PAN271x 开发套件使用手册

发布 0.1.0

Panchip PRF SoC Team

2025 年 10 月 17 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 种 EVB 核心板，1 种 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/方案等例程工程
  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（V1.0）插到 EVB 底板（V1.0）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 Blinky 例程（演示 GPIO 推挽输出控制 LED 灯闪烁）：`<PAN271x-DK>\01_SDK\samples\miscellaneous\blinky`
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 @ 32000000Hz
     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

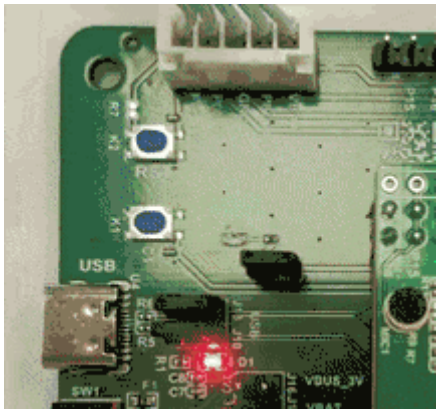


图 1: Blinky LED on EVB

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

---

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：
   - 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。

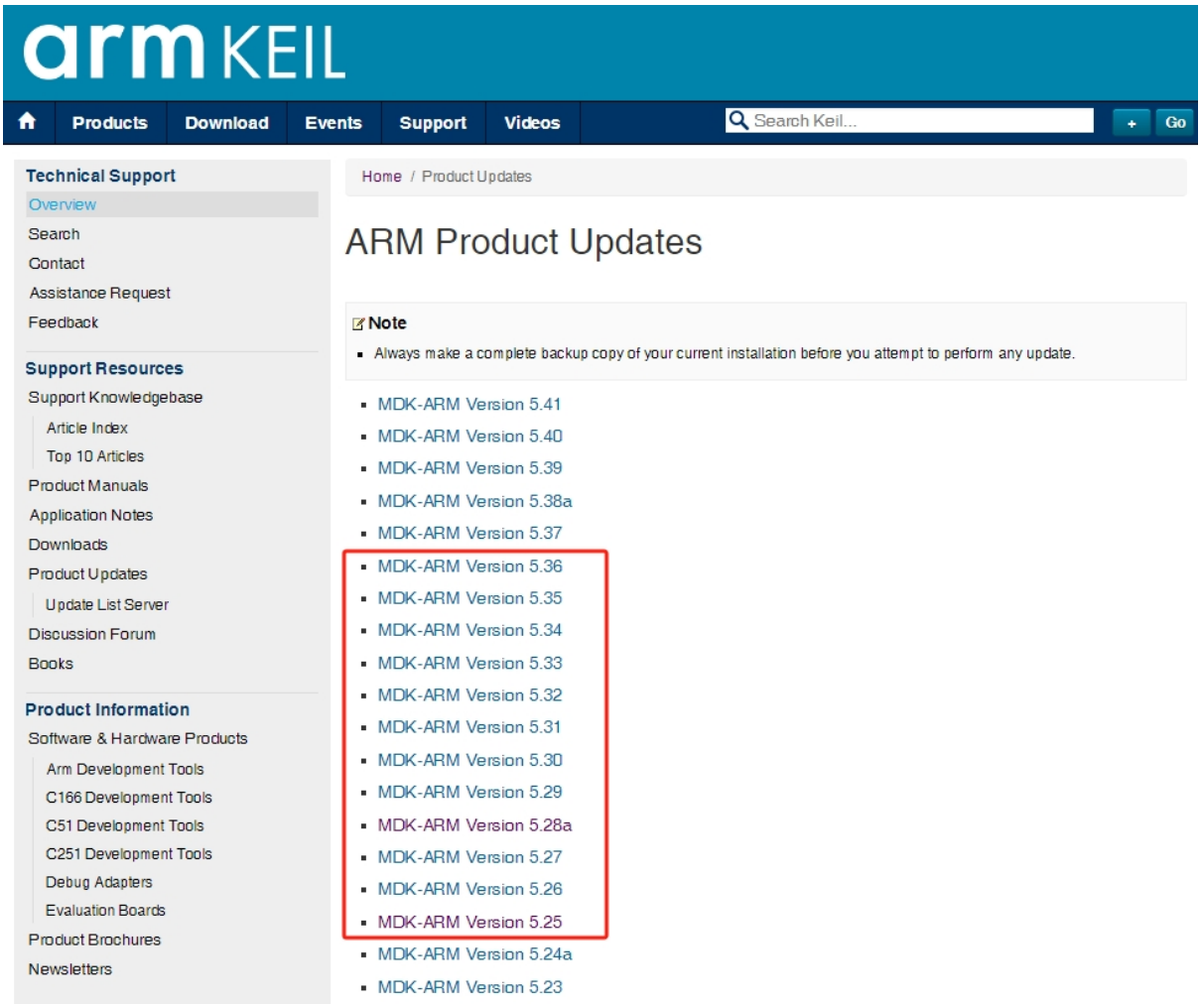


图 2: Keil MDK v5 历史版本下载页面

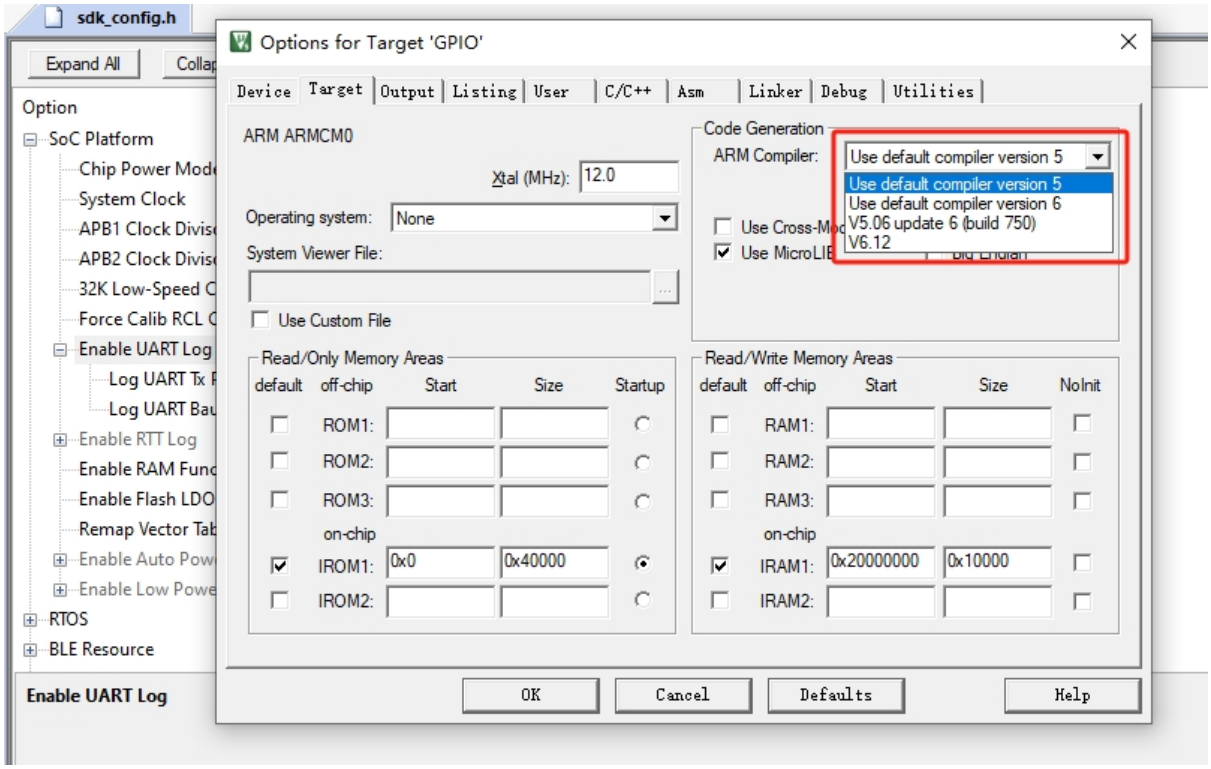


图 3: Keil 编译器版本选择

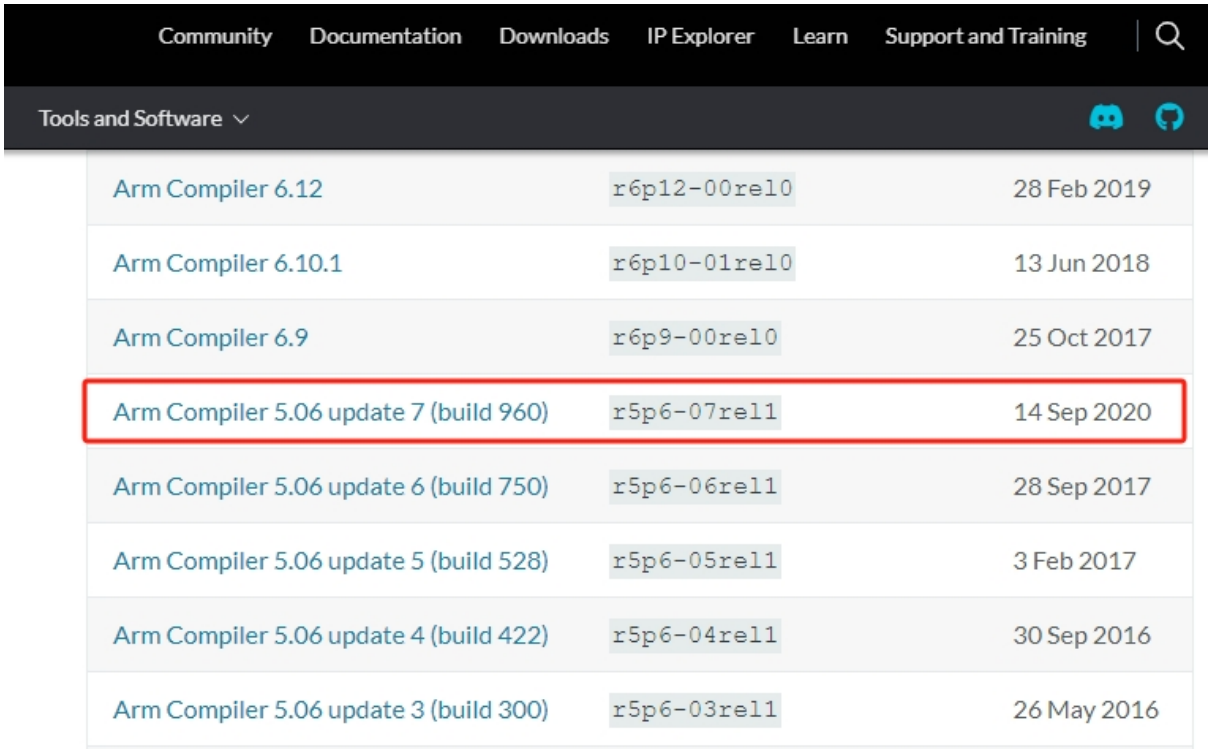


图 4: ARMCC 编译器下载

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 (v1.0) 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**）

