PAN1070 开发套件使用手册

发布 0.1.0

磐启微电子 PAN1070 项目组

2023 年 10 月 24 日

Table of contents

Chapter 1

PAN1070 ZDK

TBD

Chapter 2

PAN1070 NDK

2.1 NDK 快速入门

2.1.1 NDK 快速入门指南

1 概述

本文是 PAN107x NDK 开发的快速入门指引，旨在帮助使用者快速入门 PAN1070 NDK 的相关开发。

2 PAN107x EVB 介绍

目前只有 PAN107B QFN40 测试板。

3 PAN1070 NDK 开发环境确认

3.1 PC 环境检查 请确认 KEIL(推荐 5.25 版本以上), PAN107x 下载依赖的 flm 文件，Jlink 设备等准备就绪。

3.2 快速编译运行一个简单的例程 硬件接线准备，请确认您已经将 PAN107x 测试板的：

1. SWD（P00：SWD_CLK，P01：SWD_DAT，GND：SWD_GND）接口通过 JLink 连接至 PC
2. SoC UART0 接口通过板上的 USB 转串口模块连接至 PC
   - UART0-Tx：`P16`，UART0-Rx：`P17`
3. 打开一个 sample 工程，例如 `01_SDK\nimble\samples\solutions\esl\Keil` 下的 keil 目录下的工程文件 `esl.uvprojx`
4. 点击 `Build` 编译按钮，然后点击 `Download` 按钮进行下载（若无法正常下载，请检查 FLM 文件是否正常载入）。
5. 下载完成可以通过串口观察 log 输出（**串口波特率：**921600）

2.1.2 NDK 开发环境介绍

开发 IDE

KEIL 官方下载连接：

https://www.keil.com/download/product/

当前 NDK 开发使用的工具为 KEIL, 开发中使用的版本为 5.25, 所以建议使用该版本及以上版本。

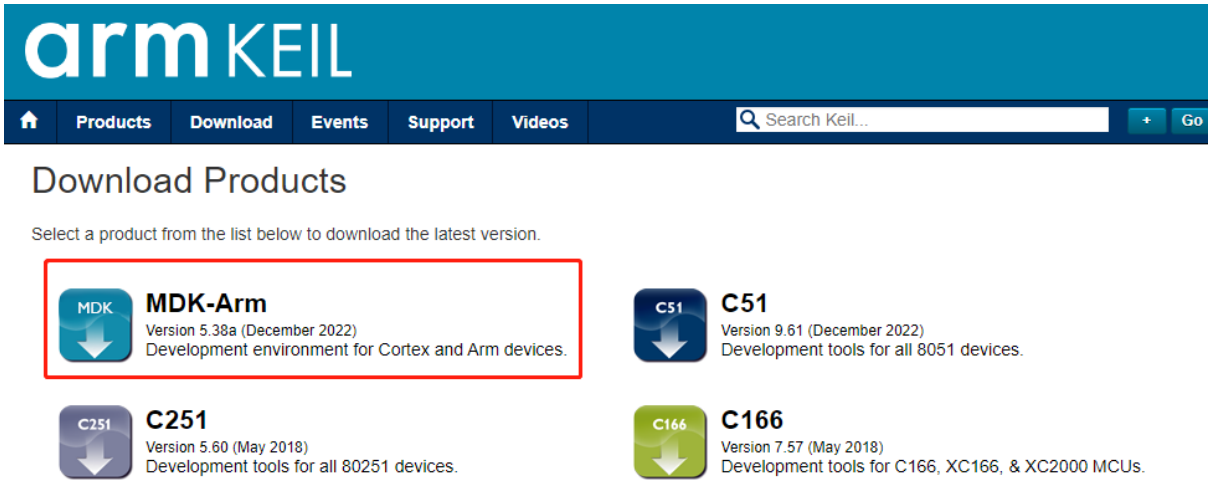


图 1: Keil 下载 MDK-Arm 版本

Keil 使用版本如下：

Keil Flash 下载程序

为使用 Keil + Jlink 烧录代码，请将此目录下的 FLM 文件，拷贝到 Keil MDK 安装目录下，如：

- C:\Keil_v5\ARM\Flash

相关 FLM 文件默认在 SDK 中路径为 `pan1070-ndk\03_MCU\mcu_misc`

- `PAN1080_508KB_FLASH_KEIL.FLM`
- `PAN1080_1020KB_FLASH_KEIL.FLM`

2.1.3 NDK 整体框架介绍

1 简介

PAN1070 NDK 是基于开源蓝牙协议栈 Nimble(Host) 以及开源系统 FreeRTOS，以及私有 BLE Controller 实现完成。Nimble 和 FreeRTOS 均开发源码，BLE Controller 通过标准化 HCI 接口实现。需要注意的是 NDK 依赖的 IDE 主要是 KEIL。

