PAN271x-dk-v0.3.0 更新日志

8.1.1 1.SDK

Build Tools 目录

- scripts：添加 ImageTool 加密脚本，用于对芯片进行加密，防止 Code 和 Date 被读出。

Proprietary RF 私有 2.4G 目录

- 在 Tools 目录下，添加代码导出工具。可以根据用户的需求配置不同的 RF 模式和参数。

Samples 演示例程目录

目前版本提供了如下演示例程：

- drivers/adc：ADC 驱动演示例程
- drivers/gpio：GPIO 驱动演示例程
- drivers/pwm：PWM 驱动演示例程
- drivers/timer：Timer 驱动演示例程
- drivers/uart：UART 驱动演示例程
- drivers/wdt：WDT 驱动演示例程
- proprietary_rf/normal_trx：PRF 2.4G PAN271 普通型收发演示例程
- proprietary_rf/enhance_trx：PRF 2.4G PAN271 增强型收发演示例程
- proprietary_rf/clock_16m_trx：PRF 2.4G 16M 晶振收发延时例程
- proprietary_rf/rf_lp_tx：PRF 2.4G 普通型发送和低功耗切换例程

ADC:

- 优化 ADC 低电压检测逻辑，提升低电压判定的准确性与稳定性。
- 优化 ADC 检测时序，缩短检测耗时并提升检测效率。

8.1.2 2. HDK

目前版本提供了如下硬件相关资料：

- PN271_Kit_Base_V1.0：PAN271x EVB 底板硬件设计资料（原理图、PCB 文件等）
- PAN271_QFN28_Kit_Core_Std_V1.2: PAN271x QFN28 核心板硬件设计资料（原理图、PCB 文件等）

8.1.3 3. DOC

目前版本提供了如下文档：

- SDK 快速入门指南
- SDK 开发环境搭建
- PAN271x EVB 硬件资源介绍
- PAN271x 硬件参考设计指南
- 外设驱动例程
- 私有 2.4G 例程
- PRF 2.4G 开发指南
- 常见问题（FAQs）
- 量产烧录流程与工具介绍
- 射频测试流程与工具介绍
- Panchip ToolBox 工具箱介绍

8.1.4 4. TOOLS

目前版本提供了如下工具（材料）：

- PANLink 量产烧录工具（PC 工具）
- Panchip ToolBox 工具箱工具（PC 工具，包含 RF 测试固件）
- 代码导出工具（PC 工具）

8.2 PAN271x-dk-v0.2.0-lite-rc2 更新日志

8.2.1 1.SDK

GPIO

1. 统一 GPIO 模式命名规范，废弃 `GPIO_MODE_OUTPUT`、`GPIO_Resved` 表述，改用 `GPIO_MODE_PushPull`、`GPIO_Mode_OpenDrain`，提升接口语义清晰度；
2. 移除句柄及句柄初始化相关代码，简化接口实现；

LowPower

1. 删除部分冗余代码，精简低功耗模块实现；
2. **优化进入低功耗模式流程：正式进入低功耗前开启快速起振，唤醒延迟后关闭快速起振功能；**

3. 将 LowPower 例程的 GPIO 唤醒，从高电平唤醒切换为上升沿唤醒，更加满足实际的现场需求；(上升沿唤醒要调用 GPIO_SetDebounceTime 函数，不然无法正常唤醒)

CLK

1. 优化 `systick_delay_ms` 实现，解除原先仅支持 500ms 以内延迟的限制；
2. **调整 `void Clock_Init(void)` 初始化流程逻辑；**

ADC

1. 新增 VBG 档位外部电压检测功能；
2. 将部分仅被单次调用的函数声明为 `inline`，减少代码体积；
3. 优化 ADC 校准及初始化流程，进一步压缩代码空间；
4. 外部接口不再使用 Channel 作为 GPIO Pin 选择方式，统一改为使用 Pin + Port；
5. 增加 VDD 档位与 VBG 档位的宏定义开关，支持用户按需裁剪功能；
6. 清理部分冗余代码；

RF

1. AGC 初始化寄存器配置类型由原类型调整为 `uint32_t`，避免栈空间溢出风险；
2. 修复增强型模型下接收 ACK 不携带 Payload 的问题；
3. **修复** PHY **跨时钟域场景下可能导致卡死的问题；**
4. 已经为射频 trx 例程的 fifo 大小添加说明，空间大小必须是 4 的倍数；
5. 添加 03_lp_rf 例程，此例程是在低功耗模式和射频发送模式下的切换例程；

加密功能

1. 增加加密文件生成插件，配置芯片的加密信息;
2. 通过配置 4 组 key(verify-key1~4) 组合计算实际加密密钥，不直接使用实际密钥;
3. 量产烧录工具导入加密文件后进行烧录，完成芯片的加密信息写入;
4. 具体使用方法请看《\build_tools\scripts\ImageTool 加密脚本使用指南.pdf》；

其他

1. 统一管理所有日志打印接口，便于调试与维护；
2. 默认关闭 `mini_printf` 的 `%s` 打印功能以减少代码体积；
3. 新增部分寄存器位段宏定义开关，便于功能裁剪；
4. 在所有例程的 keil Taget 选项卡中，设置 OTP 大小和 RAM 大小，并勾选 Linker 中的 Use Memory Layout from Target Dialog 选项。