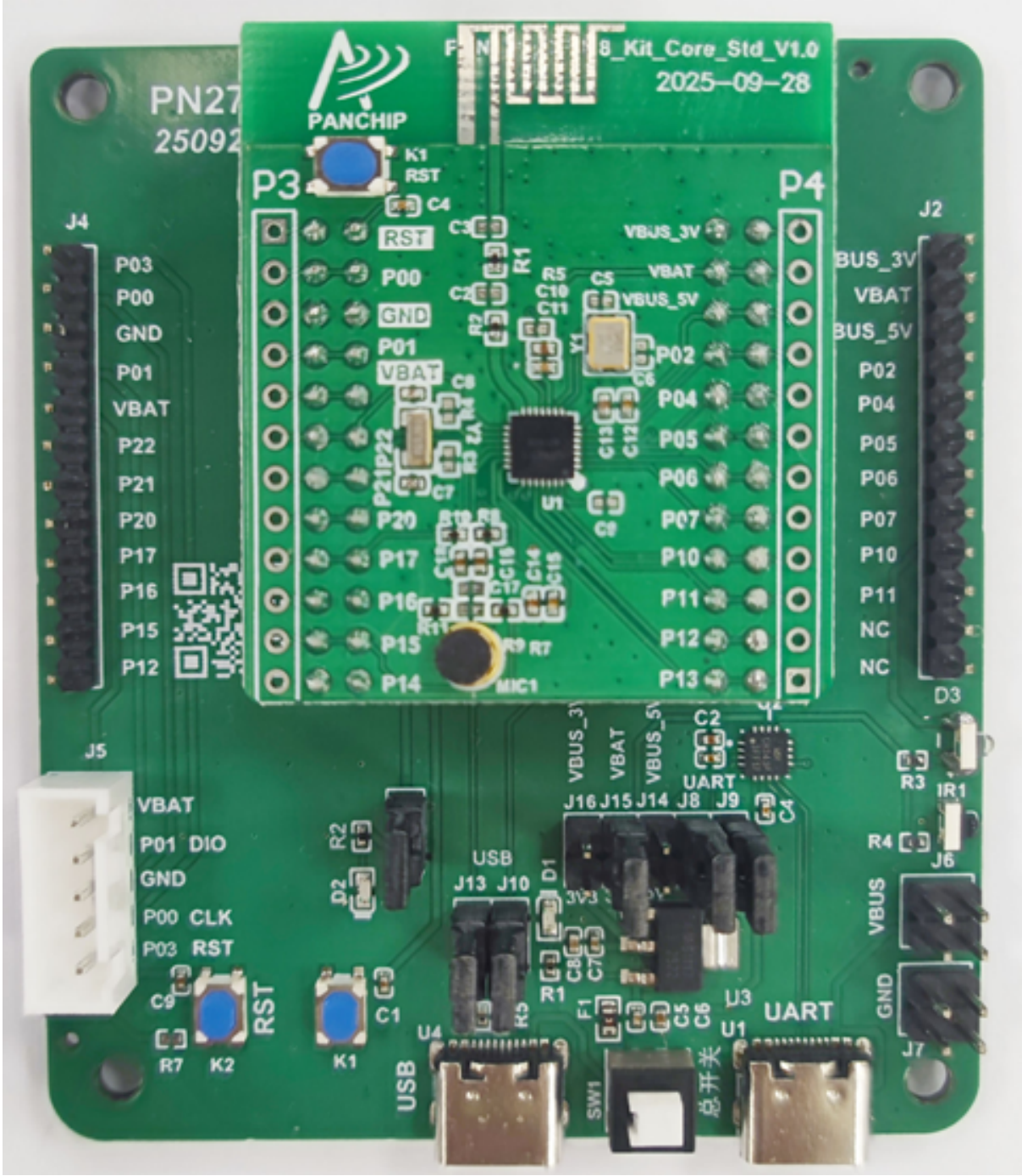


图 1: PAN271x EVB V1.0 实物图（核心版 + 底板）

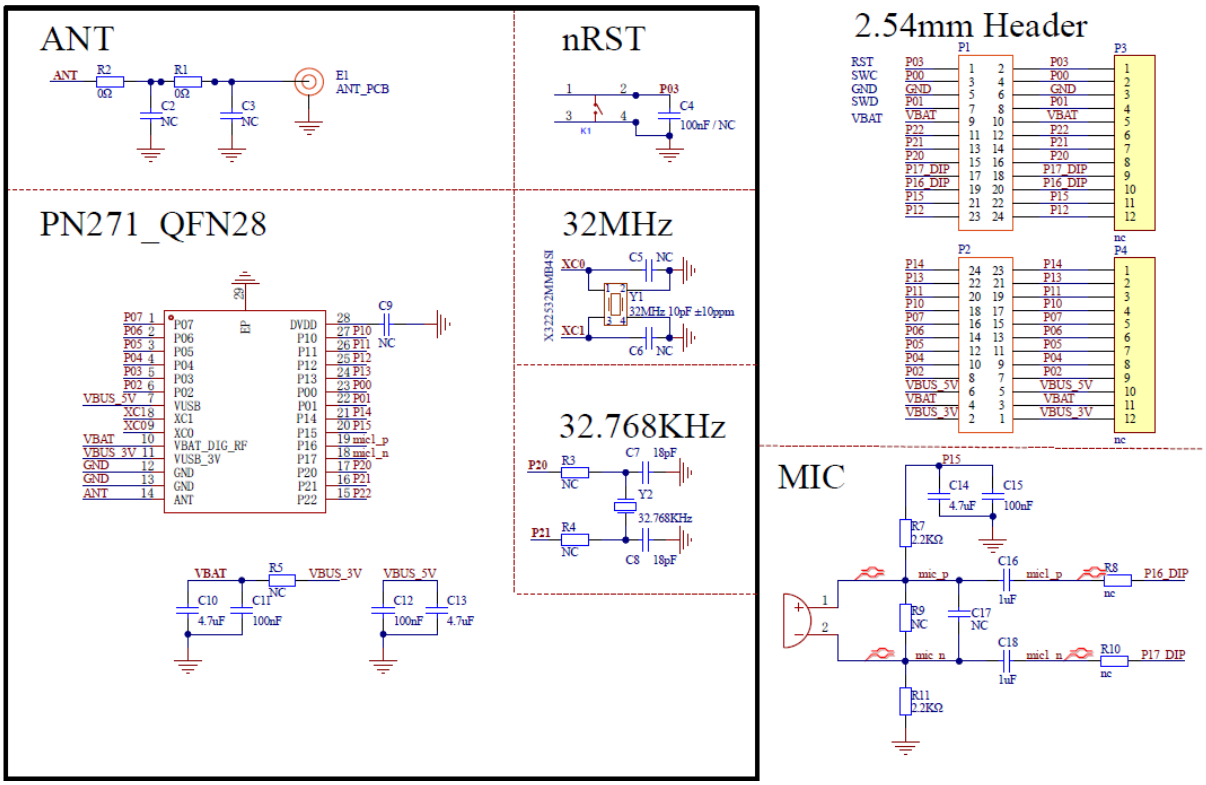


图 2: PAN2710U5GA 核心板原理图

- 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。

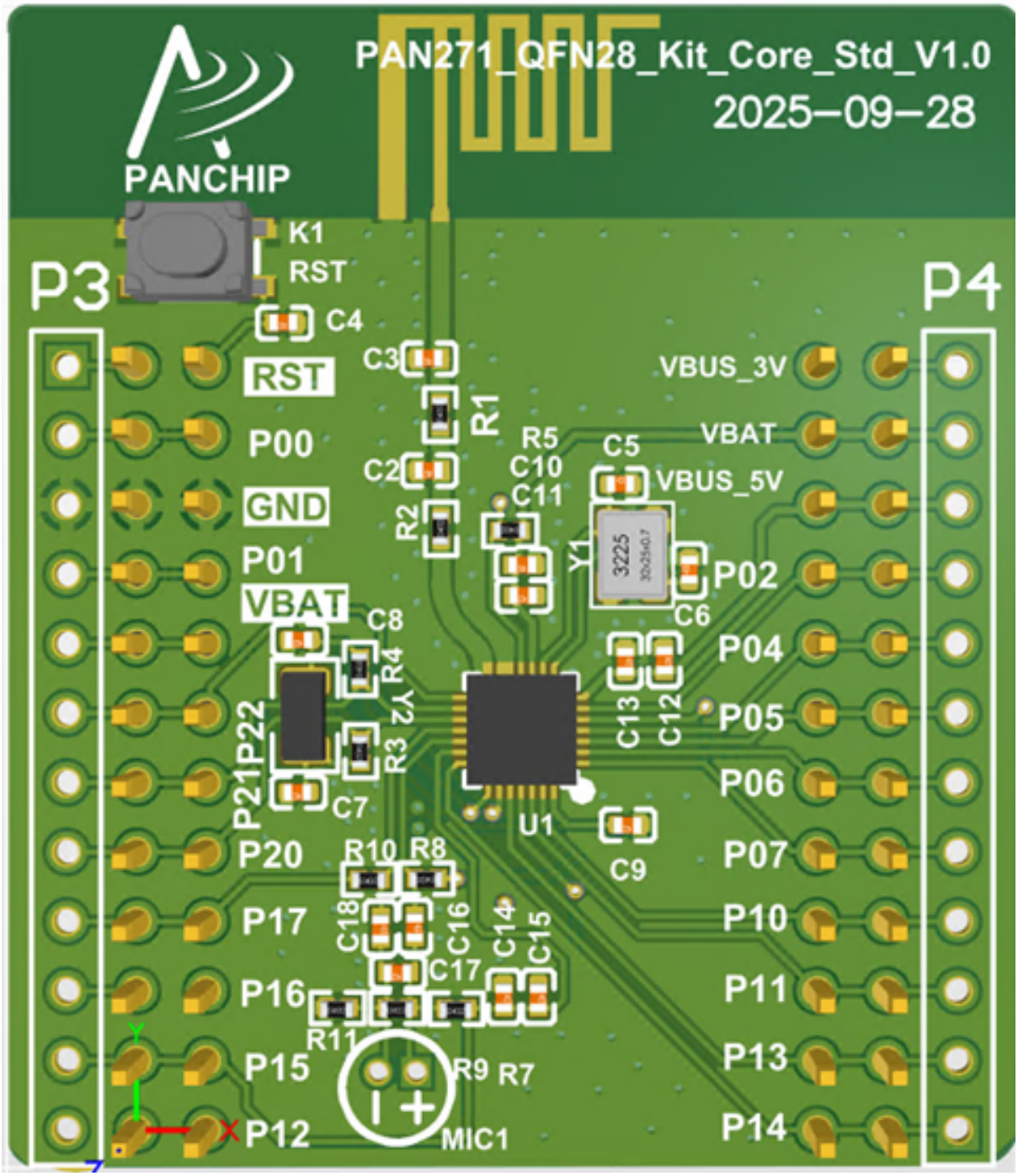


图 3: PAN2710U5GA 核心板三维图

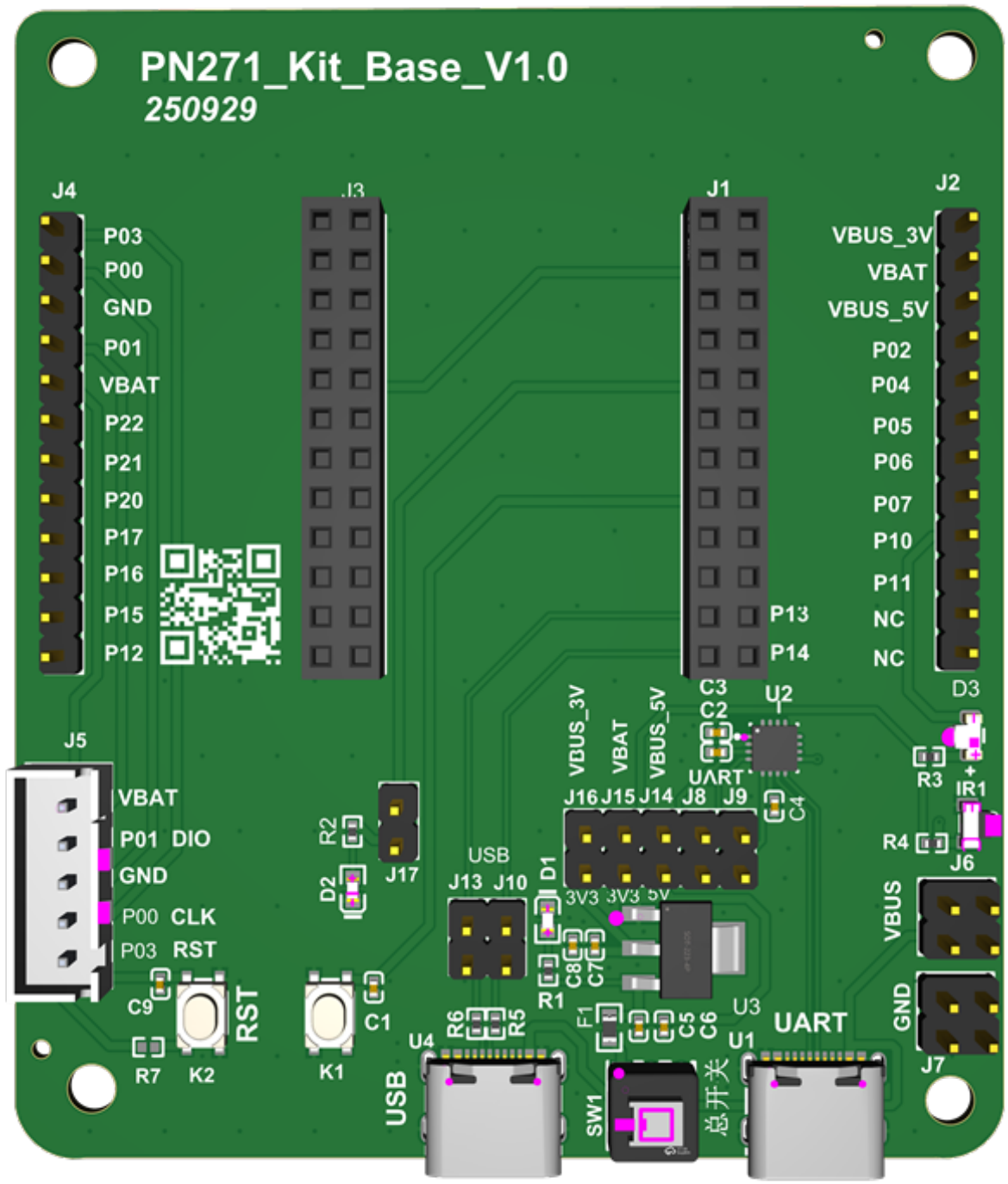


图 4: PAN271x EVB 底板三维图

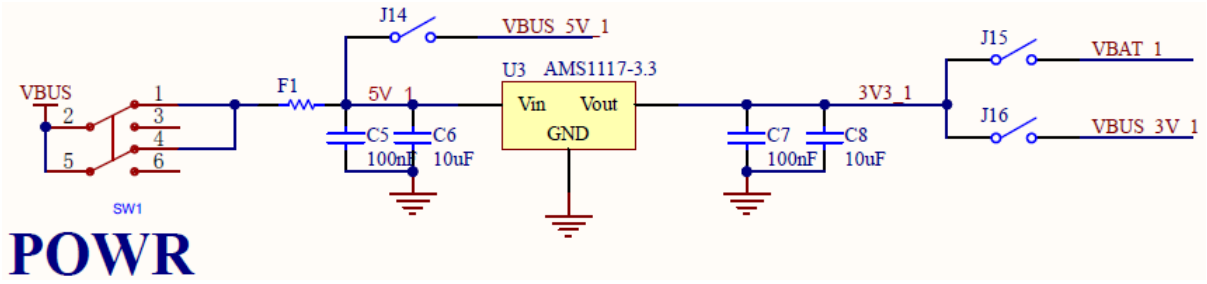


图 5: 电源模块原理图

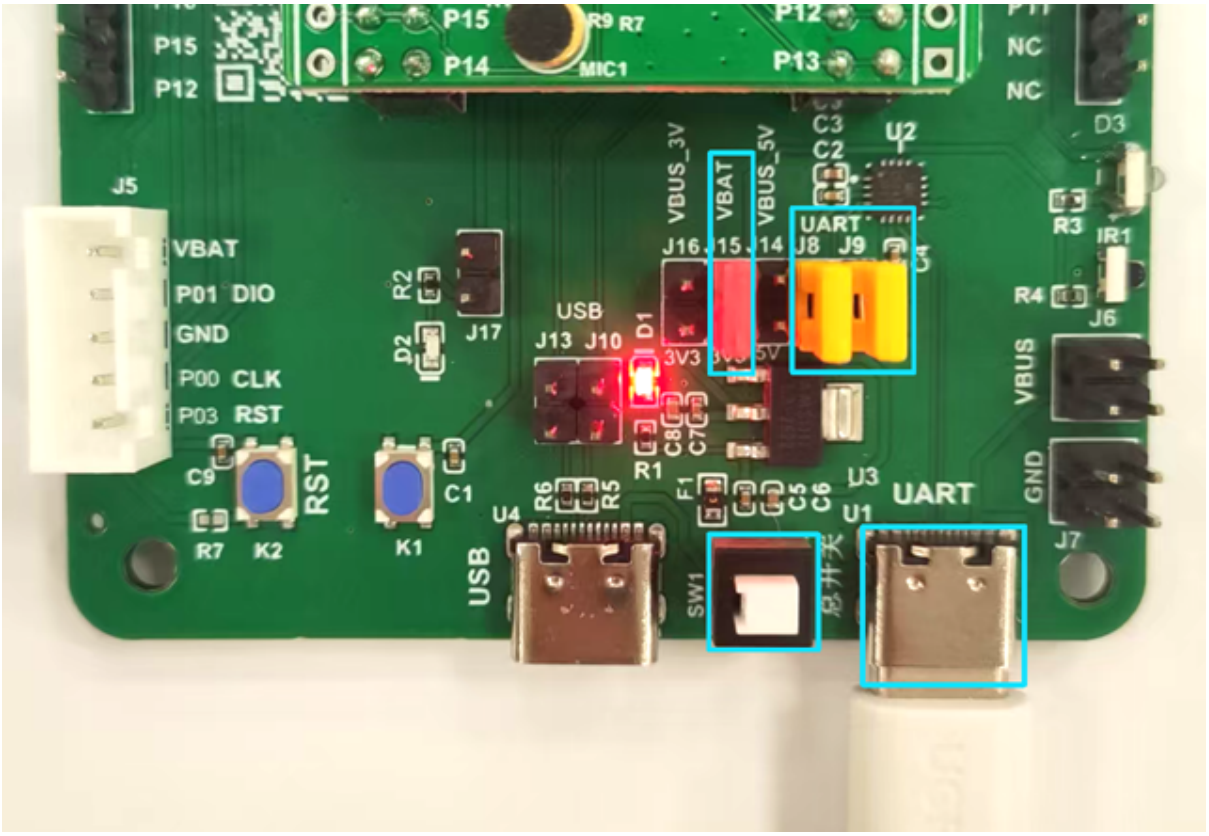


图 6: 典型供电方式示意图

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

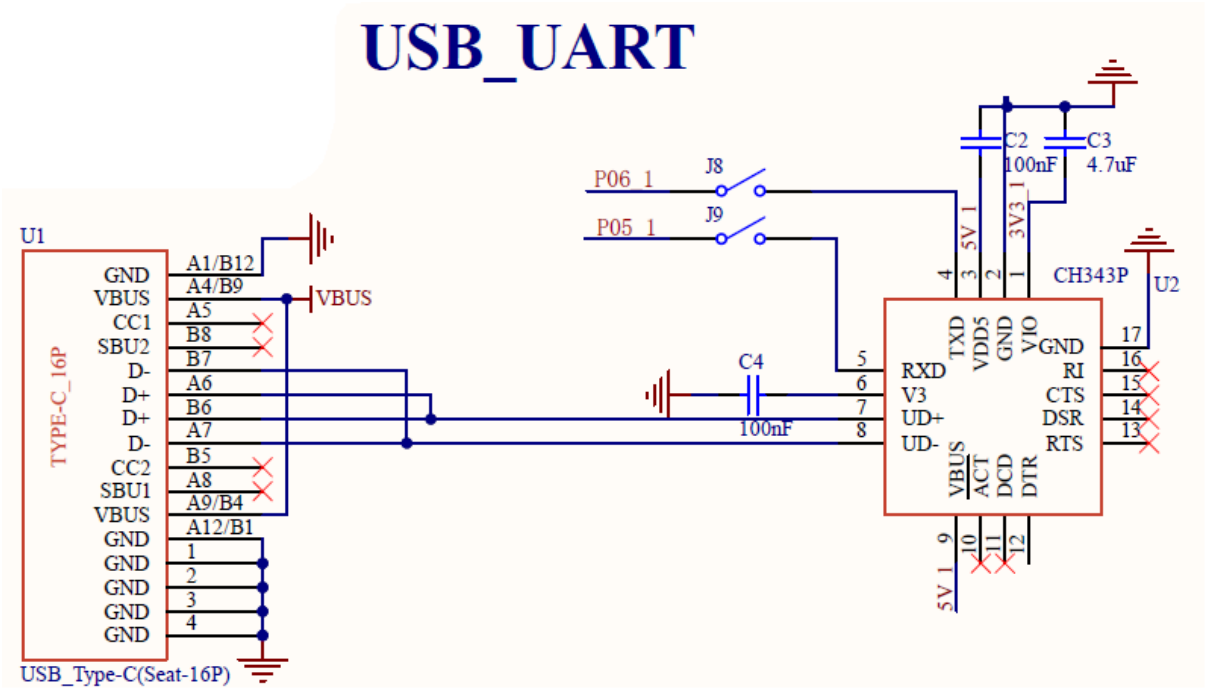


图 8: USB 转 UART 模块原理图

- 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。

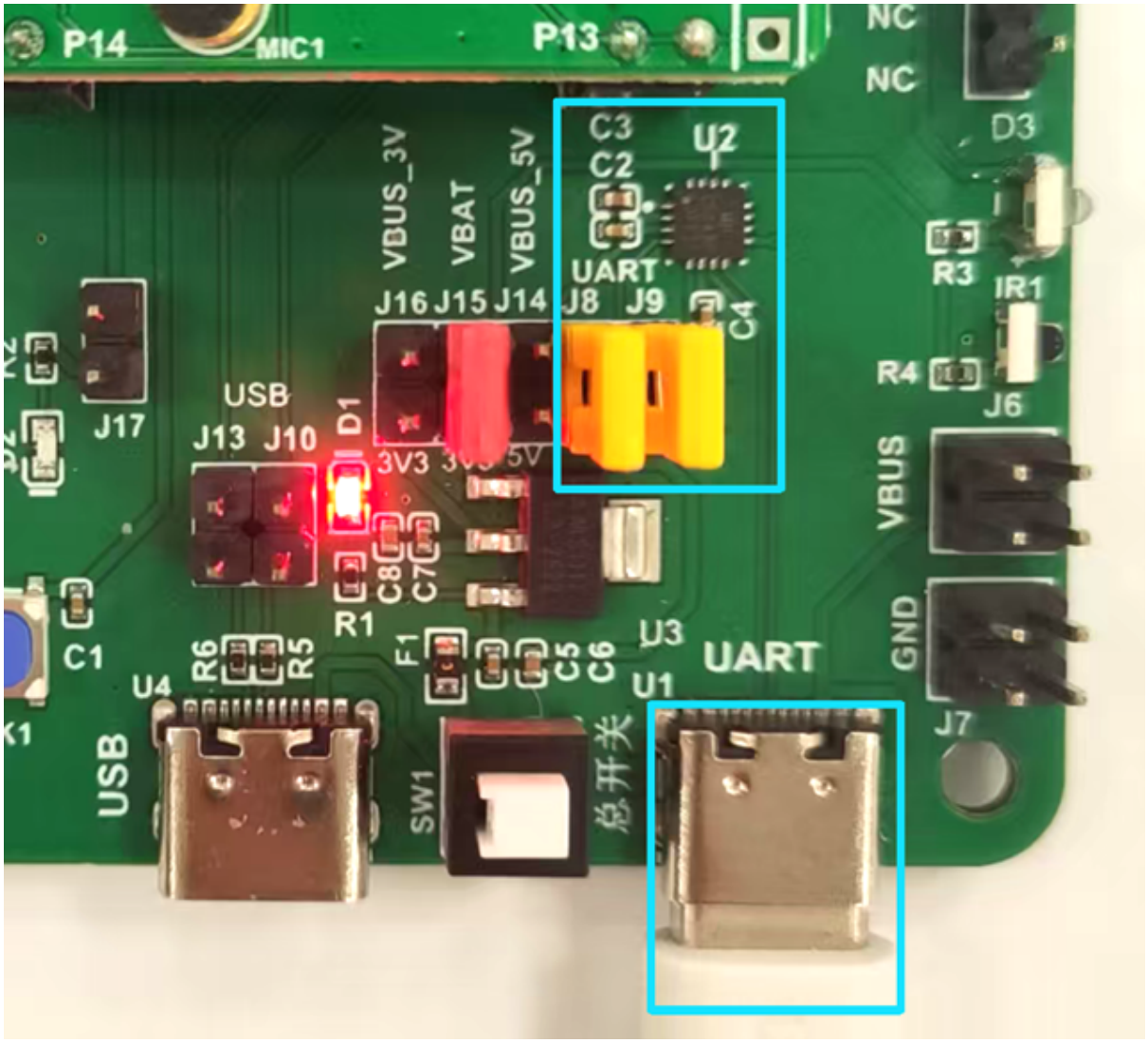


图 9: USB 转 UART 模块实物接线图

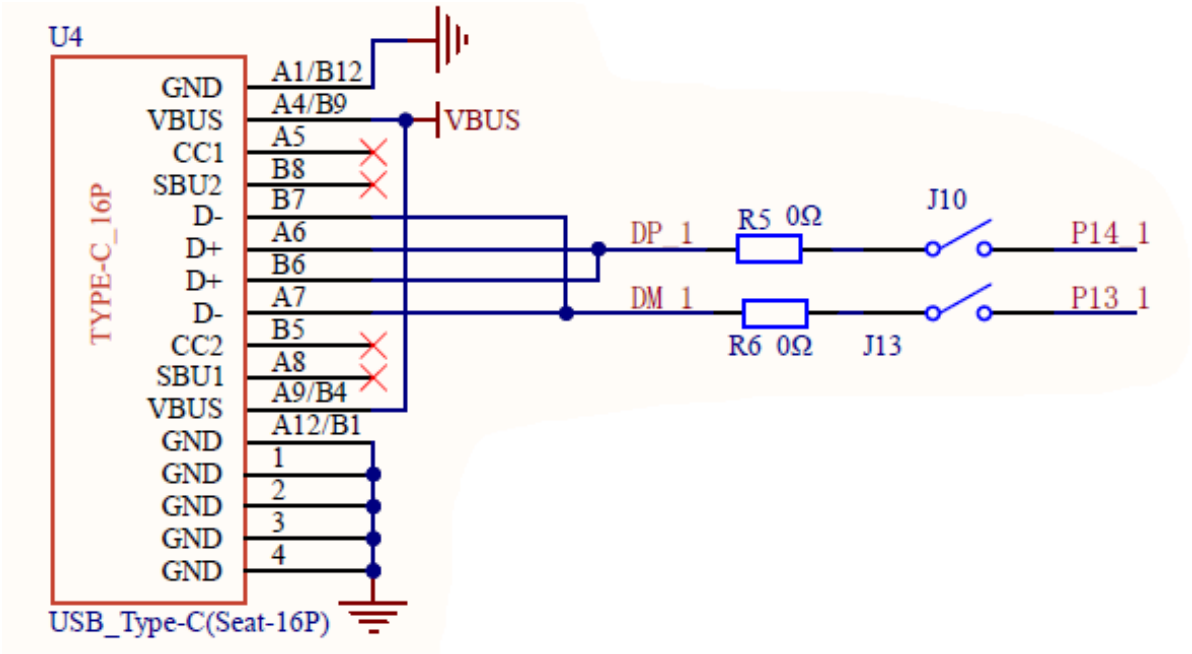


图 10: USB 模块外围电路原理图

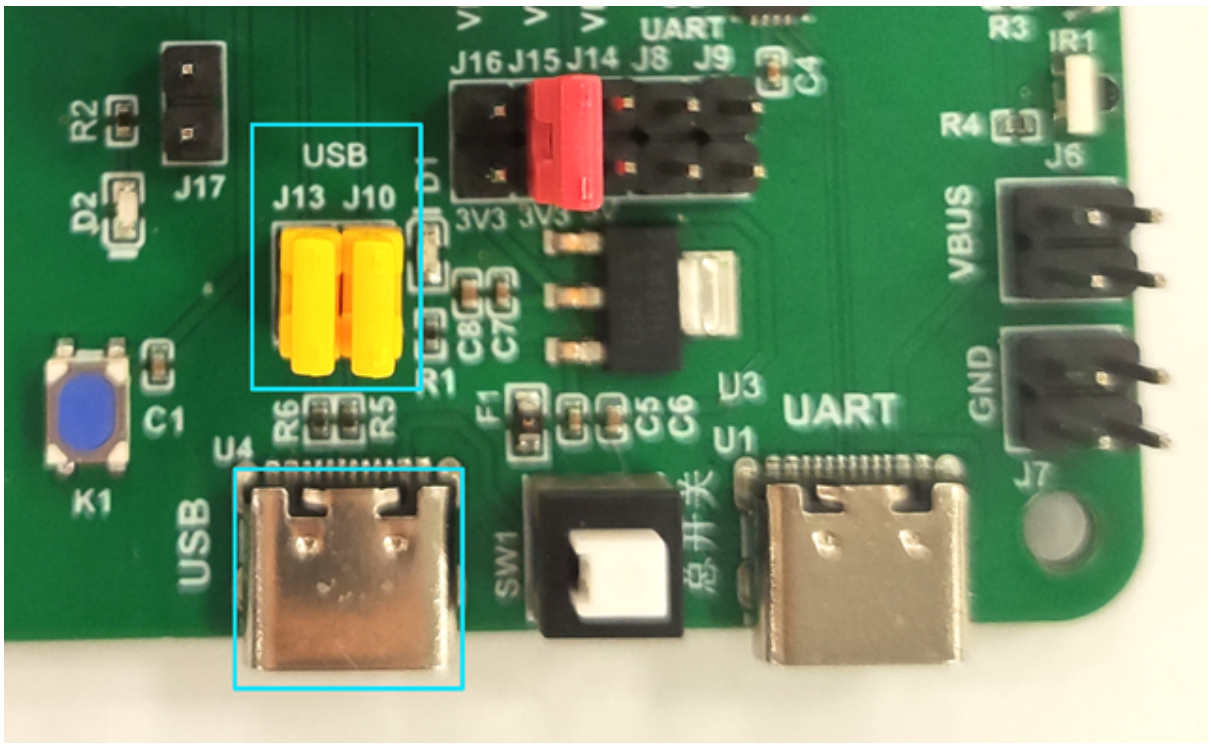


图 11: USB 模块外围电路实物图

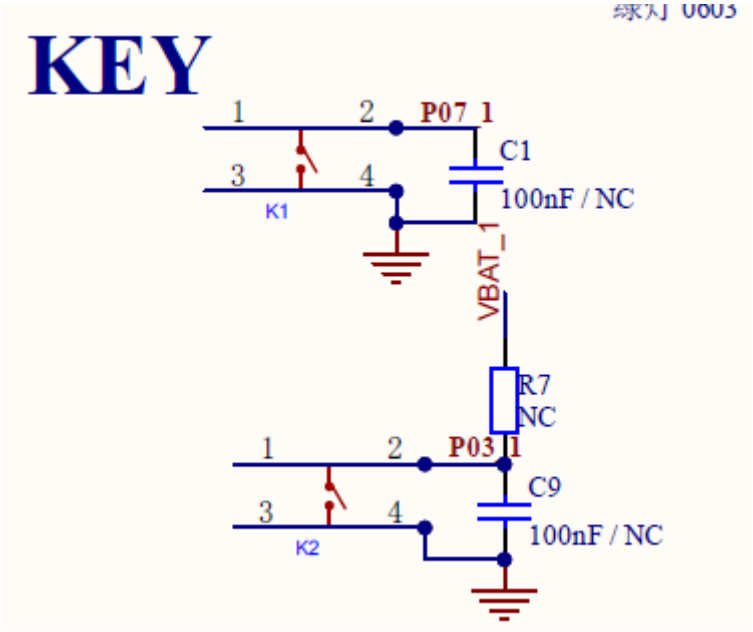


图 12: 独立按键原理图

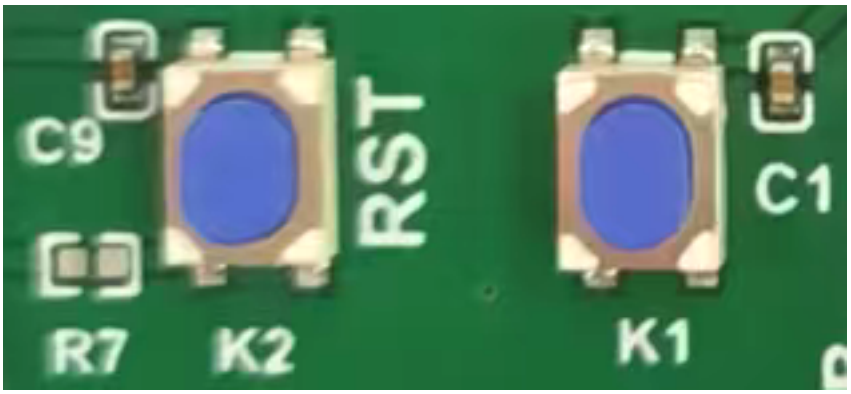


图 13: 独立按键实物图

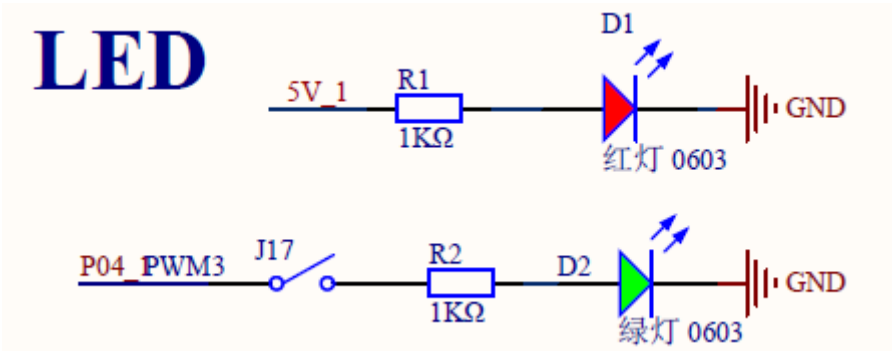


图 14: LED 灯模块原理图

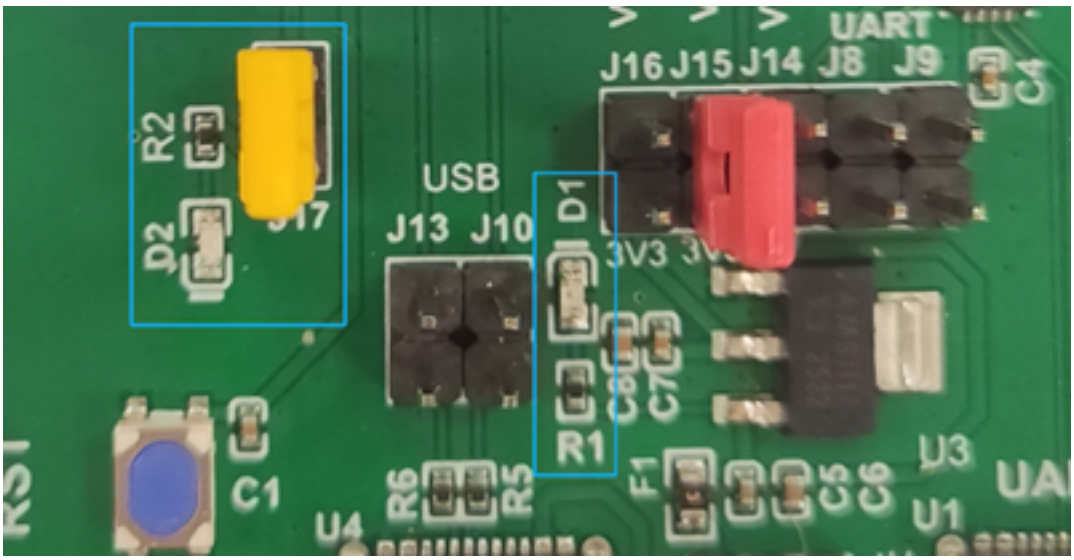


图 15: LED 灯模块实物图

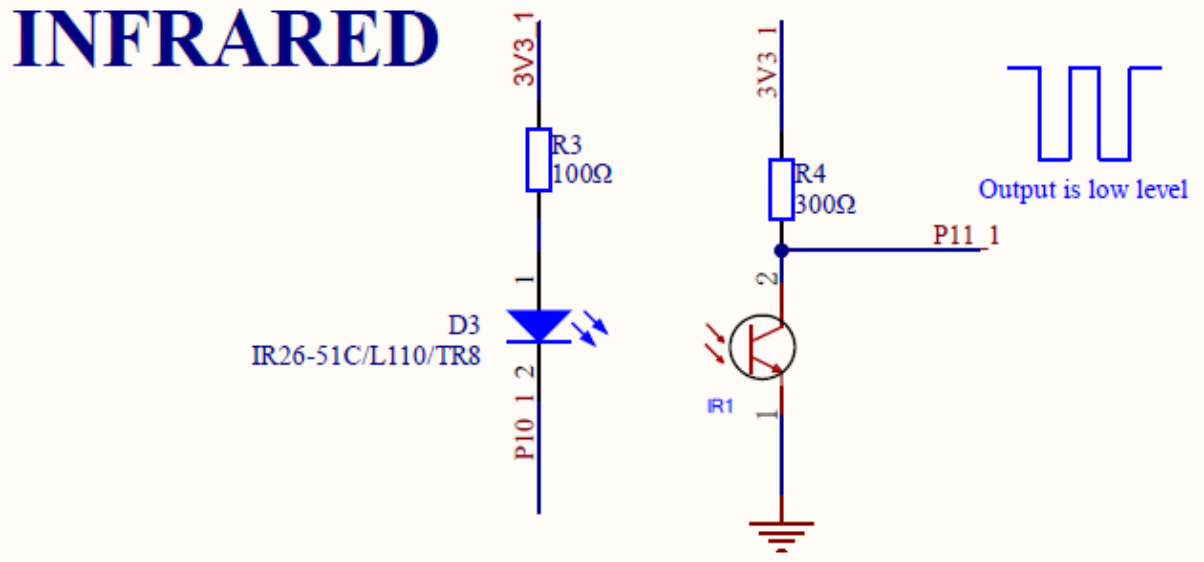


图 16: 红外模块原理图

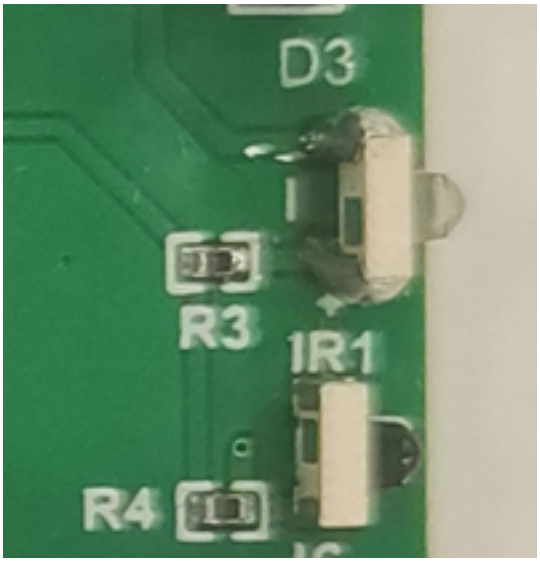


图 17: 红外模块实物接线图