2 NDK 目录结构

PAN1070 NDK 源码树结构如下：

```
<home>/01_SDK
  modules
     hal
  nimble
      README.md
      controller
      host
      lib
      os
      samples
```

- `moudules/hal`：与 ZDK 同根同源，属于外设和 driver 相关的硬件抽象层
- `README.md`：nimble 基本介绍

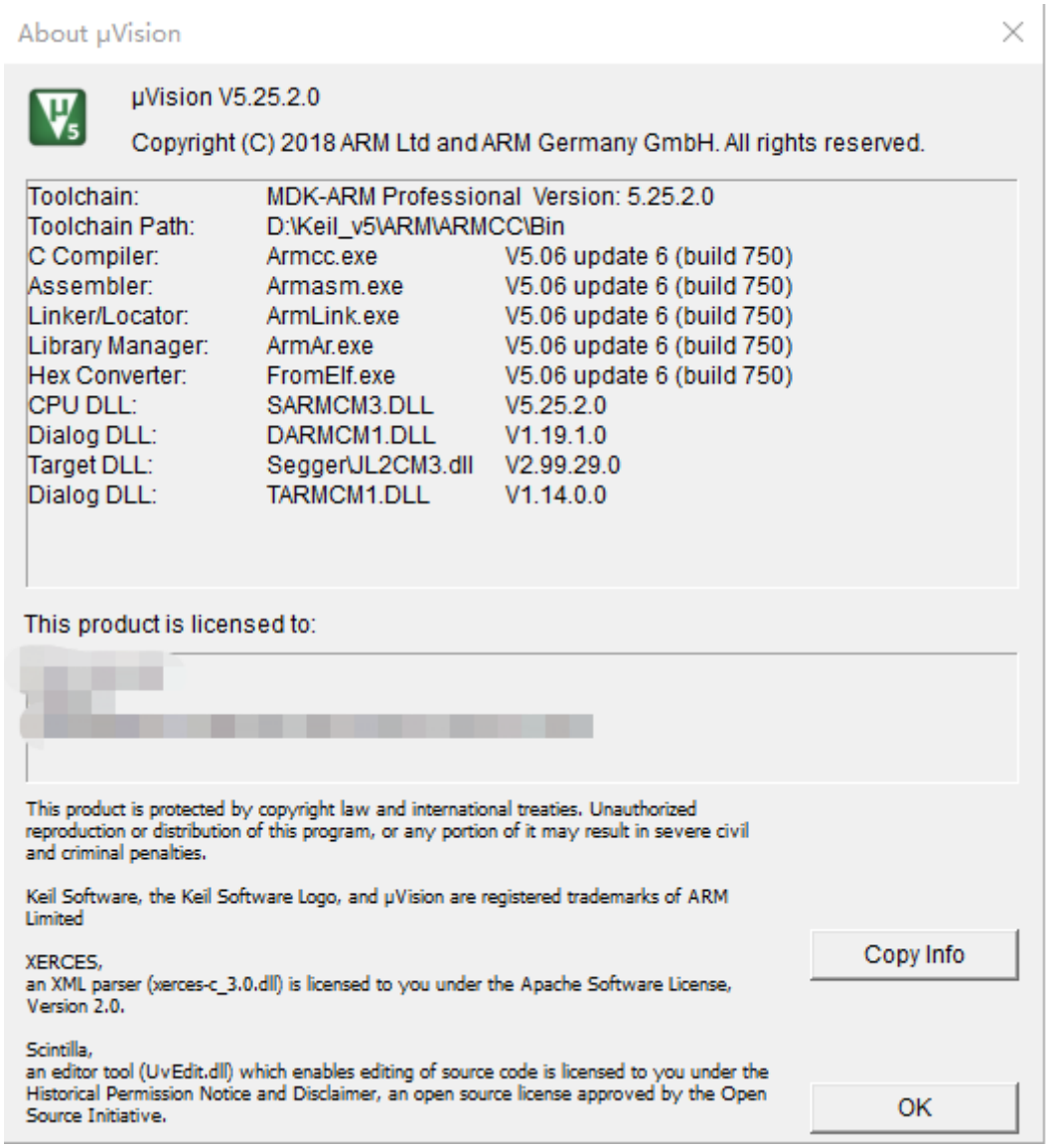

About µVision
µVision V5.25.2.0
Copyright (C) 2018 ARM Ltd and ARM Germany GmbH. All rights reserved.
Toolchain: MDK-ARM Professional Version: 5.25.2.0
Toolchain Path: D:\Keil_v5\ARM\ARMCC\Bin
C Compiler: Armcc.exe V5.06 update 6 (build 750)
Assembler: Armasm.exe V5.06 update 6 (build 750)
Linker/Locator: ArmLink.exe V5.06 update 6 (build 750)
Library Manager: ArmAr.exe V5.06 update 6 (build 750)
Hex Converter: FromElf.exe V5.06 update 6 (build 750)
CPU DLL: SARMCM3.DLL V5.25.2.0
Dialog DLL: DARMCM1.DLL V1.19.1.0
Target DLL: Segger\JL2CM3.dll V2.99.29.0
Dialog DLL: TARMCM1.DLL V1.14.0.0
This product is licensed to:
This product is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it may result in severe civil and criminal penalties.
Keil Software, the Keil Software Logo, and µVision are registered trademarks of ARM Limited