JLINK 烧录

1. **修改** FLM **文件，烧录完成后自动执行** Chip Reset**；**
2. 增加烧录芯片型号校验机制，防止误烧录；

PANLINK 烧录

1. 修复“直接下载到芯片模式”下烧录失败的问题；
2. 优化 PAN-LINK OTP 烧录机制：单次失败后自动重试 2 次，连续 3 次失败才退出，提升烧录成功率；
3. 新增滚码烧录接口，支持将滚码写入芯片 MAIN 区；
4. 新增上位机控制接口，支持：
   - 触发 PAN-LINK 离线烧录；
   - 读取当前项目滚码信息；
   - 切换 PAN-LINK 当前项目；
5. 增加烧录芯片型号校验；
6. 增加上位机读取芯片信息功能；
7. 增加下载过程加密机制；

8.3 PAN271x DK v0.1.1 更新日志

PAN271x DK `v0.1.1` (2025-11-22) 已发布：

8.3.1 1. SDK

Drivers 外设驱动目录

- 在 pan_otp.c/h 中添加 FT Info 区校准函数，并在所有例程中进行 FT 校准。
- 在 pan_adc.c/h 中添加 adc 校准代码，当对应的 Info 区为非空时进行校准。
- 在 pan_utility.c/h 添加 CRC8 的运算逻辑，可对 Info 区数据进行校验。

Samples 演示例程目录

- 在时钟初始化中，添加 FSYNXO RDY 延时控制位。
- 添加用于 FT 测试的代码，用于 OTP 的 Info 校准。
- 修改部分 AT 例程指令逻辑，并匹配拉距测试环境。

Proprietary_RF 目录

- 修改 RF Init 初始化寄存器列表。

8.4 PAN271x DK v0.1.0 更新日志

PAN271x DK `v0.1.0` (2025-10-17) 已发布：

8.4.1 1. SDK

Build Tools 目录

- JFlash：精简版本的 JFlash 工具，用于 Keil 工程烧录。

Components 系统组件目录

- PAN-USB：Panchip USB 协议组件

Drivers 外设驱动目录

目前版本提供了如下外设驱动：

- ADC Driver
- CLK Driver
- ClkTrim Driver
- GPIO Driver
- I2C Driver
- KeyScan Driver
- LowPower Driver
- OTP Driver
- SPI Driver
- SYS Driver
- Timer Driver
- UART Driver
- WDT Driver

Platform 平台目录

存放 PAN271x 平台相关代码，包括芯片启动代码、平台初始化代码、Log 机制代码等。

Proprietary RF 私有 2.4G 目录

存放 PAN271x PRF 2.4G 相关代码，包括 2.4G Lib、2.4G API 接口等。

Samples 演示例程目录

目前版本提供了如下演示例程：

- components/pan_usb_mouse：PAN-USB HID Mouse 演示例程
- drivers/adc：ADC 驱动演示例程
- drivers/clktrim：ClkTrim 驱动演示例程
- drivers/gpio：GPIO 驱动演示例程
- drivers/i2c：I2C 驱动演示例程
- drivers/kscan：KeyScan 驱动演示例程
- drivers/otp：OTP 驱动演示例程
- drivers/pwm：PWM 驱动演示例程
- drivers/spi：SPI 驱动演示例程
- drivers/timer：Timer 驱动演示例程
- drivers/uart：UART 驱动演示例程

- drivers/wdt：WDT 驱动演示例程
- miscellaneous/blinky：EVB LED 演示例程
- miscellaneous/coremark：CoreMark 基准测试演示例程
- miscellaneous/debug_proect：SWD Debug Protect 演示例程
- miscellaneous/low_power：低功耗演示例程
- miscellaneous/reset：芯片 Reset 复位特性演示例程
- proprietary_rf/prf_rx：PRF 2.4G 接收演示例程
- proprietary_rf/prf_tx：PRF 2.4G 发送演示例程
- solutions/prf_dongle：2.4G Dongle 解决方案例程

8.4.2 2. HDK

目前版本提供了如下硬件相关资料：

- PAN271x_BaseBoard_V1.0：PAN271x EVB 底板硬件设计资料（原理图、PCB 文件等）和生产资料（BOM、gerber、坐标等文件）
- PAN271x_QFN28_CoreBoard_Std_V1.0: PAN271x QFN28 核心板硬件设计资料（原理图、PCB 文件等）和生产资料（BOM、gerber、坐标等文件）

8.4.3 3. DOC

目前版本提供了如下文档：

- SDK 快速入门指南
- SDK 开发环境搭建
- PAN271x EVB 硬件资源介绍
- PAN271x 硬件参考设计指南
- 19 个演示例程介绍
- 1 个解决方案介绍（2.4G Dongle）
- PRF 2.4G 开发指南
- 常见问题（FAQs）
- 量产烧录流程与工具介绍
- 射频测试流程与工具介绍
- Panchip ToolBox 工具箱介绍

8.4.4 4. TOOLS

目前版本提供了如下工具（材料）：

- PANLink 量产烧录工具（PC 工具）
- Panchip ToolBox 工具箱工具（PC 工具）
- RF 测试固件（芯片固件）