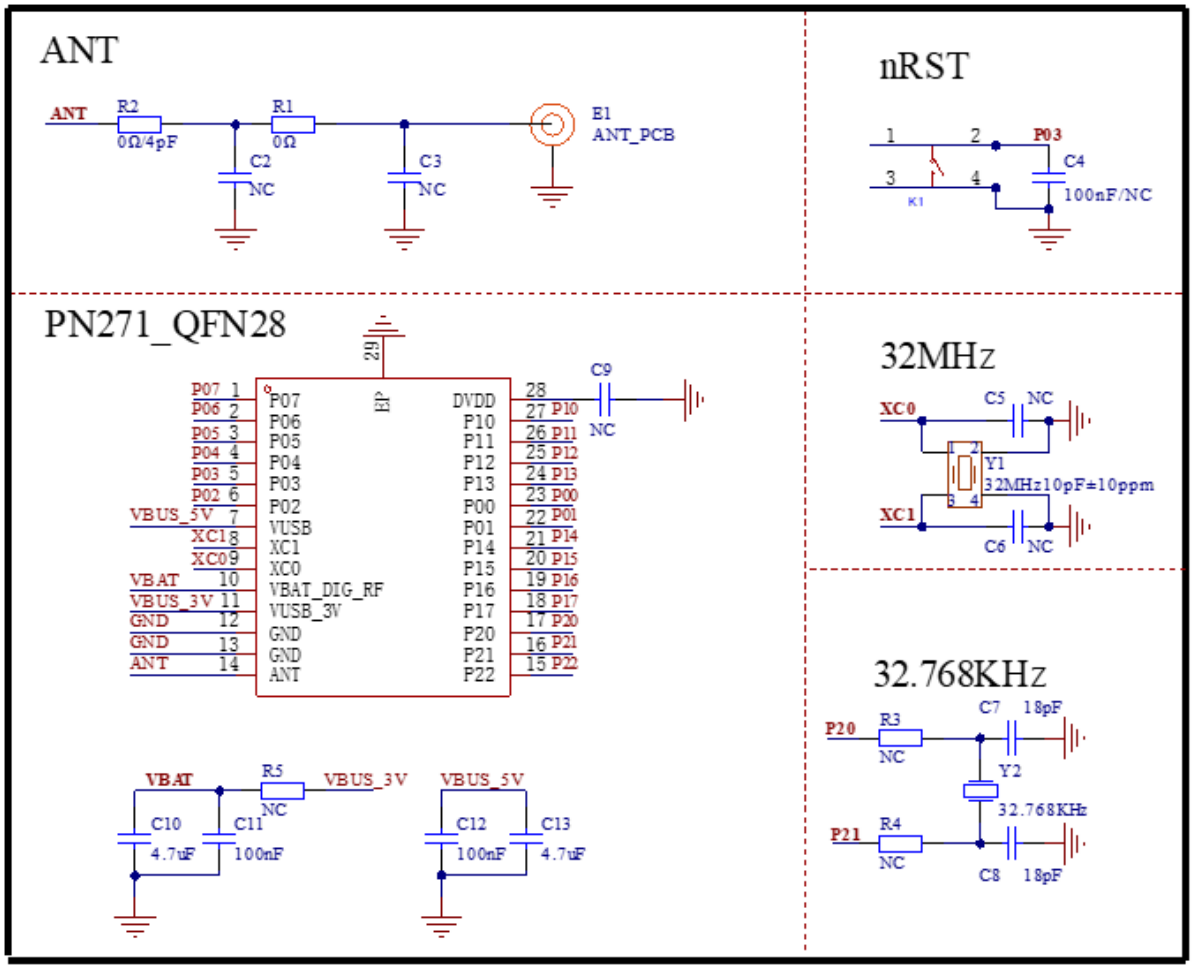


图 18: PAN2710U5GA（QFN28）最小系统参考原理图

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

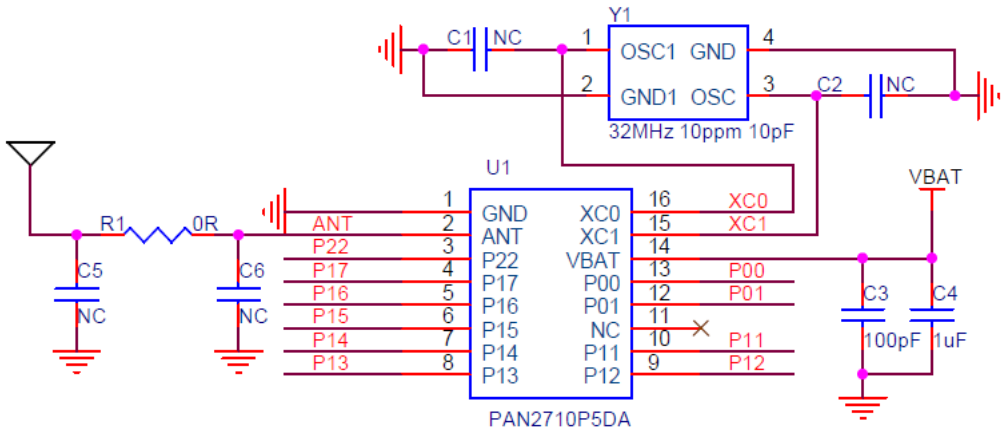


图 20: PAN2710P5DA（SOP16）最小系统参考原理图

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

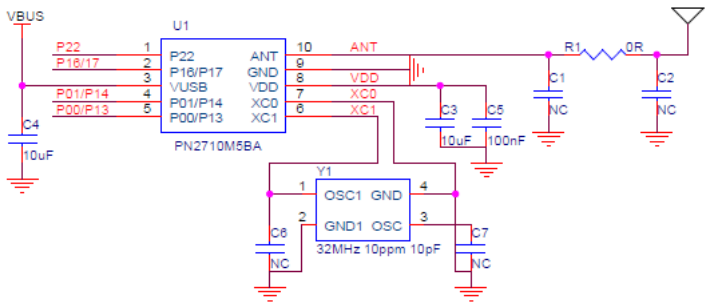


图 22: PAN2710M5BA（MSOP10）最小系统参考原理图

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小于 2mm。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。

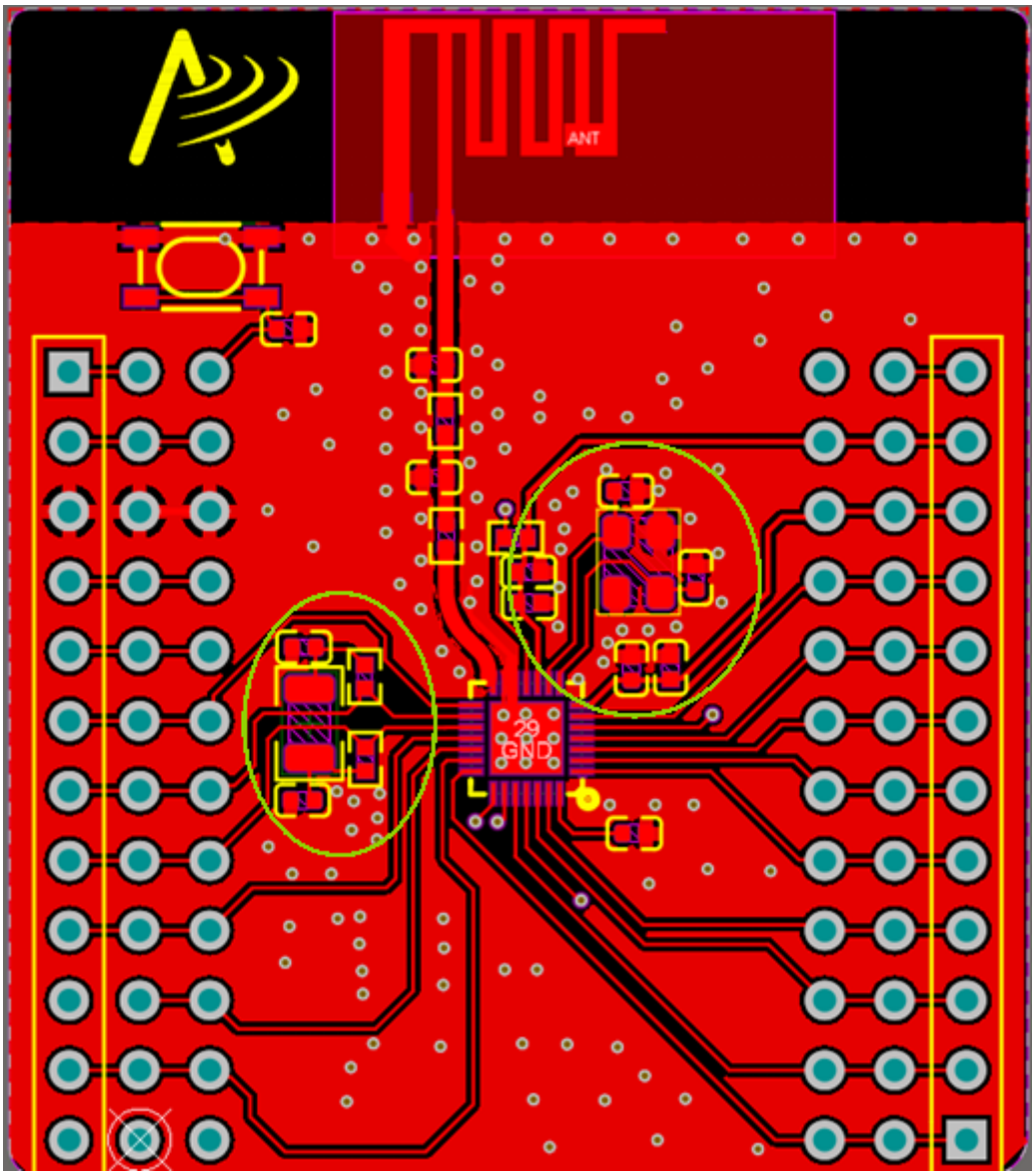


图 24: 晶体布局示意图

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：

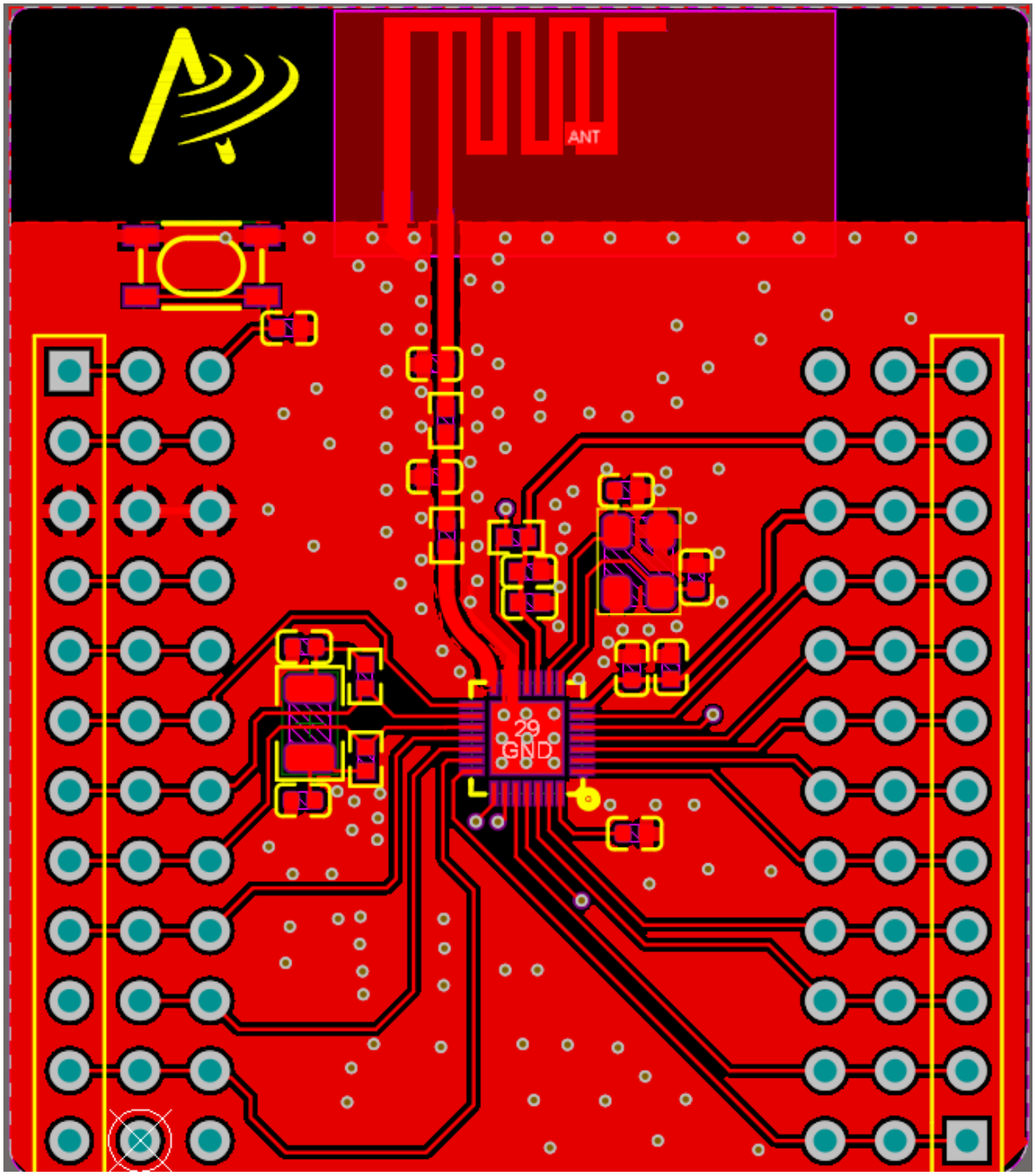


图 25: PAN2710U5GA（QFN28）EVB 核心板双面板 PIFA 天线 Top 层例

HDK 内容：

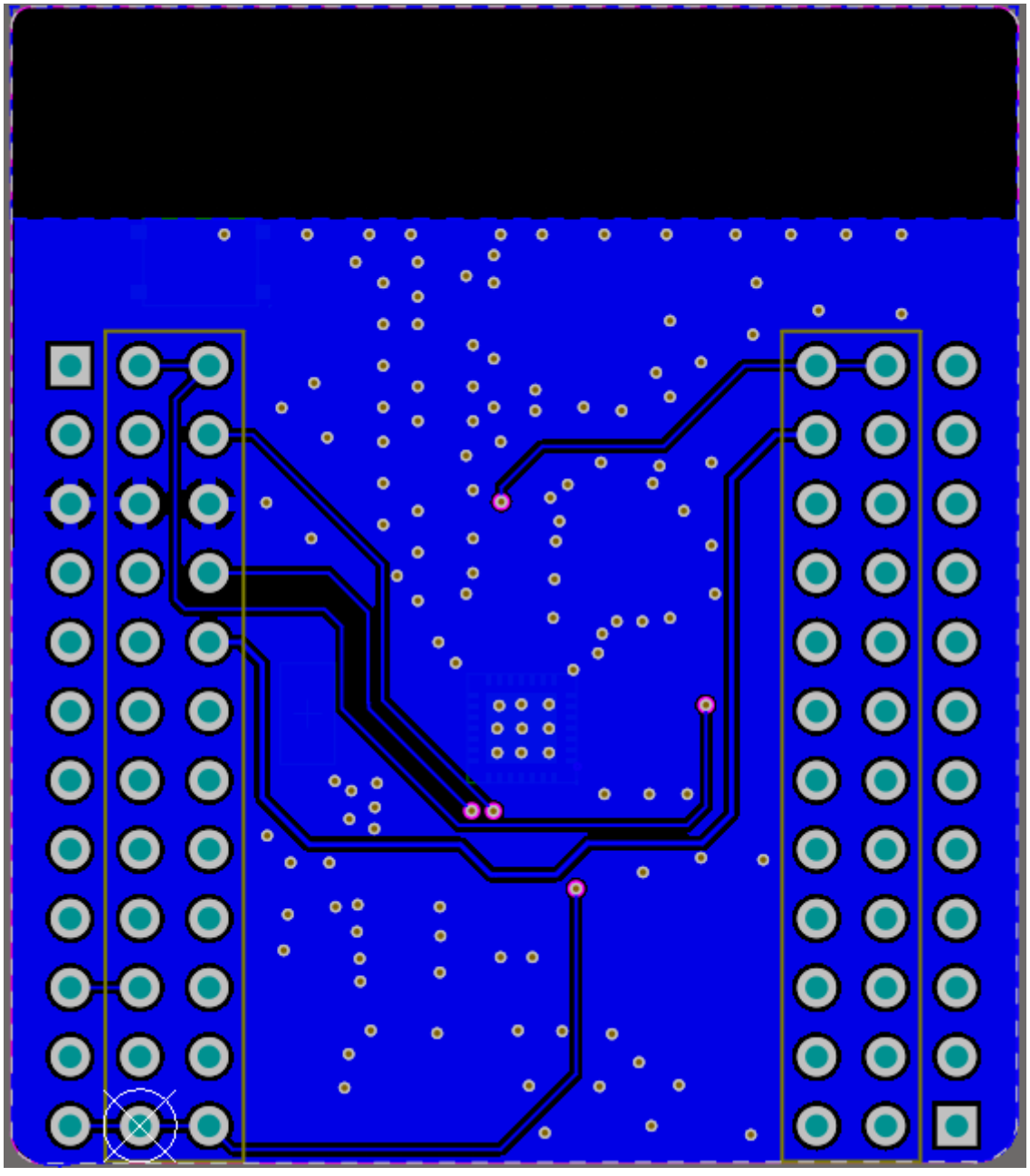


图 26: PAN2710U5GA（QFN28）EVB 核心板双面板 PIFA 天线 Bottom 层例

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 系统组件例程

3.1.1 PAN USB Mouse

1 功能概述

本例程演示 PanUsb 组件的 HID Mouse Device（鼠标设备）的基本功能。

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
  - 将 EVB 核心板 USB IO 连接到 EVB 底板 USB 接口
    - 使用跳线帽将 `J10` 排针对 (SoC_P14 & USB-DP) 短接起来
    - 使用跳线帽将 `J13` 排针对 (SoC_P13 & USB-DM) 短接起来
- 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\components\pan_usb_mouse