XERCES,
an XML parser (xerces-c_3.0.dll) is licensed to you under the Apache Software License, Version 2.0.
Scintilla,
an editor tool (UvEdit.dll) which enables editing of source code is licensed to you under the Historical Permission Notice and Disclaimer, an open source license approved by the Open Source Initiative.
Copy Info
OK

图 2: Keil 版本

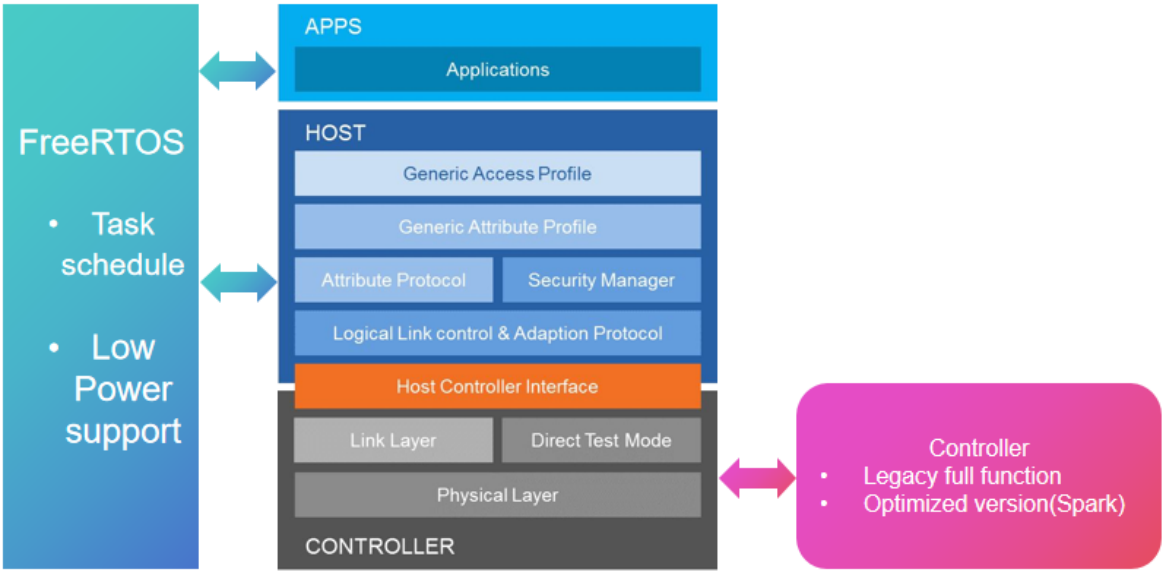


图 3: NDK 整体结构

- `controller`: nimble 工程所需要的 controller 相关头文件
- `host`: nimble host 协议栈所在的主要目录，同时包含 kv_store 组件，该组件主要用于 flash 数据库存储。
- `lib`: 该文件包含两个版本 controller 的实现，具体实现以 lib 的形式添加到 keil 工程中。Controller 两个版本分别是 Origin 版本，该版本是全功能版本，但是具体实现优化比较少，执行速度较慢，代码相对较大；另外一个版本是优化版本 (Spark 版本)，该版本为精简和优化版本，速度更快功耗更低，但是目前主要实现 BLE 4.2+ 版本 feature，其他 5.0+feature 功能后续迭代升级。
- `os`:freertos 的相关代码
- `samples`: 基于 nimble 的相关 demo

2.1 modules

modules 在 NDK 中主要为外设以及 USB，2.4G 等依赖的库文件

```
<home>/01_SDK/modules
.
  hal
      panchip
          panplat
              pan1070
                  bsp
                      cmsis
                      device
                      peripheral
                      radio
                      usb
```