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，先不将 USB 连接至 PC，观察串口 Log 打印信息：

```
CPU @ 48000000Hz
Plug out
```

2. 将 USB 连接至 PC，可观察到串口 Log 打印 USB 通信日志：

```
Plug in
RST
USB first start
DEVICE
RST
USB first start
S_ADDR
addr : 3d
DEVICE
CFG
DEVQUAL
DEVICE
CFG
CFG
S_CFG
data
data
data
data
data
data
data
...
```

3. 同时可观察到 PC 鼠标指针在做自动画圈的动作。

5 RAM/Otp 资源使用情况

- Otp Size: 5.64kB
- RAM Size: 1.20kB

3.2 外设驱动例程

3.2.1 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。

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 ADC Sample Code.                        |
+------------------------------------------------------------------------+
|    Press key to start specific testcase:                               |
|                                                                        |
|    Input '1'    Testcase 1: Convert Test.                              |
|    Input '2'    Testcase 2: ADC Interrupt Test.                        |
|    Input '3'    Testcase 3: Temperature Test Case.                     |
|    Input '4'    Testcase 4: ADC Pwm One Adc Channel Test.              |
|    Input '5'    Testcase 5: ADC Hardware timer trig Test.              |
+------------------------------------------------------------------------+
```

2. 串口输入字符 '1'，Adc 转换功能，Adc 通道为 P20，ADC_CODE = 4096 / 3.3 * Input_Voltage（输入电压 1V），输出如下图，差异点来自 Adc Vbg 校准。

```
1214.88 1214.75 1214.56 1215.13 1214.88 1214.50 1214.75 1214.19 1214.63 1214.75
1214.31 1214.63 1214.44 1214.31 1214.75 1214.75 1214.63 1215.00 1214.69 1214.81
1214.63 1214.69 1214.81 1214.75 1215.06 1214.94 1214.94 1215.19 1214.69 1214.88
1215.19 1214.75 1215.44 1215.19 1214.94 1215.00 1214.88 1215.19 1215.19 1215.00
1215.38 1215.25 1215.06 1215.00 1215.00 1215.50 1215.19 1215.13 1215.44 1215.06
1215.38 1215.31 1215.31 1215.06 1215.50 1214.94 1215.38 1215.13 1215.13 1215.69
1215.13 1215.13 1215.31 1215.25 1215.44 1215.25 1215.25 1215.44 1215.00 1215.50
1215.38 1215.00 1215.13 1215.38 1215.31 1215.50 1215.13 1215.19 1215.19 1215.56
```

（下页继续）

(续上页)

```
1216.19 1215.06 1215.81 1215.88 1215.69 1216.06 1216.25 1216.00 1216.38 1215.81
1215.63 1216.25 1215.75 1215.63 1215.38 1215.94 1215.63 1216.19 1216.00 1215.44
```

3. 串口输入字符‘2’，Adc 中断功能，Adc 通道为 P20，ADC_CODE = 4096 / 3.3 * Input_Voltage（输入电压 1V），每次中断获取 4 个 Adc Code 值，输出如下图，转换 100 次打印 25 次 Half 中断。

```
Half
...
Half
1214 1215 1215 1217 1216 1214 1215 1218 1215 1214
1216 1216 1216 1216 1215 1218 1215 1214 1216 1219
1214 1216 1217 1217 1217 1217 1216 1217 1217 1214
1217 1218 1216 1216 1217 1217 1217 1215 1216 1217
1216 1216 1216 1217 1215 1217 1217 1219 1216 1216
1217 1217 1218 1215 1217 1217 1218 1215 1217 1218
1215 1216 1218 1218 1216 1216 1217 1218 1218 1215
1216 1218 1217 1214 1217 1218 1216 1216 1216 1218
1217 1218 1217 1218 1218 1217 1219 1219 1217 1214
1220 1217 1218 1217 1217 1218 1220 1219 1217 1219
```

4. 串口输入字符‘3’，Adc 温度检测功能，Adc 通道为内部温度检测通道，输出如下图，具体温度对应的公式及系数待实验室验证后确认。

```
1003 1001 1000 1002 1002 1002 1000 1002 1002 1002
1001 1002 1003 1002 1001 1002 1002 1001 1001 1002
1002 1002 1001 1001 1003 1002 1001 1002 1003 1002
1000 1001 1003 1001 1001 1002 1002 1002 1000 1002
1003 1002 1001 1002 1002 1001 1001 1002 1002 1002
1001 1002 1003 1002 1000 1001 1002 1002 1001 1002
1002 1001 1001 1001 1002 1001 1001 1002 1002 1002
1001 1002 1002 1002 1001 1002 1002 1002 1001 1001
1003 1002 1000 1002 1003 1002 1000 1001 1003 1001
1001 1002 1003 1002 1000 1001 1003 1001 1001 1002
```

5. 串口输入字符‘4’，Adc PWM 连续触发功能，Adc 通道为 P20，Adc 设置触发方式为 PWM channel0 下降沿触发，ADC_CODE = 4096 / 3.3 * Input_Voltage（输入电压 1V），输出与 3 相同。
6. 串口输入字符‘5’，Adc 硬件 timer 触发功能，Adc 通道为 P20，Adc 设置硬件 timer 触发方式，触发间隔设置为 16K Hz，ADC_CODE = 4096 / 3.3 * Input_Voltage（输入电压 1V），输出与 3 相同。

5 RAM/Otp 资源使用情况

- Otp Size: 10.89kB
- RAM Size: 0.81kB

3.2.2 CLKTRIM

1 功能概述

本例程演示演示 ClkTrim 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\clktrim`

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
+-------------------------------------------------------------------------------+
|                      PAN271 Clk source to be tested.                          |
+-------------------------------------------------------------------------------+
|    Press key to select clk source to be tested:                               |
|    Input '0'    Clk source select internal clk.                               |
|    Input '1'    Clk source select external clk.                               |
+-------------------------------------------------------------------------------|
```

2. 串口输入字符 ‘0/1’，选择待测量时钟为内部 RCL 或者外部时钟源，此处外部时钟为 PWM 100K Hz，输入 PIN 为 P20。

```
+-------------------------------------------------------------------------------+
|                      PAN271 Clk measure sample number select.                 |
+-------------------------------------------------------------------------------+
|    Press key to select sample number:                                         |
|    Input '0'    Sample number select 1.                                       |
|    Input '1'    Sample number select 2.                                       |
|    Input '2'    Sample number select 4.                                       |
|    Input '3'    Sample number select 8.                                       |
|    Input '4'    Sample number select 16.                                      |
|    Input '5'    Sample number select 32.                                      |
|    Input '6'    Sample number select 64.                                      |
|    Input '7'    Sample number select 128.                                     |
+-------------------------------------------------------------------------------+
```

3. 串口输入字符 ‘0~7’，选择采样次数，最终测量的 cnt 值为采样次数的 cnt 平均值，这里输入 1。

```
+-------------------------------------------------------------------------------+
|                      PAN271 Clk measure edge mode select.                     |
+-------------------------------------------------------------------------------+
|    Press key to select edge mode:                                             |
|    Input '0'    Edge mode select rising.                                      |
|    Input '1'    Edge mode select failing.                                     |
|    Input '2'    Edge mode select high voltage(only one time sample).          |
|    Input '3'    Edge mode select low voltage(only one time sample).           |
+-------------------------------------------------------------------------------+
```

4. 串口输入字符 '0~3'，选择开始采样触发沿，此处注意高电平与低电平模式仅采样一次，高电平模式先上升沿后下降沿表示有效开始采样，低电平模式先下降沿后上升沿表示有效开始采样，此处以下降沿为例（输入 0）。

```
内部 RCL:
measure done
measure clk is 32M clk, real freq is :33143.45
外部 PWM
measure done
measure clk is 32M clk, real freq is :100000.00
```

5 RAM/Otp 资源使用情况

- Otp Size: 8.57kB
- RAM Size: 0.52kB

3.2.3 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。

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
  - 使用杜邦线将验证 IO P02 连接 P12，P17 连接 P20

- 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，芯片会通过串口打印初始化 Log:

```
CPU @ 48000000Hz
+------------------------------------------------------------------------------+
|                      PAN271X GPIO Sample Code.                               |
+------------------------------------------------------------------------------+
|    Press key to start specific testcase:                                     |
|    Input '1'    Testcase 1: Push-pull Mode Test.                             |
|    Input '2'    Testcase 2: Open-drain Mode Test.                            |
|    Input '3'    Testcase 3: Quasi-bidirectional Mode Test.                   |
|    Input '4'    Testcase 4: Debounce Test.                                   |
|    Input '5'    Testcase 5: Interrupt Test.                                  |
+------------------------------------------------------------------------------+
|    Board Pin Connection Scheme:                                              |
|      [Group1]                [Group2]                                        |
|      P02       <---->        P12                                             |
|      P17       <---->        P20                                             |
+------------------------------------------------------------------------------+
```

2. 串口输入字符‘1’，Gpio Push-Pull 功能，每组 IO 先设输出后输入，确保每个 IO 功能正常，输出结果。

```
GPIO Test OK, Success case: 0, Success Pin: P02
GPIO Test OK, Success case: 0, Success Pin: P12
GPIO Test OK, Success case: 0, Success Pin: P17
GPIO Test OK, Success case: 0, Success Pin: P20
```

3. 串口输入字符‘2’，Gpio Open-Drain 功能，每组 IO 先设输出后输入，确保每个 IO 功能正常，输出结果（同第二点）。
4. 串口输入字符‘3’，Gpio Quasi-Bidirectional 功能，每组 IO 先设输出后输入，确保每个 IO 功能正常，输出结果（同第二点）。
5. 串口输入字符‘4’，Gpio Debounce 功能，Gpio 设置上升沿中断，debouce 时间为 64 个 HCLK，软件模拟抖动波形，**中断触发条件为电平从低至高并维持高** 128 **个** HCLK **（稳定触发），从低至高并维持高** 64~128 HCLK **区间有可能触发中断，也有可能不触发，这取决于边沿变化与** Debouce Sample CLK **的相位，**Debouce Sample CLK **刚过发生边沿变化，那么判定高需要约** 2 **个** Debouce Sample CLK**，边沿变化后** Debouce Sample CLK **马上采集，那么判定高只需要约** 1 **个** Debouce Sample CLK。

```
Debounce Period - 64 Cycles HCLK
P0_2 INT occurred, type: GPIO_INT_RISING.
Debounce Period - 128 Cycles HCLK
GPIO Test OK, Success case: 3, Success Pin: P02
...
```

6. 串口输入字符‘5’，Gpio Interrupt 功能，软件模拟电平变化，IO 的所有中断类型均能正常触发。

```
P0_2 INT occurred, type: GPIO_INT_RISING.
P0_2 INT occurred, type: GPIO_INT_FALLING.
P0_2 INT occurred, type: GPIO_INT_BOTH_EDGE.
P0_2 INT occurred, type: GPIO_INT_BOTH_EDGE.
P0_2 INT occurred, type: GPIO_INT_HIGH.
P0_2 INT occurred, type: GPIO_INT_LOW.
GPIO Test OK, Success case: 4, Success Pin: P02
...
```

5 RAM/Otp 资源使用情况

- Otp Size: 6.71kB
- RAM Size: 0.77kB

3.2.4 I2C

1 功能概述

本例程演示演示 I2C 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
  - I2C_SCL(P14) 接从机 SCL，I2C_SDA(P13) 接从机 SDA

3 编译和烧录

例程位置：`<PAN271x-DK>\01_SDK\samples\drivers\i2c`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

PAN271x IIC **仅支持** Master **模式**

1. 烧录完成后，芯片会通过串口打印初始化 Log：

```
CPU @ 48000000Hz
+------------------------------------------------------------------------------+
|                       PAN271x I2C Sample Code.                               |
+------------------------------------------------------------------------------+
|    Press key to start specific testcase:                                     |
|    Input '1'    Testcase 1 : Master  Send Data.                              |
|    Input '2'    Testcase 2 : Master  Receive    Data.                        |
+------------------------------------------------------------------------------+
```

2. 串口输入字符‘1’，I2C Master 发送功能，PAN271x 发送数据 0x00~0x1f，从机为 PAN107x，输出接收数据。

```
Master 输出:
I2C_MasterSendDataCase
Slave 输出:
I2C_SlaveReceiveDataCase2
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D␣
↪1E 1F
```

3. 串口输入字符‘2’，I2C Master 接收功能，从机为 PAN107x，从机发送数据 0x00~0x1f，PAN271x 接收数据并输出结果。

```
Master 输出:
I2C_MasterReceiveDataCase
RCV DATA: 0 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
Slave 输出:
I2C_SlaveSendDataCase3
```

5 RAM/Otp 资源使用情况

- Otp Size: 4.74kB
- RAM Size: 0.77kB

3.2.5 KSCAN

1 功能概述

本例程演示演示 KeyScan 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
- 连接 IO 与按键（4x4）

  PAN271x 按键板

  VBAT VDD

  GND GND

  P02 ROW1

  P12 ROW2

  P14 ROW3

  P20 COL1

3 编译和烧录

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

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 KeyScan Sample Code.                      |
+------------------------------------------------------------------------+
|    Press key to start specific testcase:                               |
|    Input '1'    Testcase 1: key int Test Case.                         |
|    Input '2'    Testcase 2: key polling Test Case.                     |
+------------------------------------------------------------------------+
```

2. 串口输入字符 '1'，KeyScan 中断检测功能，示例演示一个 3 行一列的按键，按键按下与抬起均触发中断，按键按下输出当前按键 id，按键抬起无 id 输出。

```
IRQ
key 0_0 pressed
IRQ
key pressed
IRQ
key 1_0 pressed
IRQ
key pressed
IRQ
key 2_0 pressed
IRQ
key pressed
```

3. 串口输入字符 '2'，KeyScan 查询检测功能，示例演示一个 3 行一列的按键，PAN271x 循环查询按键状态标志，按键按下输出当前按键 id，按键抬起无 id 输出。

```
key 0_0 pressed
key pressed
key 1_0 pressed
key pressed
key 2_0 pressed
key pressed
```

5 Ram/Otp 资源使用情况

- Otp Size: 4.24kB
- RAM Size: 0.78kB

3.2.6 OTP

1 功能概述

本例程演示演示 OTP 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\otp`

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 @ 32000000Hz
Input tr trim u want, value will change to hex
```

2. 串口输入 tr trim 值，tr trim 为一个 16bit 的值，默认为 0，此值会在出厂时写入 OTP Row 区，上电后由硬件 load 值。

```
tr trim is 0

Input ptm u want
MAIN_READ                       (0X00)
MAIN_READ2                      (0X10)
MAIN_WEAK_READ                  (0X01)
MAIN_WEAK_READ2                 (0X11)
MAIN_MARGAIN_READ               (0X02)
MAIN_MARGAIN_READ2              (0X12)
-----------------------------------------
ROW_READ                    (0X08)
ROW_READ2                   (0X18)
ROW_WEAK_READ               (0X09)
ROW_WEAK_READ2              (0X19)
ROW_MARGAIN_READ            (0X0A)
ROW_MARGAIN_READ2           (0X1A)
-----------------------------------------
COL_READ                    (0X04)
COL_READ2                   (0X14)
COL_WEAK_READ               (0X05)
COL_WEAK_READ2              (0X15)
COL_MARGAIN_READ            (0X06)
COL_MARGAIN_READ2           (0X16)
```

3. 串口输入 read 命令，上述命令分别对应 OTP main 区、row 区及 col 区的读命令，此处以 main 区为示例。

```
ptm is 0

Input ecc_en u want
```

4. 串口输入 ecc 使能位，注意读 col 区 ecc 必须关闭。

```
ecc is 0

Input adr u want
adr is 0
0x18 0x09 0x00 0x20 0xF1 0x00 0x00 0x00 0xF9 0x00 0x00 0x00 0xD5 0x00 0x00 0x00 0x00 0x00␣
↪0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Input tr trim u want, value will change to hex
```

5. 串口输入读地址，读长度默认 32Byte，对比编译下载的 hex 文件，确认读正确。

5 RAM/Otp 资源使用情况

- Otp Size: 4.89kB
- RAM Size: 0.77kB

3.2.7 PWM

1 功能概述

本例程演示演示 PWM 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）
  - 逻辑分析仪
- 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
  - P12(PWM_CH4)/P13(PWM_CH3)/P14(PWM_CH2)/17(PWM_CH1) 连接逻辑分析仪

3 编译和烧录

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

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
pwm clk source select
a:  clk source select apb
b:  clk source select rcl/xtl
```

2. 串口输入 ‘a/b’。选择 pwm 时钟源，apb 时钟为 48M Hz，rcl 时钟为 32k Hz，xtl 时钟为 32768 Hz，其中 rcl 需校准，apb 时钟源的 pwm 输出频率是 rcl/xtl 的 1000 倍，演示的 pwm 时钟源选择为 apb clk。

```
+---------------------------------------------------------------------------+
|                      PAN107x PWM Sample Code.                             |
+---------------------------------------------------------------------------+
|    Press key to start specific testcase:                                  |
|                                                                           |
|    Input '1'    Testcase 1: PWM Operating Modes.                          |
```

(下页继续)

(续上页)

```
|    Input '2'    Testcase 2: Polarity Control.                                    |
+---------------------------------------------------------------------------------+
```

3. 串口输入字符‘1’，Pwm 输出模式演示

```
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|    Input 'A'    Independent Mode.                               |
|    Input 'B'    Complementary Mode.                             |
|    Input 'C'    Complementary Mode with Dead-Time Insertion.    |
|    Input 'D'    Synchronized Mode.                              |
|    Input 'E'    Grouping Mode.                                  |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
A 独立模式--pwm ch2 输出 100KHz, ch3 输出 200KHz。
B 互补模式--pwm ch2 输出 500KHz, ch3 输出波形完全相反 500KHz。
C 带死区的互补模式--pwm ch2 输出 500KHz, ch3 输出波形相反但相位有差异的 500KHz。
D 同步模式--pwm ch2 和 pwm ch3 输出完全相同的 500KHz。
E 组模式--pwm ch2/ch4 输出完全相同的 500KHz, ch1/ch3 输出完全相同的 400KHz。
```

4. 串口输入字符‘2’，Pwm 极性控制，pwm ch2/ch3 均设置为输出 500KHz，ch3 极性翻转使能打开。

   ch2/ch3 先设置为独立模式，波形输出应为完全互补的波形

   ch2/ch3 后设置为带死区的互补模式，波形输出应为有相位差同向波形。

   同步模式极性翻转控制不生效

```
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Output Waveform with Polarity Control.          |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
Independent TGT_PWM wave start...
TGT_PWM wave stopped.
Complementary TGT_PWM wave with Dead-Time start...
TGT_PWM wave stopped.
```

5 RAM/Otp 资源使用情况

- Otp Size: 8.69kB
- RAM Size: 0.77kB

3.2.8 SPI

1 功能概述

本例程演示演示 SPI 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 2 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
  - 两个 EVB 底板上 P07(MOSI)/P12(MISO)/13(CS)/14(CLK) 两两相连，**注意 EVB 底板 C1 电容断开，否则 MOSI 无法正常使用**

3 编译和烧录

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

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：

```
+-----------------------------------------------------------------------------+
|                       PAN271x SPI Sample Code.                              |
+-----------------------------------------------------------------------------+
|    Press key to start specific testcase:                                    |
|                                                                             |
|    Input '1'    Testcase 1: Interrupt Test.                                 |
|    Input '2'    Testcase 2: Simple Transmission Demo.                       |
+-----------------------------------------------------------------------------+
```

2. 串口输入字符‘1’，spi 中断模式传输功能

   EVB2 端输入‘B’进入接收模式，等待数据接收

   EVB1 输入‘A’进入发送，发送数据 0x00~0x9f

```
EVB2 输出：
Data received by Target SPI with Rx INT enabled:
 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f
 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0x1f
 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e 0x2f
 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3a 0x3b 0x3c 0x3d 0x3e 0x3f
 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4a 0x4b 0x4c 0x4d 0x4e 0x4f
 0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5a 0x5b 0x5c 0x5d 0x5e 0x5f
 0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6a 0x6b 0x6c 0x6d 0x6e 0x6f
 0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7a 0x7b 0x7c 0x7d 0x7e 0x7f
 0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8a 0x8b 0x8c 0x8d 0x8e 0x8f
 0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9a 0x9b 0x9c 0x9d 0x9e 0x9f
EVB1 输出：
Send data by Target SPI with Tx INT enabled:
发送 160Byte 数据 0x00~0x9f
```

3. 串口输入字符‘1’，spi 中断模式 overrun 功能，输入‘C’，spi 选择为 master 模式，并打开空中断及 rx overrun 中断，tx 一直发送但 rx 并没有接收数据，触发 overrun 中断。

```
SPI Rx Overrun!
...
SPI Rx Overrun!
```

4. 串口输入字符‘2’，Spi 查询模式传输功能。

   EVB2 端输入‘B’进入从机接收模式，等待数据接收

   EVB1 输入‘A’进入主机发送模式，发送数据”Master Send, Slave Recv.”

```
EVB2 输出:
Recv data, act as Slave...
Data rcvd: "Master Send, Slave Recv."
EVB1 输出:
Send data, act as Master.
```

5. 串口输入字符‘2’，Spi 查询模式传输功能。

   EVB2 端输入‘C’进入从机发送模式，等待主机发送时钟

   EVB1 输入‘D’进入主机接收模式，发送时钟并接收数据

```
EVB2 输出:
Send data, act as Slave....
EVB1 输出:
Recv data, act as Master.
Data rcvd: "Slave Send, Master Recv."
```

5 RAM/Otp 资源使用情况

- Otp Size: 8.33kB
- RAM Size: 1.31kB

3.2.9 TIMER

1 功能概述

本例程演示演示 Timer 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
  - 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：

```
+------------------------------------------------------------------+
|    Press key to test specific function:                          |
|    Input 'A'    Periodic Mode.                                   |
|    Input 'B'    Continuous-Counting Mode.                        |
|    Input 'C'    Capture function.                                |
|    Press ESC key to back to the top level case list.             |
+------------------------------------------------------------------+
```

2. 串口输入字符‘A’，timer 周期计数功能，软件设置 timeout 时间为 333ms，每隔 333ms 触发一次中断，输出结果查看时间差符合预期。

```
Timer0, Compare Value:16000000
Start Timer...
[17:34:26.237] 收 ←CNT
[17:34:26.571] 收 ←CNT
[17:34:26.904] 收 ←CNT
[17:34:27.237] 收 ←CNT
[17:34:27.570] 收 ←CNT
```

3. 串口输入字符‘B’，timer 连续计数功能，软件设置比较值为 300000，到达后触发中断并重新设置比较值为 600000，最后设置为 1200000，每次到达后获取当前的计数器值。

```
Start Timer...
CNT
Timer0, Compare Value:300000
CNT
Timer0, Compare Value:600000
CNT
Timer0, Compare Value:1200000
```

4. 串口输入字符‘C’，timer 捕获功能，timer 内部设计的捕获 usb sof 的间隔，sof 理论为 1ms 来一次，系统时钟为 48MHz，理论捕获输出 cnt 为 48000。

```
+----------------------------------------------------------+
|    Press key to test capture edge:                       |
|    Input '1'    Rising edge.                             |
|    Input '2'    Falling edge.                            |
|    Input '3'    Both edge.                               |
+--------------------------------------------- -----+
48000 48001 47999 48000 48000 48002 48000 48000 48000
```

5 RAM/Otp 资源使用情况

- Otp Size: 5.02kB
- RAM Size: 0.85kB

3.2.10 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

3.2.11 WDT

1 功能概述

本例程演示演示 WatchDog 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）
  - 逻辑分析仪
- 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

1. 烧录完成后，芯片会通过串口打印初始化 Log：

```
CPU @ 48000000Hz
NOTE: Wdt clock source is RCL or XTL!!!
+------------------------------------------------------------------------+
|                      PAN271x WDT Sample Code.                          |
+------------------------------------------------------------------------+
|    Press key to start specific testcase:                               |
|    Input '1'    Testcase 1: Interrupt Mode.                            |
|    Input '2'    Testcase 2: Reset Mode.                                |
+------------------------------------------------------------------------+
```

2. 串口输入字符‘1’，wdt timeout 中断功能，软件设置 timeout 时间为 4096 个 32K 时钟，中断触发串口输出 “WDT INT”，同时通过逻辑分析仪抓取 debug io p20，观测 timeout 时间是否符合预期，注意 rcl 时钟需要校准。

```
Start WDT Counting (WDT_CLK = 32KHz, TimeoutCnt = 4096)...
[18:47:24.643] 收 ←WDT INT
[18:47:24.779] 收 ←WDT INT
[18:47:24.914] 收 ←WDT INT
[18:47:25.049] 收 ←WDT INT
[18:47:25.184] 收 ←WDT INT
[18:47:25.319] 收 ←WDT INT
```

3. 串口输入字符‘2’，wdt reset 功能，软件设置 timeout 时间为 4096 个 32K 时钟，同时设置 reset delay 时间，观测是否发生复位，同时通过逻辑分析仪抓取 debug io p20，观测 reset 时间是否符合预期。

```
WDT Reset Delay Count = 17
Start WDT Counting (WDT_CLK = 32KHz, TimeoutCnt = 4096)...
WDT INT
CPU @ 48000000Hz
NOTE: Wdt clock source is RCL or XTL!!!
+------------------------------------------------------------------------+
|                      PAN271x WDT Sample Code.                          |
+------------------------------------------------------------------------+
|    Press key to start specific testcase:                               |
|    Input '1'    Testcase 1: Interrupt Mode.                            |
|    Input '2'    Testcase 2: Reset Mode.                                |
+------------------------------------------------------------------------+
```

5 RAM/Otp 资源使用情况

- Otp Size: 6.33kB
- RAM Size: 0.77kB

3.3 其他例程

3.3.1 Blinky

1 功能概述

本例程演示通过 GPIO 推挽输出控制 EVB LED 灯闪烁的方法。

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
  - 使用杜邦线或跳线帽将 EVB 底板 `J17` 排针对短接起来，以使能 EVB 底板上的 LED 灯
- 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\miscellaneous\blinky`

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，观察串口 Log 打印，可以看到系统成功初始化并闪灯的 Log：

```
CPU @ 32000000Hz
LED on
LED off
LED on
LED off
...
```

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

5 RAM/Otp 资源使用情况

- Otp Size: 2.30kB
- RAM Size: 0.77kB

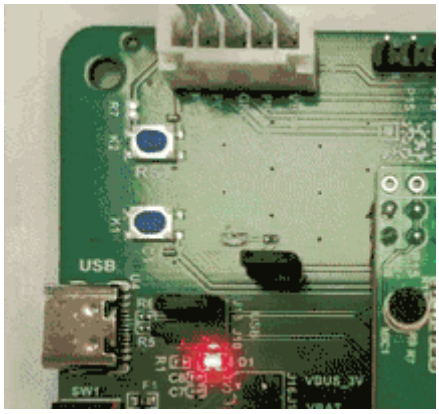


图 1: Blinky LED on EVB

3.3.2 CoreMark

1 功能概述

本例程演示 CoreMark 基准测试。

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
- 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\miscellaneous\coremark`

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，观察串口 Log 打印，可以看到系统成功初始化并启动 CoreMark 测试的 Log：

```
CPU @ 32000000Hz
CoreMark start...
```

2. 等待一段时间后（约 1 分钟），可以观察到 CoreMark 基准测试结果：

```
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 51370
Total time (secs): 51
Iterations/Sec   : 19
Iterations       : 1000
Compiler version : GCC4.7 (EDG gcc mode)
Compiler flags   : -o3
Memory location  : STATIC
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal      : 0xd340
Correct operation validated. See readme.txt for run and reporting rules.
```

5 RAM/Otp 资源使用情况

- Otp Size: 7.50kB
- RAM Size: 2.84kB

3.3.3 Debug Protect

1 功能概述

本例程演示芯片 Debug Protect 调试保护机制。

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
- 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\miscellaneous\debug_protect`
双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

此例程仅可在 OTP 版本的芯片上执行，暂不提供演示说明。

3.3.4 Low Power

1 功能概述

本例程演示芯片 Low Power 低功耗模式。

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\miscellaneous\low_power`

双击 Keil Project 文件打开工程进行编译烧录。

4 低功耗模式说明

模式名称	唤醒	时钟	电源
STANDBY	所有 GPIO，BOD，LVR（可选，需开启慢时钟）PIN RESET	全部关闭	LPLDOH：PMU，GPIO，WDT，BOD，LVR
STANDBY	所有 GPIO（边沿去抖）SLPTMR，WDT，BOD，LVR（可选）PIN RESET	慢时钟其他全部关闭	LPLDOH：PMU，GPIO，WDT，BOD,LVR LPLDOL/LPLDOH：PHY REG，MAC REG，SRAM，RCC REG，SYSTEM REG
DEEP-SLEEP	所有 GPIO，SLPTMR，WDT，BOD，LVR（可选）PIN RESET	慢时钟其他全部关闭	LPLDOH：PMU，GPIO，WDT，BOD,LVR LPLDOL/LPLDOH：ALL OTHER DIG MODULES
SLEEP	所有外设中断 BOD，LVR（可选）PIN RESET	慢时钟 CPU_CLK 关闭 RCH/XTH/DPLL 根据软件配置选择打开	HP_LDO: ALL DIG MODULES

5 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：

```
sram addr 0x20000B00,value=15d34c13
+------------------------------------------------------------------------+
|                      PAN271x LowPower Sample Code.                     |
+------------------------------------------------------------------------+
|    press key to start test                                             |
|    Input'1'   LP_SleepModeTest();                                      |
|    Input'2'   LP_DeepSleepPwmOutTest();                                |
|    Input'3'   LP_ContinuesSleepWakeByGpioClk32k();                     |
|    Input'4'   LP_StandbyModeTest();                                    |
|    Input'5'   LP_ContinuesStdbyWakeByGpio32k();                        |
+------------------------------------------------------------------------+
```

2. 串口输入字符 '1'，sleep 休眠唤醒功能，此模式下仅关闭 CPU 时钟，任何中断均可以唤醒，例程演示 32K 定时器、gpio 及 wdt 唤醒，唤醒后输出中断类型，程序继续执行。

```
Choose wake up mode:
 'A'  for sleep timer wake up
 'B'  for gpio wake up
 'C'  for wdt wake up
输入 'A':
wake up by rcl 32k
slptmr int
sleep mode run continue
输入 'B':
wake up by gpio P12
p1 interrupt
sleep mode run continue
输入 'C':
```

(下页继续)

(续上页)

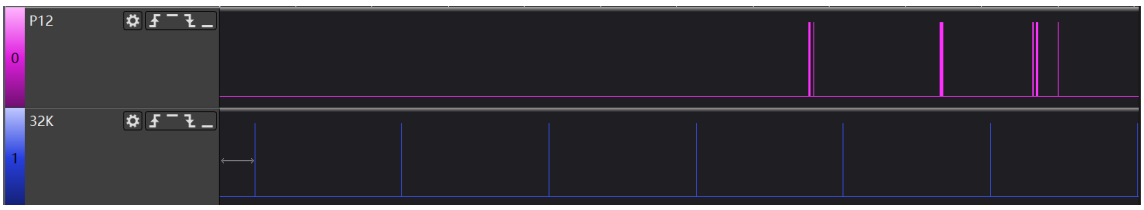
```
wake up by watchdog
WDT interrupt
sleep mode run continue
```

3. 串口输入字符‘2’，deepsleep 休眠唤醒及低功耗下 pwm 输出演示，唤醒方式为 32K timer，pwm 通道为 P13，唤醒触发 slptmr 及 lp 中断并输出中断 log，同时逻辑分析仪抓取 P13 的波形（正常输出 pwm 波形），程序继续执行。

```
wake up by rcl 32k(wait for wake up)
slptmr int
Deepsleep
deep sleep mode run continue
```

4. 串口输入字符‘3’，deepsleep 多唤醒源休眠唤醒演示，唤醒方式支持 32k timer 及 gpio P12，32k timer 定时 2s 唤醒，gpio 随机唤醒，循环进入休眠，通过逻辑分析仪抓取唤醒波形。

```
Input lpldol trim(输入 LPLDOL 电压档位，范围 0~15)
4
lpldol = 4
Input lpldoh trim(输入 LPLDOH 电压档位，范围 0~15)
4
lpldoh = 4

wake up by gpio P12 or 32K(进入休眠)
```



5. 串口输入字符‘4’，standby m0/m1 休眠唤醒功能，standby m0 唤醒仅支持 gpio/bod/lvr。
   - 例程演示使用的 sram 版本芯片，芯片掉电程序丢失，故不演示 bod/lvr 唤醒。
   - 例程演示的 standby m0/m1 唤醒后均发生复位
   - 例程演示的 standby m0/m1 唤醒后产生 lp 中断
   - 例程输入参数根据串口输出的 log 信息输入

```
standby m1(32k 唤醒):
wake up by rcl 32k
Standby M1
slptmr int

sram addr 0x20000B00,value=20000348
+------------------------------------------------------------------------------+
|                      PAN271x LowPower Sample Code.                           |
+------------------------------------------------------------------------------+
...

standby m0(gpio 唤醒):
wake up by normal gpio
Standby M0

sram addr 0x20000B00,value=20000348
+------------------------------------------------------------------------------+
|                      PAN271x LowPower Sample Code.                           |
+------------------------------------------------------------------------------+
...
```

6. 串口输入字符 '5'，standby m1 cpu 保持及多唤醒源持续唤醒，唤醒方式支持 32k timer 及 gpio P12，32k timer 定时 2s 唤醒，gpio 随机唤醒，循环进入休眠，唤醒后程序表现为不复位重新执行而是继续执行，输出结果同第 4 点。

6 RAM/Otp 资源使用情况

- Otp Size: 10.71kB
- RAM Size: 0.82kB

3.3.5 Reset

1 功能概述

本例程演示芯片复位相关的功能。

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
- 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\miscellaneous\reset`

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：

```
+-------------------------------+
reset flag reg RSTSTS[0] --> CHIP0RF
reset flag reg RSTSTS[1] --> PINRF
reset flag reg RSTSTS[2] --> WDTRF
reset flag reg RSTSTS[3] --> LVRRF
reset flag reg RSTSTS[4] --> BODRF
reset flag reg RSTSTS[5] --> SYSRF
reset flag reg RSTSTS[6] --> PORRF
reset flag reg RSTSTS[7] --> CPURF
reset flag reg RSTSTS[8] --> CHIP1RF

Initial reset state is 0x00000002
Start clear all reset status
reset state is 0x00000000

+-------------------------------------------------------+
|                     PAN271x Reset Sample Code.        |
+-------------------------------------------------------+
|    press key to start test                            |
|    Input'0'   chip0_reset();                          |
|    Input'1'   pin_reset();                            |
|    Input'2'   wdt_reset();                            |
|    Input'3'   lvr_reset();                            |
|    Input'4'   bod_reset();                            |
|    Input'5'   system_reset();                         |
|    Input'6'   cpu_reset();                            |
|    Input'7'   chip1_reset();                          |
+-------------------------------------------------------+
```

2. 串口输入字符‘0’，chip0 reset 功能，复位范围为整个芯片，复位后 RSTSTS bit0 置位。

```
Initial reset state is 0x00000001
Start clear all reset status
reset state is 0x00000000
```

3. 串口输入字符‘1’，pin reset 功能，复位范围为整个芯片，复位后 RSTSTS bit1 置位。

```
Initial reset state is 0x00000002
Start clear all reset status
reset state is 0x00000000
```

4. 串口输入字符‘2’，wdt reset 功能，复位范围为整个芯片 (DVDD 除外)，复位后 RSTSTS bit2 置位。

```
Initial reset state is 0x00000004
Start clear all reset status
reset state is 0x00000000
```

5. 串口输入字符‘3/4’，lvr/bod reset 功能，复位范围为整个芯片 (DVDD，自身数字逻辑及 32k 时钟除外)，复位后 RSTSTS bit3/bit4 置位，此处由于 sram 芯片暂不做演示。
6. 串口输入字符‘5’，system reset 功能，复位数字控制区 (cpu debug 及 pmu 模块除外)，复位后 RSTSTS bit5 置位。

```
Initial reset state is 0x00000020
Start clear all reset status
reset state is 0x00000000
```

7. 串口输入字符‘6’，cpu reset 功能，仅复位 CPU，复位后 RSTSTS bit7 置位。

```
Initial reset state is 0x00000080
Start clear all reset status
reset state is 0x00000000
```

8. 串口输入字符 '7'，chip1 reset 功能，复位范围为整个芯片 (DVDD 除外)，复位后 RSTSTS bit8 置位。

```
Initial reset state is 0x00000100
Start clear all reset status
reset state is 0x00000000
```

5 RAM/Otp 资源使用情况

- Otp Size: 5.07kB
- RAM Size: 0.77kB

3.4 私有 2.4G 例程

3.4.1 PRF 2.4G Rx

1 功能概述

本例程演示 PRF 2.4G-Rx 通信的基本流程。

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
- 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\proprietary_rf\prf_rx`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

参考文档: prf_dev_guidance

5 开发者说明

参考文档: prf_dev_guidance

3.4.2 PRF 2.4G Tx

1 功能概述

本例程演示 PRF 2.4G-Tx 通信的基本流程。

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
- 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\proprietary_rf\prf_tx`

双击 Keil Project 文件打开工程进行编译烧录。

4 例程演示说明

参考文档: prf_dev_guidance

5 开发者说明

参考文档: prf_dev_guidance

3.5 解决方案

3.5.1 PRF Dongle

1 功能概述

此 sample 为在 `pan271x` 上演示 Dongle 配合 pan10xx 作为鼠标端控制功能

2 环境要求

- 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
- 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: `01_SDK\samples\solutions\prf_dongle\keil\prf_dongle`

使用 `keil` 打开项目进行编译烧录。

4 演示说明

1. keil 编译烧录 `prf_dongle` 程序，此时进入对码状态，dongle 端在 2412 频点 `{ 0x7b, 0x41, 0x29, 0x71 }` 地址（PIPE0）进行扫描，并根据本地 mac 地址的后四位计算出私有的 8 个跳频地址。
2. pan10xx 鼠标按下左中右三个按键进入配对模式，在 2412 频点 `{ 0x7b, 0x41, 0x29, 0x71 }` 发送鼠标端的 mac 地址。
3. pan271x dongle 收到对码地址频点的鼠标端 mac 地址后，回复本地 mac 地址，并且把鼠标端 mac 地址后四位设置为 PIPE1 地址，如果在 PIPE1 地址上任意私有频点收取到数据，结束对码状态，如果没有收到私有地址频点的通信包，timeout 中仍有机会切换到配对频点（鼠标端收到对码地址频点的 dongle mac 地址后，计算出私有的 8 个跳频地址，切换到鼠标 mac 地址后四位作为地址，跳频发送数据）。

4. 对码成功后鼠标可以控制 PC。

   串口 LOG 如下所示：

```
[11:27:05.484] 收 ←CPU @ 48000000Hz
usb_init
plug in

[11:27:05.535] 收 ←get_mac_addr: 0x11 0x22 0x33 0x44 0x55 0x66
calculate_combo_freq use 0x33 0x44 0x55 0x66
x = 1 y = 5
2464 abandon
calculate private freq: 2428 2446 2466 2402 2420 2438 2456 2474

[11:27:05.566] 收 ←0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x50
USB isr in: Reset evt

[11:27:05.637] 收 ←USB isr in: Reset evt

[11:27:14.782] 收 ←hop end @ 2438

[11:27:14.816] 收 ←save new pair addr1: 0xE1 0xA0 0x0 0xA 0x1B 0x9B
```

5 开发说明

5.1 配对信息存储及接口说明 PAN271x 使用 EEPROM 来保存配对信息，配对信息主要包含配对后的通信地址。EEPROM 使用 GPIO 模拟 IIC 控制。

EEPROM 读写接口：

```
P24C02_ReadSequential(0x10, read_addr0, 6);
P24C02_ReadSequential(0x20, read_addr1, 6);

data_printf(read_addr0, 6);
data_printf(read_addr1, 6);
memcpy(&prf_state.pair_private_addr[0][0], read_addr0, 6);
memcpy(&prf_state.pair_private_addr[1][0], read_addr1, 6);

void write_pair_addr(void)
{
        if(pair_addr0_write) {
                pair_addr0_write = false;
                P24C02_WritePage(0x10, &prf_state.pair_private_addr[0][0], 6);
                mini_printf("save new pair addr1: ");
                data_printf(&prf_state.pair_private_addr[0][0], 6);
        }

        if(pair_addr1_write) {
                pair_addr1_write = false;
                P24C02_WritePage(0x20, &prf_state.pair_private_addr[1][0], 6);
                mini_printf("save new pair addr2: ");
                data_printf(&prf_state.pair_private_addr[1][0], 6);
        }
}
```

5.2 PRF 说明

5.2.1 初始化

```
pan_prf_config_t __align(4) rf_config =
{
        .work_mode                          = PRF_MODE_ENHANCE,//PRF_MODE_NORMAL,PRF_MODE_ENHANCE
        .chip_mode                          = PRF_CHIP_MODE_SEL_XN297,//PRF_CHIP_MODE_SEL_NRF,//
↪PRF_CHIP_MODE_SEL_XN297,
        .trx_mode                          = PRF_RX_MODE,
        .phy                                  = PRF_PHY_2M,//PRF_PHY_250K,//PRF_PHY_1M,PRF_PHY_2M
        .crc                                  = PRF_CRC_SEL_CRC16,
        .scr                                  = PRF_SCR_SEL_EN,
        .mode_conf                          = PRF_BLE_CONF,//PRF_BLE_CONF
        .pipe                                  = PRF_PIPE0,
        .rx_timeout                         = 10000,                          //us
        .rf_channel                         = PAIR_PUBLIC_CHANNEL,
        .tx_no_ack                          = DISABLE,
        .rx_length                          = 30,
        .sync_length                  = 4,
        .sync                                  = PAIR_PUBLIC_ADDR,
        .crc_include_sync         = ENABLE,
        .scr_include_sync         = ENABLE,
        .auto_pyl_flag                = ENABLE,
        .pid_manual_flag         = ENABLE,
        .endian                                = PRF_BIG_ENDIAN,
        .tx_power                          = 8,
};


void dongle_prf_init(void)
{
        panchip_prf_init(&rf_config);