2.2 controller

controller 中主要包含了两个版本 controller 依赖的头文件

```
<home>/01_SDK/nimble/controller
.
  dummy.txt
  pan107x
      shrd_utils
  pan107x_spark
      include
```

2.3 host　host 中包含 nimble 的主体实现以及其他必须的组件，目前 nvs 数据库存储使用是的 kv_store 组件，主要用于存储 ble 的配对信息。当然开发者也可以用于自己的项目存储数据。

```
<home>/01_SDK/nimble/host
.
  kv_store
      mtb_init.c
      mtb_kvstore.c
      mtb_kvstore.h
      mtd_kv_port.h
  nimble
      CODING_STANDARDS.md
      LICENSE
      NOTICE
      README.md
      RELEASE_NOTES.md
      apps
      babblesim
      docs
      ext
      nimble
      porting
      repository.yml
      targets
      uncrustify.cfg
      version.yml
```

- `kv_store` 组件：
  - 支持任何可以建模为块设备的存储，包括内部闪存或外部闪存（例如通过 QSPI）。
  - 通过实例化库的多个实例来对存储进行分区。
  - 设计为抗电源故障。
  - 旨在促进存储的均匀磨损。
- `nimble`：

  参考 nimble 专页介绍，nimble 基于 V1.5.0 版本移植实现。

2.4 lib　lib 中主要包含了两个版本 controller 主体实现的库文件:

```
<home>/01_SDK/nimble/lib
.
  pan107x
      ble.lib
  pan107x_spark
      ble_spark.lib
```

Controller 两个版本分别是 `Origin` 版本，该版本是全功能版本，但是具体实现优化比较少，执行速度较慢，代码相对较大，所对应的库为 `pan107x/ble.lib`;

另外一个版本是优化版本，内部代号为 `Spark`, 所对应的库为 `pan107x_spark/ble_spark.lib`。该版本为精简和优化版本，速度更快功耗更低，但是目前主要实现 BLE 4.2+ 版本 feature，其他 5.0+feature 功能后续迭代升级。

2.5 OS　OS 目前只包含 freertos 的主体代码

2.6 samples　nimble 示例工程：

```
<home>/01_SDK/nimble/samples
.
  solutions
      esl
          keil
          src
```

目前包含一个工程：

- `solutions/esl`：电子货架标签方案 Demo 工程。

2.1.4 Nimble 简介

`NDK 版本基于 Nimble V1.5.0 版本`

Overview

Apache NimBLE 是一个开源的蓝牙 5.1 协议栈（包括主机和控制器），完全取代了 Nordic 芯片上的专有协议栈。它是 Apache Mynewt 项目的一部分。

特点:

- 支持 251 字节数据包长度。
- 支持 4 种角色并发工作：Broadcaster, Observer, Peripheral and Central。
- 支持 32 个连接并发工作。
- 支持 Legacy 和 SC (secure connections) SMP 配对和绑定。
- 支持扩展广播。
- 支持周期广播。
- 支持 Code phy 和 2M phy。
- 支持蓝牙 mesh[NDK 暂未测试对接]。

支持硬件

目前支持 PAN107x 系列芯片，同时 OS 使用 freertos。

概览

如果您在浏览源代码树，并且想要查看一些主要的功能块，这里有一些指引：

- nimble/controller: nrf 的部分 controller 实现，Pan108x 封装于相关 lib 中。
- nimble/drivers: 射频相关收发 (Nordic nRF51 and nRF52)，Pan108x 封装于相关 lib 中。
- nimble/host: 包含主机子系统的代码。这包括 L2CAP 和 ATT 等协议，支持 HCI 命令和事件，通用访问配置文件（GAP），通用属性配置文件（GATT）和安全管理器（SM）。
- nimble/host/mesh: 包含蓝牙 Mesh 子系统的代码。
- nimble/transport: 包含支持主机和控制器之间的传输协议的代码。这包括 UART、emSPI 和 RAM（在主机和控制器在同一 CPU 上运行时使用的组合构建）。
- porting: 包含支持的操作系统的 NimBLE 移植层（NPL）的实现。
- ext: 包含 NimBLE 使用的外部库。如果操作系统没有提供这些库，则会使用它们。

应用示例

还有一些示例应用程序，展示了如何使用 Apache Mynewt NimBLE 协议栈。如下:

- `ble_central`: 蓝牙主机示例，主要用于演示主机连接从机 ANS 报警通知服务. 该示例可以直接和 `bleprph_enc` 完成对测。
- `bleprph_hr`: 蓝牙从机示例，主要演示从机心跳服务.
- `bleprph_enc`: 蓝牙从机加密示例，主要演示从机特定特性读写时会进行加密配对服务. 可以和 `ble_central` 完成连接对测。但是 `ble_central` 并没有访问加密特性，所以不会触发加密配对流程。

API 接口

如果想在线了解 API 可以参考官方提供的 API 接口指南 https://mynewt.apache.org/v1_10_0/network/ble_hs/ble_hs.html 去了解相关接口功能。

当然也可以通过磐启官方 wiki 了解相关使用指南 https://docs.panchip.com/pan1080dk-doc/latest/index.html。

2.2 硬件资料

文档列表:

2.2.1 PAN107x 硬件参考设计

1 概述

本文档主要介绍 PAN107BUA2A 芯片方案的硬件原理图设计、PCB 设计建议、天线设计。本文档提供 PAN107BUA2A 芯片的硬件设计方法。

2 原理图设计建议

2.1 PAN107BUA2A **参考设计原理图** 如上图电路系统由电源去耦电容、DC-DC 降压、晶振电路、天线匹配网络组成。

2.2 电源

- VBAT 为芯片电源脚，要求供电能力不小于 60mA，供电范围为 1.8V–3.6V 。
- VBAT、VCC_RF、VOUT_BK 电源相关引脚需要至少预留 1 个电容，预留一大一小 2 个电容更佳。电容推荐为 4.7uF 和 100nF。
- 电容靠近芯片引脚摆放，电容焊盘和芯片焊盘之间最大距离不超过 5mm。请遵循指导要求，否则易引起 DC-DC 带不起 RF 以及 EVM 异常。

2.2.1 DC-DC DC-DC 芯片外围电路

1. 芯片外围电路组成为：L1、C3，C4。
2. L1 推荐型号：PIM252010-2R2MTS00。选择功率电感，2.2 H，**最小峰值电流为** 150mA，DCR **不超过** 80mΩ，未满足要求在 DC-DC 模式可能会造成 RF 功能异常。
3. DCR 过大会影响 BUCK 效率，能量会转化成热量损耗掉，DC-DC 输出的驱动电流是有限的，效率越低，能够供给到芯片的有效能量就越少。

- DC-DC 的两种工作模式:

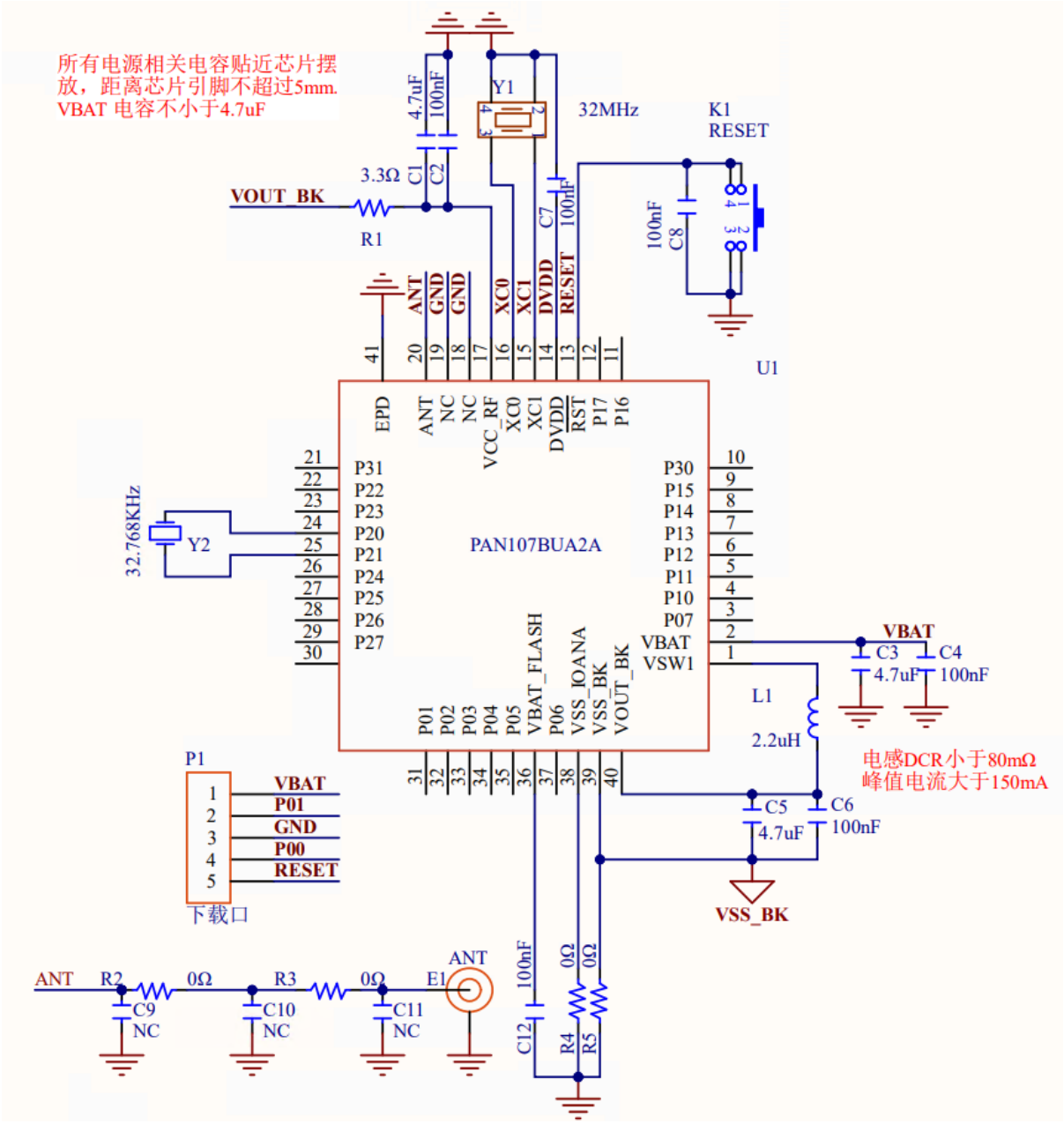


图 4: PAN107BUA2A 最小系统参考设计原理图

1. 开启 DC-DC 模式可以降低系统功耗。
2. 开启 LDO（Bypass）模式后芯片内部将 VBAT 连接到 VSW1，这时 VSW1 处的 2.2uH 电感作用为一段导体，可以用 0Ω 电阻替换。
3. DC-DC、LDO 两种模式不能同时开启。
4. 在不考虑功耗的前提下，可将 VCC_RF 直接连接到 VBAT，此时应将电源模式设置为 LDO 模式。

- DC-DC 相关引脚说明：

1. VBAT 为 DC-DC 的供电引脚。
2. VSW1 为 DC-DC 的功率开关（P-MOS）漏极输出引脚，功率电感应靠近该引脚放置。
3. VOUT_BK 为 DC-DC 的反馈引脚，电容应靠近该引脚放置。
4. VSS_BK 为 DC-DC 电源的 GND 引脚。

2.2.2 DVDD　DVDD 需要**放置** 100nF **电容**。电容最大不应超过 1uF，否则会影响芯片正常启动，电容应靠近该引脚放置。

注意：为避免电路异常，该电容容值请不要随意更改！

2.2.3 VCC_RF　VCC_RF 外部需要接一级 RC 滤波器并尽量靠近该引脚。R1 为 3.3Ω、C1 为 4.7uF，截止频率约为 10KHz。请遵循先 R 后 C，电容摆放位置距离芯片引脚不超过 5mm。

2.2.4 VBAT_FLASH　VABT_FLASH 处预留一个 100nF 的电容，默认焊接。

2.3 晶振

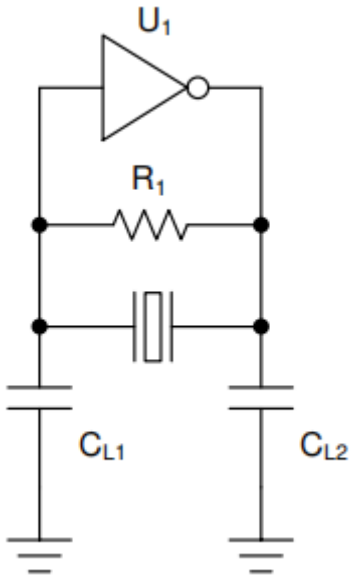


图 5: 32MHz 晶振外围电路示意图

2.3.1 晶振 32Mhz

- 上图振荡器由晶振、反馈电阻 R1、负载电容 CL、放大器构成。可以通过晶体所需负载电容 CL 来调整晶体振荡器频率。
- PAN107x 系列芯片内部集成了电阻 R1、负载电容 CL1、CL2。**只需要焊接一个晶振即可工作**。推荐型号为：X322532MOB4SI 32MHz 12pF ±10ppm 。该晶体下的 RF EVM 参数实测效果较好。