        /* enable len err isr */
        panchip_prf_rx_length_irq_cfg(rf_config.rx_length);

        SYS_SET_MFP(P2,1,LL_DEBUG_0);                //tx_on
        SYS_SET_MFP(P2,0,LL_DEBUG_1);                //rx_on
        SYS_SET_MFP(P0,4,LL_DEBUG_9);                //match

        dongle_prf_pair_init();

        panchip_prf_trx_start();
}
```

1. 全局变量 `rf_config` 基本包含了 2.4g 的所有配置，定义好后就可以使用 `panchip_prf_init(&rf_config);` 接口对 RF 初始化。
2. RF 初始化完后，开启 len err 中断，rf 状态可控，增强型模式下，一定会进行完成 `r->t/timeout/crc/len err` 状态，中断内控制好下一次 trx start 之前进行数据存储操作、状态控制，不会引起 rf 死机状态
3. LL DEBUG 信号可以抓取 rf 工作中的时序，但需要注意，开启 debug 信号时会一定程度影响 rf，crc 率会变高。调试阶段可以打开，正常使用要关闭。
4. `dongle_prf_pair_init();` 中获取 dongle mac 地址，计算出配对频点（通过 dongle mac 后四字节），并且设置可能存储过的配对地址作为 rf 收取地址

```
 /* set PRI_RF_MODE_SEL_RX addr will set the rx addr & pipe */
 /* set PRI_RF_MODE_SEL_TX addr will set the single tx addr, no set tx will use init tx␣
↪addr */
 /* set PRI_RF_MODE_SEL_TRX addr will set the trx addr both, no set tx will use init tx␣
↪addr */
panchip_prf_set_addr(addr_format, 4, PRF_PIPE1, PRI_RF_MODE_SEL_RX);
panchip_prf_set_addr(addr_format, 4, PRF_PIPE2, PRI_RF_MODE_SEL_RX);
```

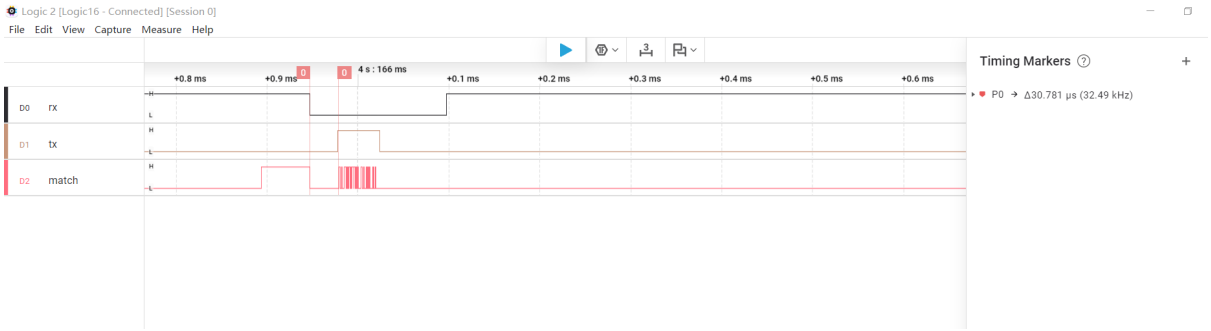
设置 PRI_RF_MODE_SEL_RX 地址仅设置接收 PIPE，不会设置发送地址

rx isr 中需要设置 payload 并且设置回复时的 tx 地址，设置发送地址参考

```
/* set addr will set the tx addr, if want to ack data from dynamic rx pipe ,rx isr must set tx addr */
CONFIG_RAM_CODE void app_prf_set_tx_addr(uint8_t *addr)
{
#if SET_TX_ADDR_DIRECT
/* 1.25us */
uint32_t tx_addr;

tx_addr = (addr[0]) | (addr[1] << 8) | (addr[2] << 16) | (addr[3] << 24);

PRI_RF_WRITE_REG_VALUE(PRI_RF, R05_TX_ADDR_L, L32B, tx_addr);
#else
/* 6us no used third param */
panchip_prf_set_addr(addr, 4, PRF_PIPE0, PRI_RF_MODE_SEL_TX);
#endif

}
```

1. 最后进行 panchip_prf_trx_start(); 开启 tx;

5.2.2 **时间说明** 为了适配 pan10xx 的键鼠，pan271x dongle 的 rx 转 tx 的时间配置成 30us。如果往大了调鼠标端会收不到 ack。如下图所示：



rx isr 中需要设置 payload 并且设置回复时的 tx 地址，设置地址时，可以快速设置 4 字节地址或者调用底层封装接口设置，分别占用 1us 或者 6us，影响还好，就算当前次不能及时设置好地址，依然会在下次配对地址收到数据后在对应地址回复出

```
/* set addr will set the tx addr, if want to ack data from dynamic rx pipe ,rx isr must set tx addr */
CONFIG_RAM_CODE void app_prf_set_tx_addr(uint8_t *addr)
{
 #if SET_TX_ADDR_DIRECT
 /* 1.25us */
 uint32_t tx_addr;

 tx_addr = (addr[0]) | (addr[1] << 8) | (addr[2] << 16) | (addr[3] << 24);

 PRI_RF_WRITE_REG_VALUE(PRI_RF, R05_TX_ADDR_L, L32B, tx_addr);
 #else
 /* 6us no used third param */
 panchip_prf_set_addr(addr, 4, PRF_PIPE0, PRI_RF_MODE_SEL_TX);
 #endif
}
```

5.2.3 多 pipe 使用说明

多 pipe 地址读取的功能，灵活使用可以实现配对地址和多通道地址同时可以收取回复数据，基于此 feature 可以扩展多个主机设备配对同一 dongle，目前主机端最快速度在 250us 内可以完成发收，所以多设备自行处理的条件下，通过重传可以做到至少 2 个设备配对同一 dongle 后，可以达到 1000hz 上报率

演示说明：

1. 准备 2 个 pan1080 8k 鼠标，鼠标 1 下载默认程序，鼠标 2 下载 PAIR_COMBO_DG_DEV2=1 的第二设备程序
2. 编译下载 PAIR_ONLY_ONE=0 的 pan271x dongle 程序
3. 鼠标 1 进入配对状态，复位 pan271x，鼠标 1 配对成功，鼠标 1 不移动情况下鼠标 2 进入配对或者上电状态，鼠标 2 配对成功
4. 鼠标 1 可以达到 1000hz 上报率，鼠标 2 默认配对后 125hz 上报率

对码说明参考演示说明部分的第四点，补充说明鼠标 1 和鼠标 2 的区别在于，配对过程数据字段，回复的数据除了各自 mac 地址，还有额外的数据字段区分对应配对到哪一个 pipe，pan271x rf 中确认私有 mac 搜索到数据通信后不再添加其他地址，重新上电会重新进入可添加 pipe 地址状态

```
        pkt_pair.magic = PAIR_MAGIC;
        #if PAIR_COMBO_DG_DEV2
        pkt_pair.type = 0x20;
        #else
        pkt_pair.type = PAIR_MOUSE_8K_FLAG;
        #endif
        memcpy(pkt_pair.mac, pair_ctrl.pair_own_addr, 6);
        memcpy(tx_payload.data, &pkt_pair, 9);
```

5.3 USB 说明

USB init 软件过程基本与 pan10xx 一致，其他部分使用也一致

```
void usb_init(void)
{
        uint32_t reg_tmp;

        SYS->SYS_CTRL |= (SYS_CTRL_USB_PU_Msk | SYS_CTRL_USB_PU2_Msk | SYS_CTRL_USB_EN_Msk);
        /*usb debounce time*/
        SYS->SYS_CTRL = ((SYS->SYS_CTRL & ~SYS_CTRL_VALID_REMOVAL_CYCLES_Msk) | 100);

        reg_tmp = READ_REG(USB->INT_USBE);
        reg_tmp |= 0x10;
        WRITE_REG(USB->INT_USBE,reg_tmp);

//        NVIC_EnableIRQ(USB_IRQn);

        mini_printf("usb_init\n");
        NVIC_SetPriority(USB_IRQn, 1);
        NVIC_EnableIRQ(USB_IRQn);
}
```

6 RAM/Flash 资源使用情况

PAN271x:

```
- Flash Size:  14.79kB
- RAM Size:  1.80kB
```

系统组件例程

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

例程	说明
PAN USB Mouse	演示 PanUsb 组件的 HID Mouse Device（鼠标设备）的基本功能

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
CLKTRIM	演示 ClkTrim 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
I2C	演示 I2C Driver 的基本功能与使用方法
KSCAN	演示 KeyScan Driver 的基本功能与使用方法
OTP	演示 OTP 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
SPI	演示 SPI 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

私有 2.4G 例程

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

例程	说明
PRF 2.4G Rx	演示 PRF 2.4G-Rx 通信的基本流程
PRF 2.4G Tx	演示 PRF 2.4G-Tx 通信的基本流程

其他例程

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

例程	说明
Blinky	演示通过 GPIO 推挽输出控制 EVB LED 灯闪烁的方法
CoreMark	演示 CoreMark 基准测试
Debug Protect	演示芯片 Debug Protect 调试保护机制
Low Power	演示芯片 Low Power 低功耗模式
Reset	演示芯片复位相关的功能

解决方案

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

例程	说明
PRF Dongle	PAN271x 2.4G Dongle 例程，用于配合 PAN107x/PAN108x 多模键鼠使用

Chapter 4

开发指南

4.1 PRF 2.4G 开发指南

4.1.1 1 概述

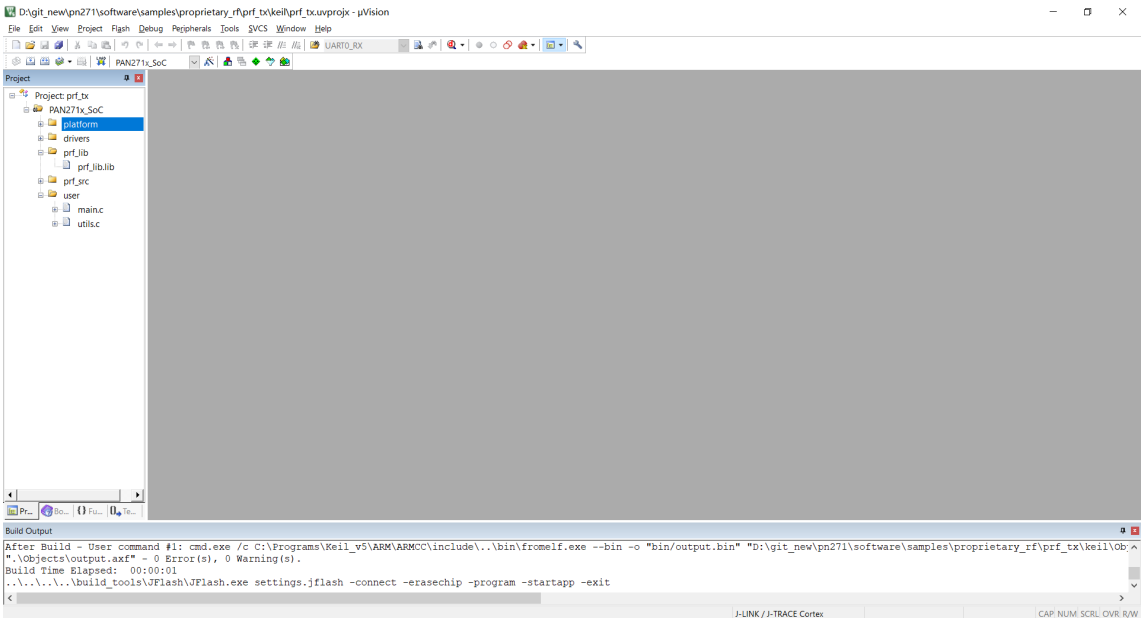
本文介绍的内容有以下几点：

1. TRX 例程功能：
   - 发射端例程的功能：每隔 500ms 发送一次 2.4G 数据包，长度 5 个字节。
   - 发射端例程的功能：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

4.1.2 2 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
   - PC 串口工具：Panchip Serial Assistant V0.0.011.exe

   “PRF_TX_SAMPLE” 的板子需要搭配和” PRF_RX_SAMPLE” 的板子一起使用，两个例程的配置要一样。
2. 编译和烧录：
   1. 项目位置：
   - TX 端：” samples\proprietary_rf\prf_tx”。
   - RX 端：” samples\proprietary_rf\prf_rx”。
   1. TX 端和 RX 端工程界面，如下图所示：



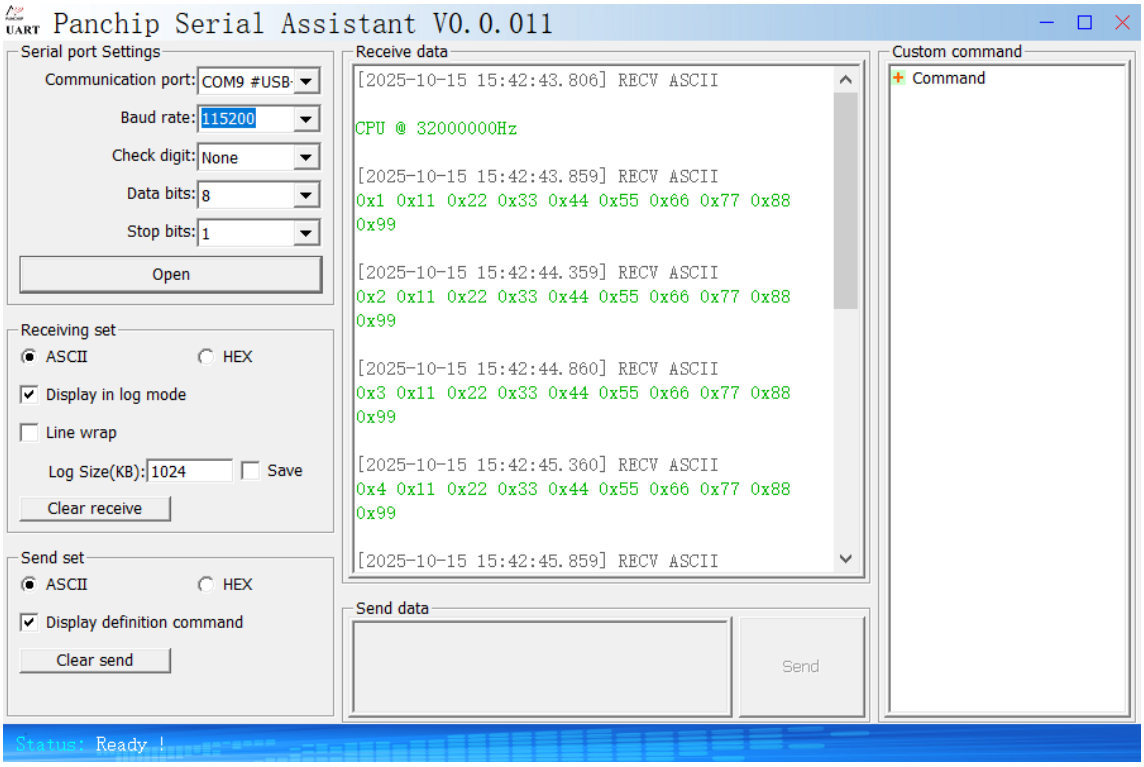
工程默认配置是 XN297 模式，增强型。

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。

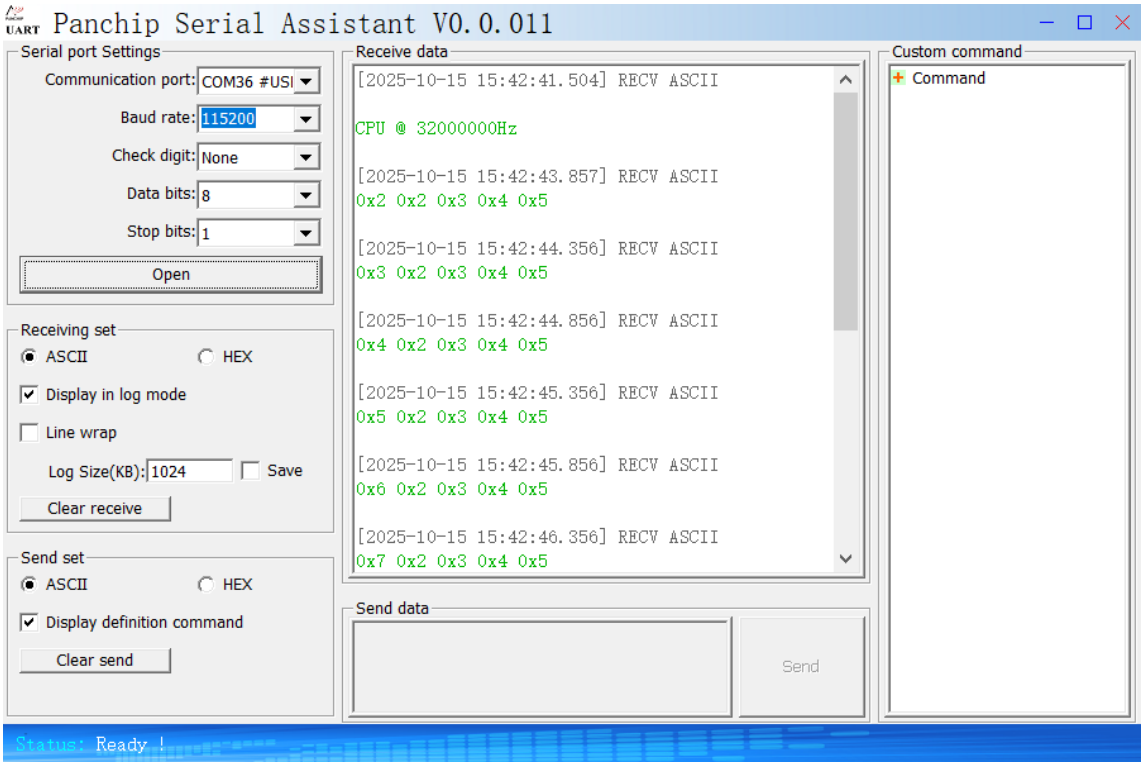
4.1.3 3 演示说明

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：



接收端输出结果：



4.1.4 4 开发说明

4.1 2.4G 初始化配置

配置的结构体“pan_prf_config_t”，各成员介绍如下：

Type	name	Description
prf_mode_t	work_mode	工作模式配置，包括普通型、增强型、自定义增强型
prf_chip_mode_sel_t	chip_mode	xn297 通信协议、NRF 通信协议和 HS62XX 通信协议配置
prf_trx_mode_t	trx_mode	收发模式配置
prf_phy_t	phy	通信速率配置，可配置为 1M、2M、CODED_S8、CODED_S2、250K、250K_S2、250K_S8
prf_crc_sel_t	crc	数据包 CRC 配置，可配置为 crc 8bit，crc 16bit，crc 24bit ，no crc
prf_scramble_sel_t	scr	数据包扰码的配置，可配置为使用扰码和不使用扰码
prf_mode_conf_sel_t	mode_conf	2.4g 不同模式射频参数配置，设置 deviation
uint16_t	rx_timeout	接收超时时间配置，最大 50000us
uint16_t	rf_channel	2.4g 频点配置，任意频点可设 (2400Mhz~2484Mhz)
uint8_t	tx_no_ack	配置增强型模式下 tx 是否需要 ack
uint8_t	rx_length	rx 接收数据包长度配置，增强型模式配置不生效
uint8_t	sync_length	接入地址长度配置，可配置为 2、3、4、5 字节
uint8_t	addr[5]	接入地址的内容
uint8_t	crc_include_sync	crc 作用域是否包含接入地址（XN297 和 NRF24L01 必须使能），NRF52 可配
uint8_t	src_include_sync	扰码作用域是否包含接入地址，XN297 包含，NRF 不包含
uint8_t	auto_pyl_flag	是能后普通型也能动态 payload，仅支持 NRF 模式，XN297 模式不支持
prf_pipe_t	pipe	设置不同地址通信通道（0~2）
int8_t	tx_power	设置发射功率，范围 (0dbm~12dbm)
uint8_t	pid_manual_flag	pid 手动配置的标志，使能后可以手动设置 pid（0~3）
prf_endian_t	endian	payload 大小端配置，XN297L 和 NRF24L01 都是大端，NRF52 可配

prf_mode_t：

Type	Value	Description
PRF_MODE_NORMAL	0	普通型
PRF_MODE_ENHANCE	1	增强型
PRF_MODE_NORMAL_M1	2	自定义增强型

prf_chip_mode_sel_t：

Type	Value	Description
PRF_CHIP_MODE_SEL_XN297	2	XN297 模式
PRF_CHIP_MODE_SEL_NRF	3	NRF 模式
PRF_CHIP_MODE_SEL_NRF52	4	NRF52 模式
PRF_CHIP_MODE_SEL_HS62XX	5	HS62XX 模式

prf_trx_mode_t：

Type	Value	Description
PRF_TX_MODE	0	2.4G 发射
PRF_RX_MODE	1	2.4G 接收

prf_phy_t：

Type	Value	Description
PRF_PHY_1M	1	1M 通信速率
PRF_PHY_2M	2	2M 通信速率
PRF_PHY_CODED_S8	3	S8 编码
PRF_PHY_CODED_S2	4	S2 编码
PRF_PHY_250K	5	250k 通信速率
PRF_PHY_250K_S2	6	250K S2 编码
PRF_PHY_250K_S8	7	250K S8 编码

prf_crc_sel_t:

Type	Value	Description
PRF_CRC_SEL_NOCRC	0	no crc
PRF_CRC_SEL_CRC8	1	crc 8bit
PRF_CRC_SEL_CRC16	2	crc 16bit
PRF_CRC_SEL_CRC24	3	crc 24bit

prf_scramble_sel_t:

Type	Value	Description
PRF_SRC_SEL_NOSRC	0	不使能扰码
PRF_SRC_SEL_EN	1	使能扰码

prf_mode_conf_sel_t:

Type	Value	Description
PRF_NRF_CONF	0	NRF 模式 deviation 配置，1M PHY 160k，2M PHY 320K，250K PHY 160K
PRF_NRF52_CONF	1	NRF52 模式 deviation 配置，1M PHY 170k，2M PHY 330K，250K PHY 170K
PRF_BLE_CONF	2	模拟 BLE 帧格式 deviation 配置，1M PHY 250k，2M PHY 500K，LR&250K PHY 250K
PRF_XN297L_CONF	3	XN297 模式 deviation 配置，1M PHY 300k，2M PHY 600K，250K PHY 170K

prf_addr_length_sel_t:

Type	Value	Description
PRF_ADDR_LENGTH_SEL_3	3	3 BYTE 地址长度
PRF_ADDR_LENGTH_SEL_4	4	4 BYTE 地址长度
PRF_ADDR_LENGTH_SEL_5	5	5 BYTE 地址长度

prf_pipe_t:

Type	Value	Description
PRF_PIPE0	1«0	管道 0
PRF_PIPE1	1«1	管道 1
PRF_PIPE2	1«2	管道 2

prf_endian_t:

Type	Value	Description
PRF_BIG_ENDIAN	0	大端模式传输
PRF_LITTLE_ENDIAN	1	小端模式传输

4.2 API 介绍

panchip_prf_init