- **芯片内部有可调负载电容，不需要外部焊接电容**。需要软件调整负载电容（值 00-0x2f）。该值在出厂前已经校准好，需要软件导入最佳配置值。2.4G 最大允许频偏为 50khz，频偏越小越好。

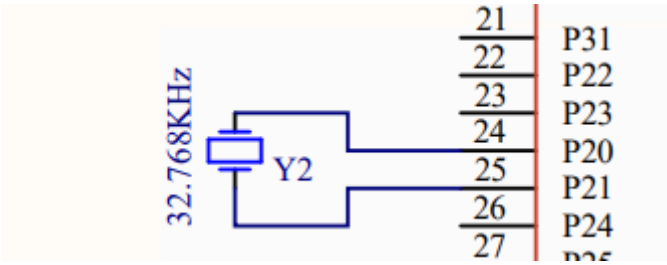


图 6: 32KHz 晶振外围电路原理图

2.3.2 晶振 32.768Khz

- 低速晶振电路支持外部 32.768KHz 无源晶振。**内部有可调负载电容**；低速晶振推荐用户选择 ESR<80KΩ 的晶振。

2.4 复位电路

复位引脚可以悬空，或增加外部按键。在外部按键应用中必须有电容，参数为 100nF。加电容的作用是在系统受到强干扰时，稳定复位脚的电平状态。

> **注意**：该引脚内部有一个 50KΩ 左右的上拉电阻，低电平会使复位生效，为避免电路异常，该电容容值请不要随意更改！

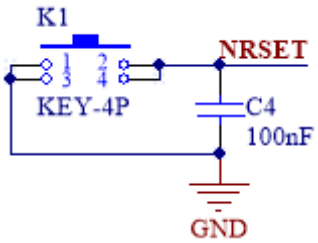


图 7: 复位电路

2.5 静电防护

2.5.1 IO 端静电防护

使用的 IO 要预留串联电阻和 ESD 静电防护元件焊盘位置，便于过认证前的调试整改。串电阻的作用主要是减少 IO 信号的反射、降低外部毛刺信号干扰以及削弱静电对 IO 的影响。**频繁与外部进行数据交换的 IO 例如使用 USB 功能脚等，必须使用 TVS 管进行保护**。建议使用的 TVS 管类型如单向的 ESD5Z3V3 或双向的 CESD923NC5VB，**靠近外设接口位置摆放，**ESD **静电防护元件附近建议保留完整、连续的地，周围打尽量多过孔，有利电荷泄放**。

2.5.2 电源端静电防护

电源输入端 VBAT 请加上静电防护元件，若有静电进入可快速将电荷泄放到地，尽可能避免损坏芯片。ESD **静电防护元件靠近电源输入端接口摆放**。可选择 ESD5Z3V3。

2.5.3 天线端静电防护

无论是板载天线还是其他天线，本质上都是一段长导体，必然有概率吸引到静电电荷，为预防静电打坏芯片 RF，天线端建议加上静电防护元件，**必须使用低容值 (小于** 3pF) **的** TVS **元器件，尽可能不影响** RF **阻抗，如** BTRD04A035。ESD **静电防护元件靠近芯片摆放，若** RF **阻抗受到影响还可通过 匹配调整**。在天线的位置放置一个 ESD 管。推荐使用有馈地点的天线，如 PF 天线。不建议使用导线天线。

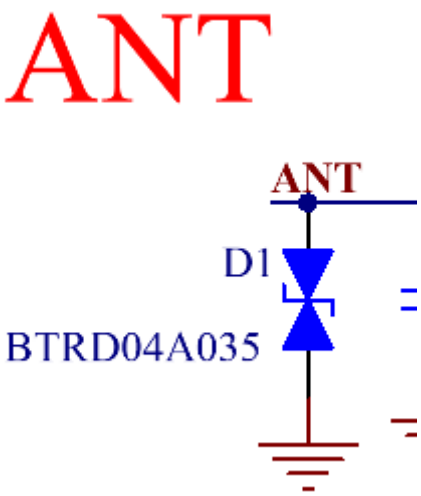


图 8: 天线端静电防护示意图

3 PCB 设计建议

3.1 制版工艺

- 本 Guide 主要针对二层板并且单面贴设计，叠层如下图所示。PCB 具体厚度根据实际情况和阻抗要求适当调整。

	Layer Name	Type	Material	Thickness (mil)	Dielectric Material	Dielectric Constant
	Top Overlay	Overlay				
	Top Solder	Solder Mask/Coverlay	Surface Material	0.4	Solder Resist	3.5
1	Top Layer	Signal	Copper	1.8		
	Dielectric1	Dielectric	None	59.4	FR-4	4.6
2	Bottom Layer	Signal	Copper	1.8		
	Bottom Solder	Solder Mask/Coverlay	Surface Material	0.4	Solder Resist	3.5
	Bottom Overlay	Overlay				