```
void panchip_prf_init(pan_prf_config_t *p_config);
```

初始化配置的结构体“pan_prf_config_t”

panchip_prf_trx_start

```
void panchip_prf_trx_start(void);
```

调用此接口，在 RX 状态下开始 RX，在 TX 状态下开始 TX。

panchip_prf_set_data

```
void panchip_prf_set_data(panchip_prf_payload_t *p_payload);
```

TX 模式下设置 payload 的内容和长度，RX 模式下设置 ack payload 的内容和长度

panchip_prf_data_rec

```
uint8_t panchip_prf_data_rec(panchip_prf_payload_t *p_payload);
```

RX 模式下获取接收的数据和长度，返回值是数据长度。**必须在 RX 中断或者 TX 中断读取数据**，在其他地方读数据会异常。

panchip_switch_prf

```
void panchip_switch_prf(pan_prf_config_t *p_config);
```

切换通信协议的帧结构，XN297 模式和 NRF 模式。

panchip_prf_set_chn

```
void panchip_prf_set_chn(pan_prf_config_t *p_config);
```

设置通信的频点，带内 2400~2484 任意频点可设。

panchip_prf_set_tx_pwr

```
void panchip_prf_set_tx_pwr(int8_t tx_pwr);
```

设置发射的功率，功率范围 0dbm~12dbm。

panchip_prf_mode_conf_set

```
void panchip_prf_mode_conf_set(prf_mode_conf_sel_t conf);
```

设置 deviation，不同模式下射频 dev 参数配置，可以选择 BLE 模式、NRF 模式、XN297 模式、NRF52 模式。

panchip_prf_set_phy

```
void panchip_prf_set_phy(pan_prf_config_t *p_config);
```

设置通信速率，可以选择 1M、2M、PRF_PHY_CODED_S8、PRF_PHY_CODED_S2、PRF_PHY_250K、PRF_PHY_250K_S2、PRF_PHY_250K_S8

panchip_prf_set_trx_mode

```
void panchip_prf_set_trx_mode(prf_trx_mode_t trx_mode);
```

设置接收模式和发射模式。

panchip_prf_set_work_mode

```
void panchip_prf_set_work_mode(prf_mode_t work_mode);
```

设置普通型工作模式和增强型工作模式。

panchip_prf_set_addr

```
void panchip_prf_set_addr(uint8_t *addr, uint8_t len, prf_pipe_t pipe, uint8_t trx_addr);
```

设置通信地址内容、长度、地址通道、地址类型。

地址类型分为以下三种：

trx_addr	Value	Description
PRI_RF_MODE_SEL_TX	0	设置 TX 地址
PRI_RF_MODE_SEL_RX	1	设置 RX 地址
PRI_RF_MODE_SEL_TRX	2	设置 TRX 地址，TX 和 RX 地址相同

TX 地址只有一个 PIPE，RX 地址有 3 个 PIPE。

NOTE：**地址内容不能出现连续的** 10101 **或者** 01010，**否则接收性能会受影响**。

panchip_prf_set_tx_noack

```
void panchip_prf_set_tx_noack(bool flag);
```

设置 TX 增强型模式下是否需要接收 ACK，设置成 true 后，TX 后不会进接收。

panchip_prf_rx_timeout

```
void panchip_prf_rx_timeout(uint16_t time);
```

设置接收的超时时间，单位 us，最大 50000us。

panchip_prf_reset

```
void panchip_prf_reset(void);
```

复位 PRF 状态机，使其处于 IDLE 状态。

panchip_prf_pid_cfg

```
void panchip_prf_pid_cfg(uint8_t pid);
```

增强型模式手动设置 TX PID，PID 设置的范围是 0~3。

panchip_white_init_value

```
void panchip_white_init_value(uint8_t value);
```

白化使能后，白化初始值设置，默认是 0x7f。

panchip_prf_carrier_start

```
void panchip_prf_carrier_start(uint16_t  tx_channel);
```

发射单载波，参数设置任意频点。**必须在** TX **和** RX **退出后使用**。

panchip_prf_carrier_stop

```
void panchip_prf_carrier_stop(void);
```

退出发射单载波模式。

panchip_prf_read_data_rssi

```
int16_t panchip_prf_read_rssi(void);
```

收到数据后，在接收中断中获取数据强度的 RSSI 值。这个 RSSI 会缓存，收到新的数据后值会覆盖。

panchip_prf_read_carrier_rssi

```
int16_t panchip_prf_read_carrier_rssi(void);
```

获取载波强度的 RSSI 值，这个 RSSI 值是实时的，不会缓存。

panchip_prf_read_agcindex

```
int16_t panchip_prf_read_agcindex(void);
```

获取 agc 的档位，范围 0~3。

panchip_prf_rx_length_irq_cfg

```
void panchip_prf_rx_length_irq_cfg(uint8_t value);
```

配置是否使能 rx length err 中断，当 rx 收到数据硬件解析出来的 length 值大于配置的 rx length（结构体中“rx_length”字段），rx length err irq 会产生，普通型和增强型接收都有效。在使用前需要先注册中断函数。

panchip_prf_get_pipe

```
uint8_t panchip_prf_get_pipe (void);
```

收到数据获取地址通道号，范围 0~7。

panchip_prf_set_crc

```
void panchip_prf_set_crc (pcrc_sel_t crc, uint8_t crc_include_sync);
```

设置 crc 的类型和 crc 的范围是否包含地址。

panchip_prf_set_whitening

```
void panchip_prf_set_whitening (prf_scramble_sel_t src, uint8_t src_include_sync);
```

设置白化类型和白化范围是否包含地址。

panchip_prf_set_endian

```
void panchip_prf_set_endian (prf_endian_t endian);
```

设置 payload 数据传输的大小端。

panchip_prf_wl_set

```
void panchip_prf_wl_set (uint8_t match_mode);
```

设置 payload 白名单过滤，payload 过滤的长度可配。

panchip_prf_tx_retrans_set

```
void panchip_prf_tx_retrans_set(uint8_t times, uint16_t interval_us);
```

设置 tx 端重传次数和重传间隔，重传次数最大 7 次，重传间隔最大 32ms。

4.3 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. 涉及到两个接口，panchip_prf_set_addr 和 panchip_prf_get_pipe。panchip_prf_set_addr 为每个 pipe 设置不同的地址，接收端收到数据后可以使用 panchip_prf_get_pipe 获取当前收到数据对应的 pipe 地址。

4.4 中断介绍

中断介绍如如下代码所示：

```
void event_tx_fun(void)
{
        printk("tx done\n");
}               //增强型和普通型模式 TX 结束后会产生 TX 中断，TX 退出，增强型 TX 完了之后会马上进入 RX。

void event_rx_fun(void)
{
        panchip_prf_payload_t rx_payload;

        rx_payload.data_length = panchip_prf_data_rec(&rx_payload);
        printk("rx data:");
        data_printk(rx_payload.data, rx_payload.data_length);
}               //增强型和普通型模式 RX 收到数据后会进入 RX 中断,RX 退出，增强型 RX 完了之后会马上进入 TX。

void event_rx_timeout_fun(void)
{
        printk("rx timeout\n");
}               //接收超时后会进入 timeout 中断,RX 退出

void event_crc_err_fun(void)
{
        printk("crc err\n");
}               //接收数据错误会进入 crc 中断，RX 退出，不会进入 TX

/* 普通型 TX，TX 结束退出后产生 TX 中断，进入下一次 TX 后须调用"panchip_prf_trx_start"。*/

/* 普通型 RX，收到数据后会进入 RX 中断，接收超时会进入 timeout 中断，收到错误数据会进入 crc 中断，进入下一次 RX 后须调用"panchip_prf_trx_start"。*/

/* 增强型 TX，TX 结束后硬件会自动进入 RX，RX 收到数据后会进入 RX 中断，接收超时会进入 timeout 中断，收到错误数据会进入 crc 中断，进入下一次 TX 后须调用"panchip_prf_trx_start"。*/

/* 增强型 RX，RX 收到数据后会硬件会自动进入 TX，TX 结束退出后产生 TX 中断，接收超时会进入 timeout 中断，收到错误数据会进入 crc 中断，进入下一次 RX 后须调用"panchip_prf_trx_start"。*/
```

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，不然会影响 TX 中断从而导致时序错误。

4.1.5 5 Sample 运行流程

prf_sample_rx 例程运行流程

如下图所示：

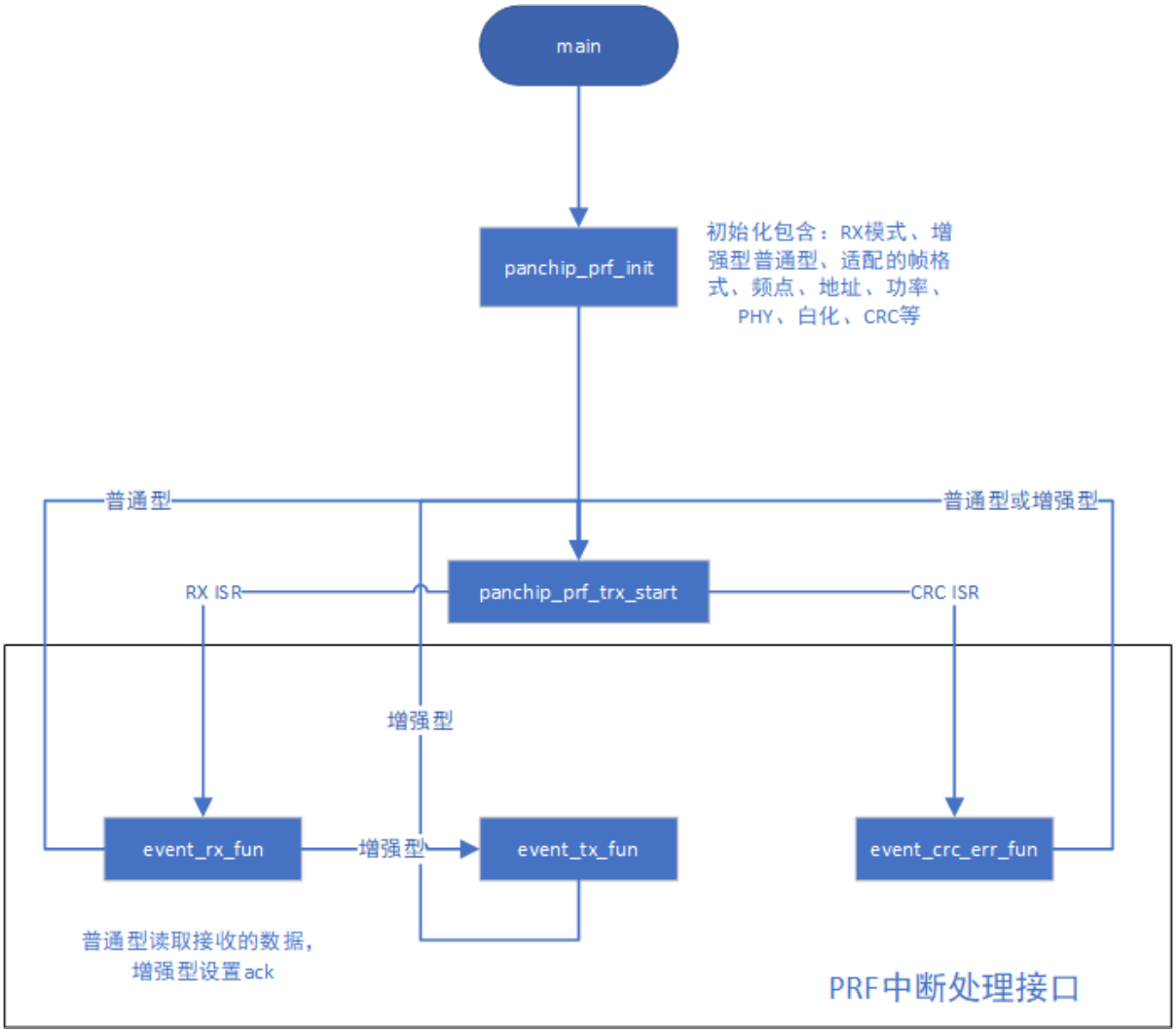


图 1: RX 例程流程图

prf_sample_tx 例程运行流程

如下图所示：

4.1.6 6 2.4G 帧结构介绍

兼容三种帧结构：xn297、nrf24l01、nrf52、hs26xx

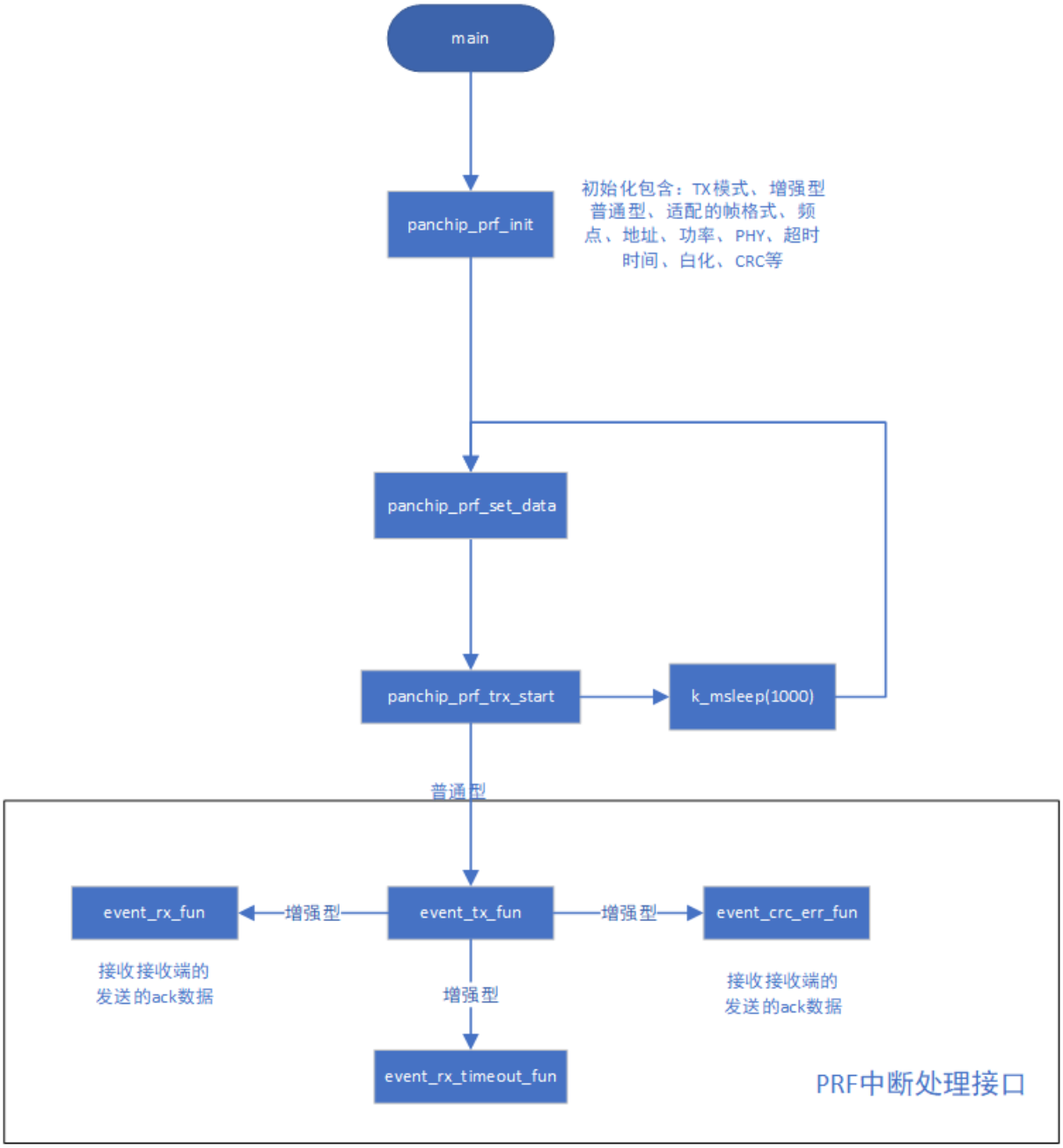

main
panchip_prf_init
初始化包含：TX模式、增强型
普通型、适配的帧格式、频
点、地址、功率、PHY、超时
时间、白化、CRC等
panchip_prf_set_data
panchip_prf_trx_start
k_msleep(1000)
普通型
event_rx_fun
增强型
event_tx_fun
增强型
event_crc_err_fun
接收接收端的
发送的ack数据
增强型
接收接收端的
发送的ack数据
event_rx_timeout_fun
PRF中断处理接口

图 2: TX 例程流程图

6.1 xn297 兼容帧结构

普通型：

3 byte	2~5 byte	0~255 byte	0/1/2/3 byte
preamble(0x710f55)	addr	payload	crc(包含 addr 和 payload)

空中 bit 序：大端模式

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

3 byte	2~5 byte	10bit	0~64 byte	0/1/2/3 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)

数据长度 7bit 最大包长发送可支持 127byte，接收最大包长可支持 128byte。

白化和 crc 的作用域包括 addr+payload+signal。

6.2 nrf2401 兼容帧结构

普通型：

空中 bit 序：大端模式

1 byte	2~5 byte	0~32byte	0/1/2/3 byte
preamble(0x55 或者 0xaa)	addr	payload	crc(包含 addr 和 payload)

preamble：地址第一个 bit 为 0，取 0xaa；地址第一个 bit 为 1，取 0x55。

增强型：在地址和 payload 之间插入 signal(9bit) 数据。

1 byte	2~5 byte	9bit	0~32byte	0/1/2/3 byte
preamble(0x55 或者 0xaa)	addr	signal	payload	crc(包含 addr 和 payload)

signal 结构：

6bit	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)

数据长度 6bit 最大包长发送可支持 63byte，接收最大包长可支持 64byte。

nrf2401 无白化，crc 的作用域包括 addr+payload+signal。

6.3 nrf52 兼容帧结构

nrf52 帧结构如下图所示：

当 s0=0，length=0，s1=0 时，和 nrf2401 普通型帧结构一样。

当 s0=0，length=6，s1=3 时，和 nrf2401 增强型帧结构一样。payload 长度最大 32byte。

当 s0=0，length=8，s1=3 时，长包模式。payload 长度最大 255byte。

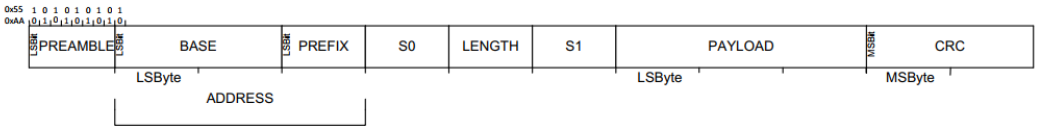


图 3: nrf52 帧结构

空中 bit 序：大小端模式可配

crc 作用域：addr(可配)+s0+length+s1+payload

长包模式下，PAN271x signal 结构如下：

8bit	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)

6.4 HS26XX 帧结构

普通型：

空中 bit 序：大端模式

1 byte	2~5 byte	2byte	0~255byte	1/2 byte
preamble(0x55 或者 0xaa)	addr	guard	payload	crc(包含 addr 和 payload)

preamble：地址第一个 bit 为 0，取 0xaa；地址第一个 bit 为 1，取 0x55。

增强型：在 guard 和 payload 之间插入 signal(9bit) 数据。

1 byte	2~5 byte	9bit	0~32byte	1/2 byte
preamble(0x55 或者 0xaa)	addr	signal	payload	crc(包含 addr 和 payload)

signal 结构：

6bit	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)

数据长度 6bit 最大包长发送可支持 63byte，接收最大包长可支持 64byte。

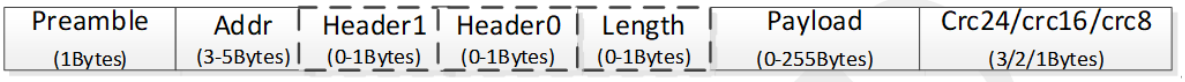
当芯片为 hs6220 时，guard 内容为 ~addr0,addr0。

当芯片为 hs6200 时，guard 内容为 addr0,addr0。

hs26xx 白化初始值为 0x3D，白化作用域包括 payload+signal；crc 的作用域包括 addr+payload+signal。

6.5 自定义帧结构

chip_mode 字段为 “PRF_CHIP_MODE_SEL_NORDIC” 模式时，帧结构如下：



header0、header1、length 为可选字段，可以适配 BLE 和 NRF52 不同模式的帧结构，例如 BLE 广播帧结构：

蓝牙广播中的报头和长度对应 PRF_MODE_NORMAL_M1 模式中的 header0 和 length，header1 没有。

4.1.7 7 2.4G PID 流程

tx 和 rx pid 处理逻辑如下：

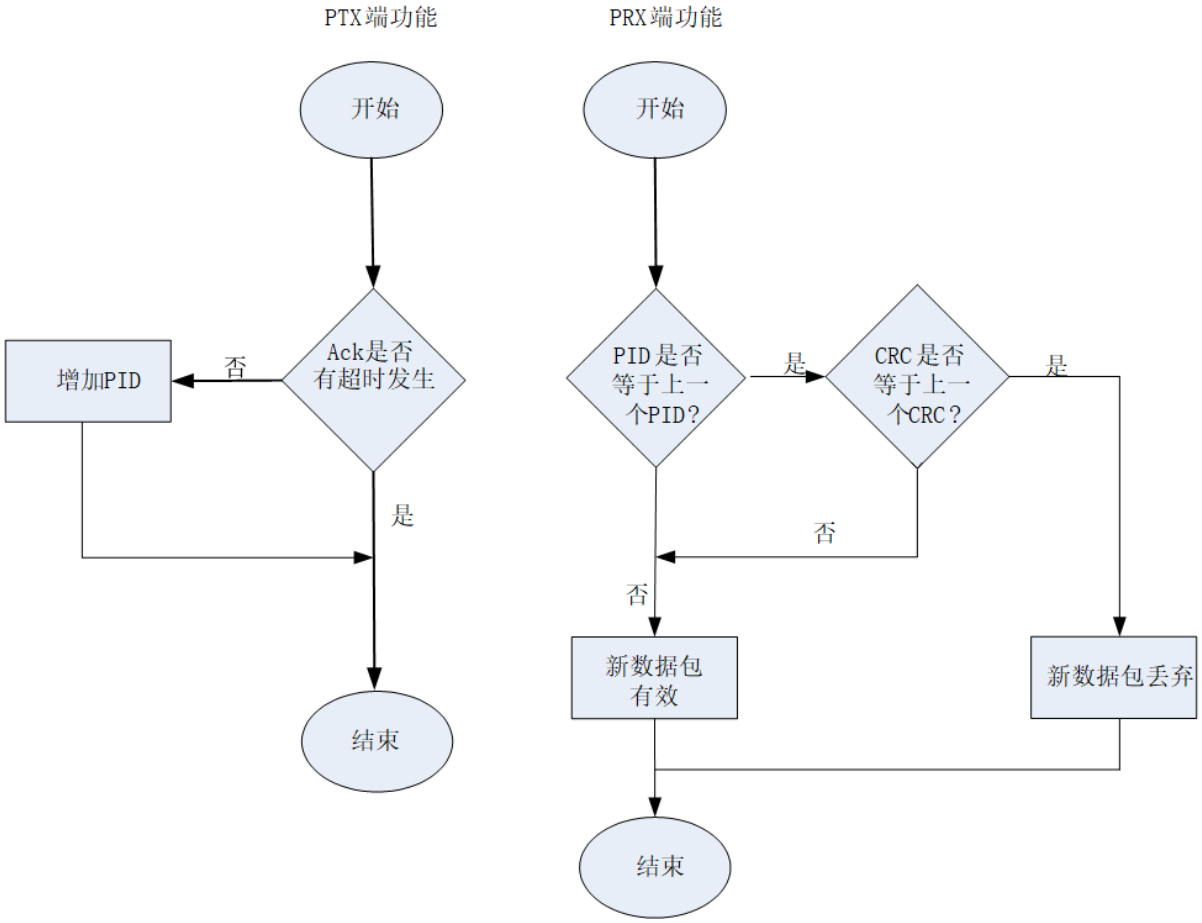


图 4: PID 生成和检测

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：OTP 版本芯片是否支持直接使用 JLink 烧录？

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 一般仅用于调试烧录预生产版本（SRAM 模拟 OTP）的芯片。

若确有使用 JLink 烧录 OTP 版本芯片的需求，我们实际上也是支持的，方法是在给芯片正常供电的基础上，额外向芯片 P22 引脚灌入 6.5v（±250mV）的电压，使 OTP 处于可写状态，然后直接在 Keil 工程中点击 Download 按钮即可烧录。

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，可支持通过 SWD 接口烧录程序。

a）芯片支持通过 JLink 或 Panlink 两种方式进行烧录，其接线方式分别如表 1-1、表 1-2 所示。

JLink	连接	PAN271x SoC
VTref 3.3V	<—>	VBAT
6.5V	<—>	P22
GND	<—>	GND
SWDIO	<—>	P01
SWDCLK	<—>	P00

注：6.5V 为外接高压电源，提供烧录 PAN271x 芯片 OTP 用。需要单独电路提供。

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。

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。

下载

2.1 硬件准备

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

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。

或按照帮助文档方法更新 Panlink 2.0 固件程序。

图 1: 图 2-1 Panlink 2.0 烧录器

图 2: 图 2-2 MiniUSB 连接线

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。

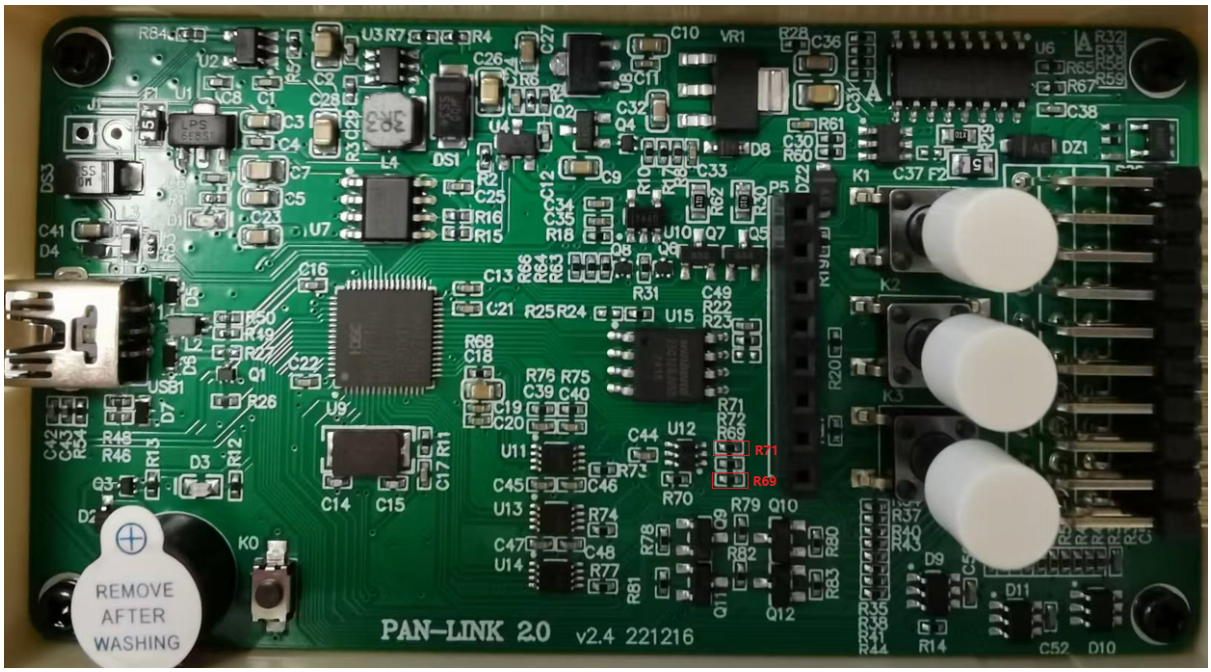


图 3: 图 2-3 Panlink 2.0 烧录器硬件需要特殊处理区域红色标注

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

如上图 2-3 所示为烧录工具界面。

1. 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。

2. 根据需求选择设置其他下载配置
3. 选择下载模式，或直接默认下载到 PAN-LINK 后下载到芯片模式
4. 点击下载开始下载程序到芯片

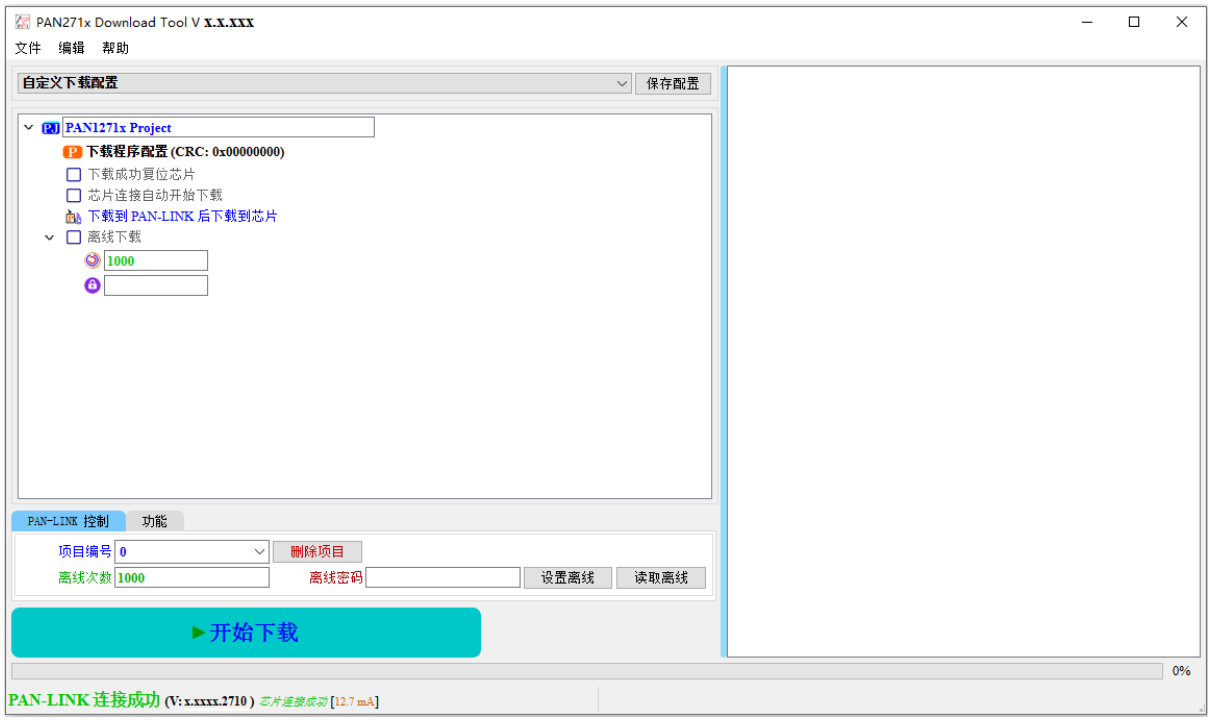


图 4: 图 2-3 烧录工具界面

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。

图 5: 图 2-4 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。

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”。

   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
- 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。

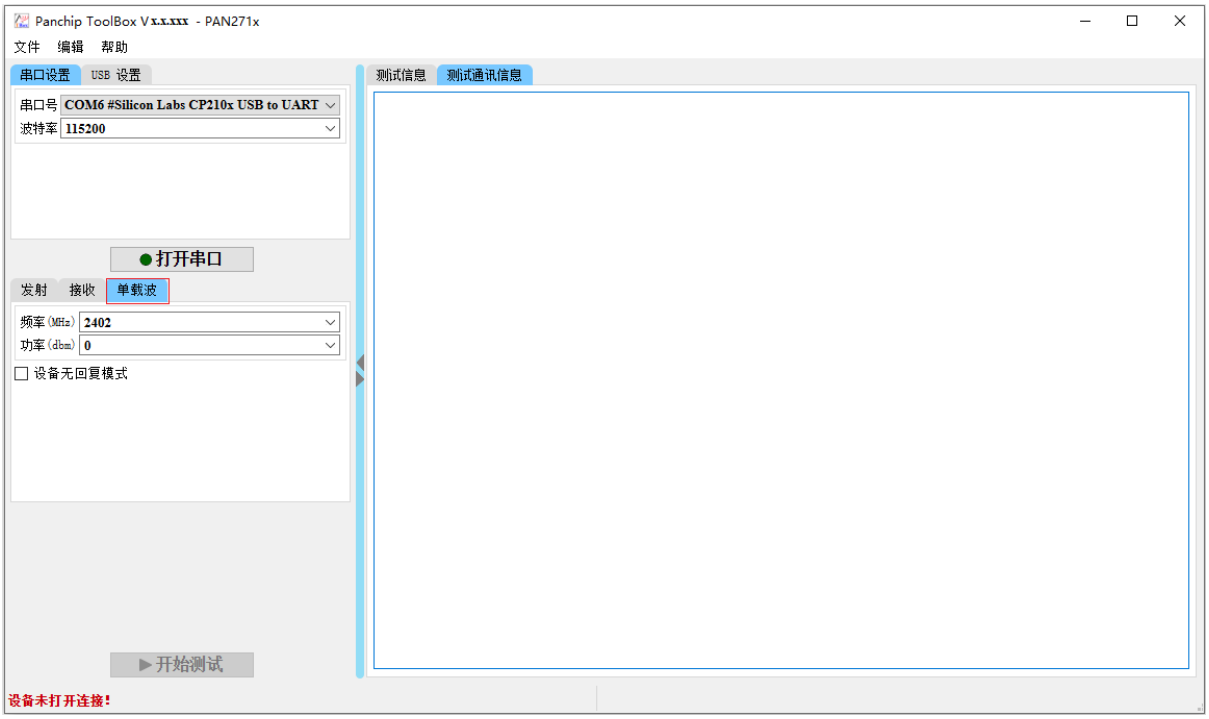


图 6: 图-1 单载波配置界面

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

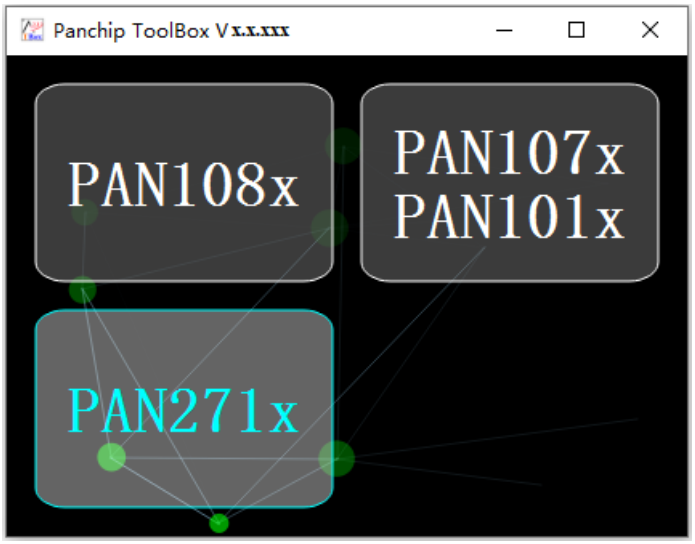


图 1: 启动界面选择 PAN271xx 进入 ToolBox 功能界面

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。

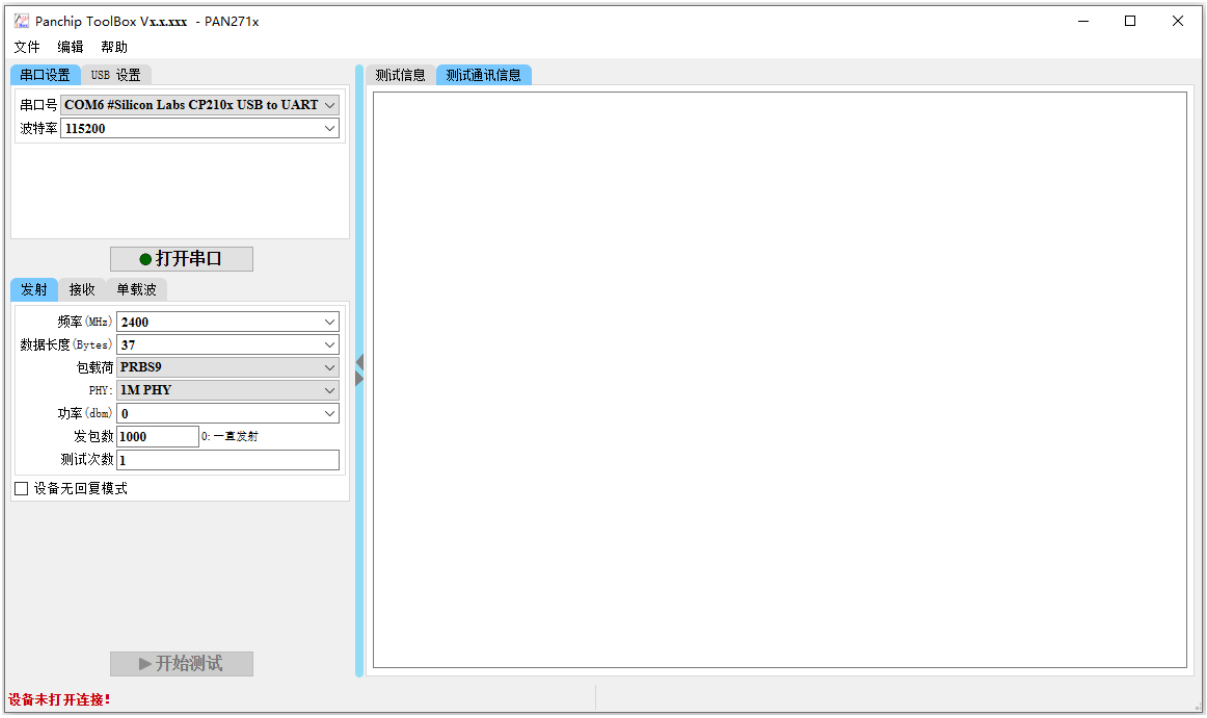


图 2: PAN271x ToolBox 功能界面

图 3: 显示切换功能界面

1.1. RF 测试界面

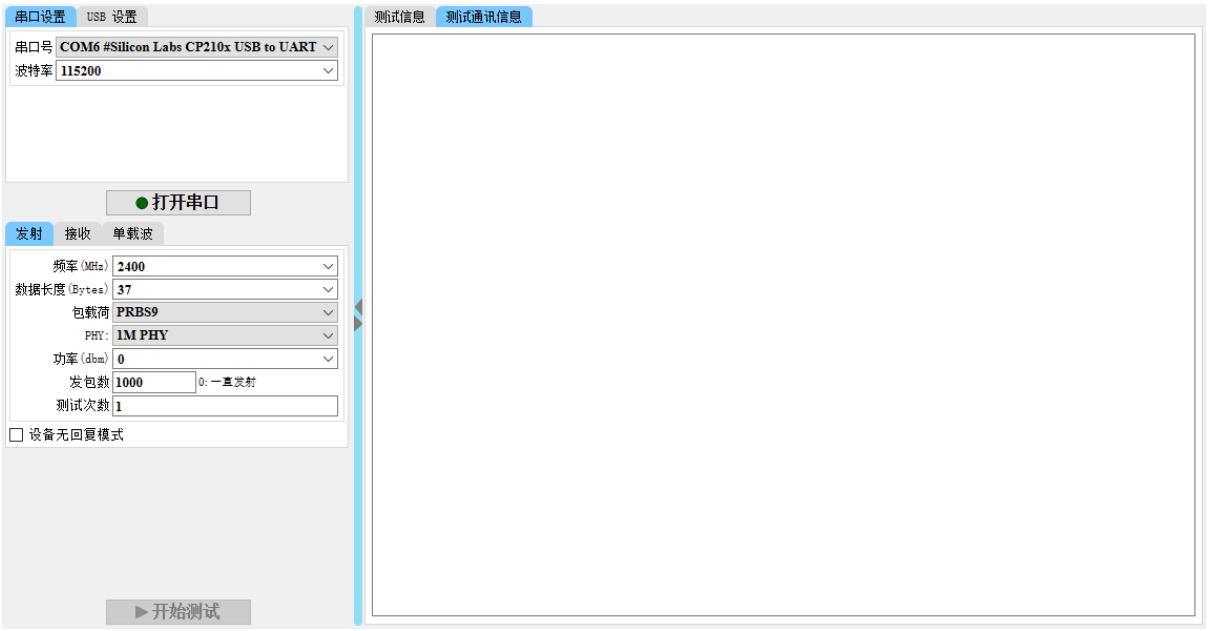


图 4: RF 测试界面

6.1.4 3. 芯片引脚配置

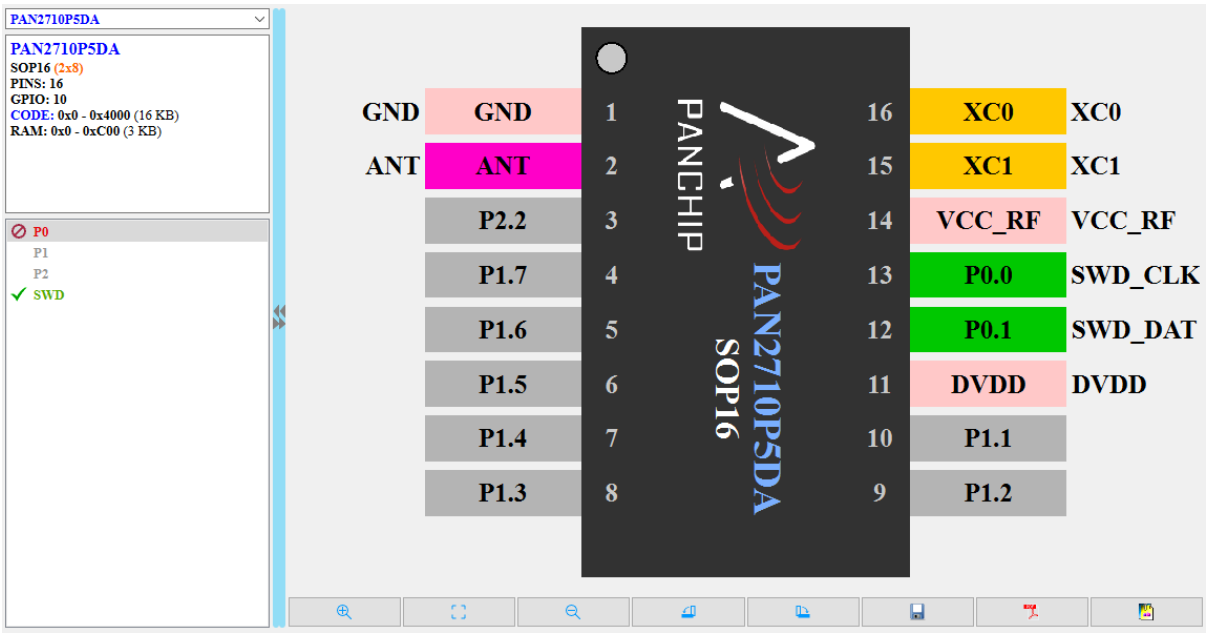


图 5: 芯片引脚配置界面

Chapter 7

其他文档

Chapter 8

更新日志

8.1 PAN271x DK v0.1.0

PAN271x DK `v0.1.0` (2025-10-17) 已发布：

8.1.1 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

8.1.2 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）

8.1.3 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
- PAN107x EVB 硬件资源介绍
- PAN107x 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

8.1.4 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）
- RF 测试固件（芯片固件）