图 9: 制版工艺说明

* 线宽推荐如下：

板材属性	参数
PCB 板材	FR4
PCB 板厚	1.6mm
50 欧姆 RF 线宽	20mil
接地铺铜与 RF 走线间距	5mil

3.2 电源部分注意事项

3.2.1 电源去耦电容布局

- VBAT，VCC_RF, VOUT_BK,DVDD 管脚就近放置电容，走线尽量短粗，如下图绿色框图部分。

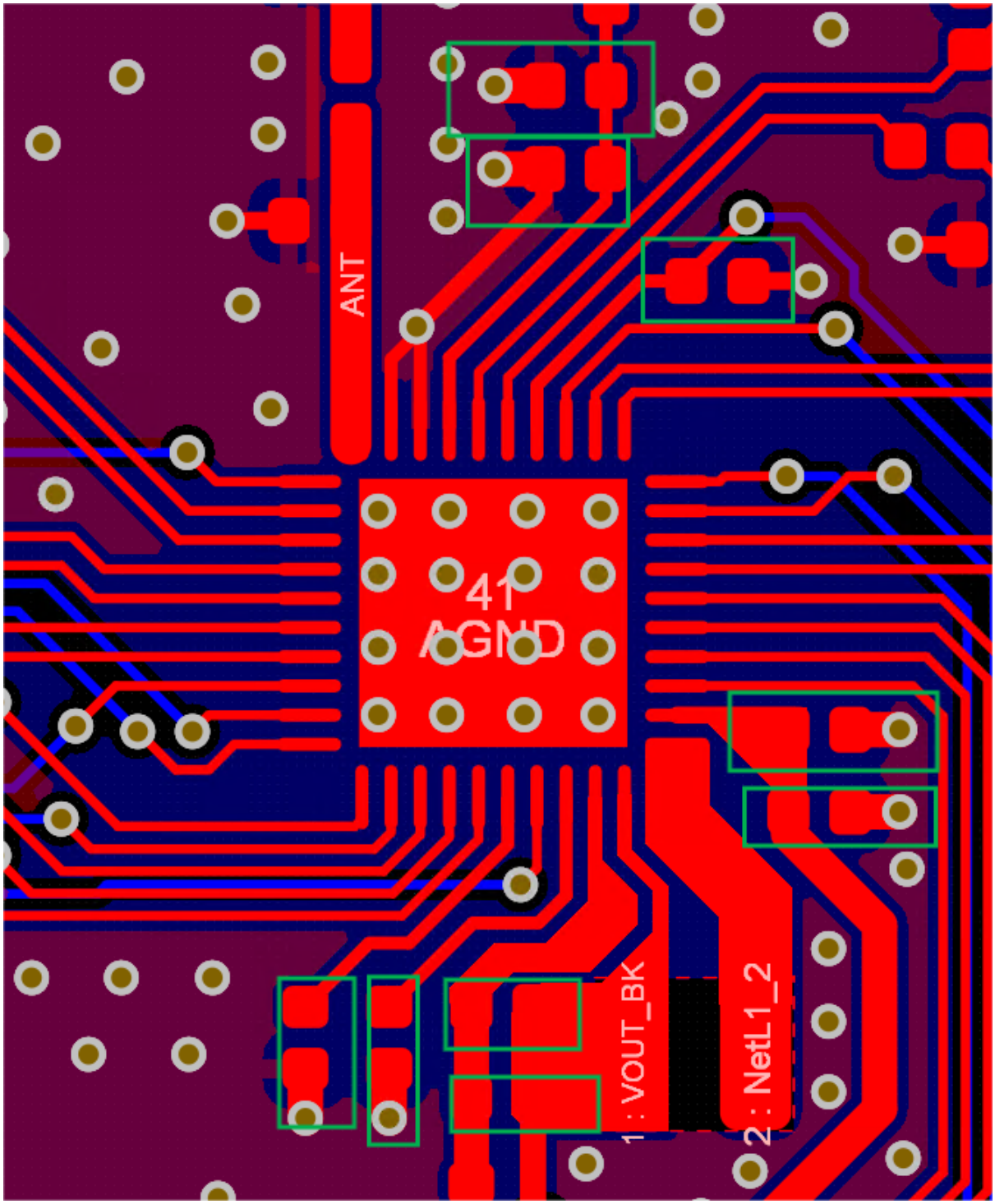


图 10: 电源去耦电容布局

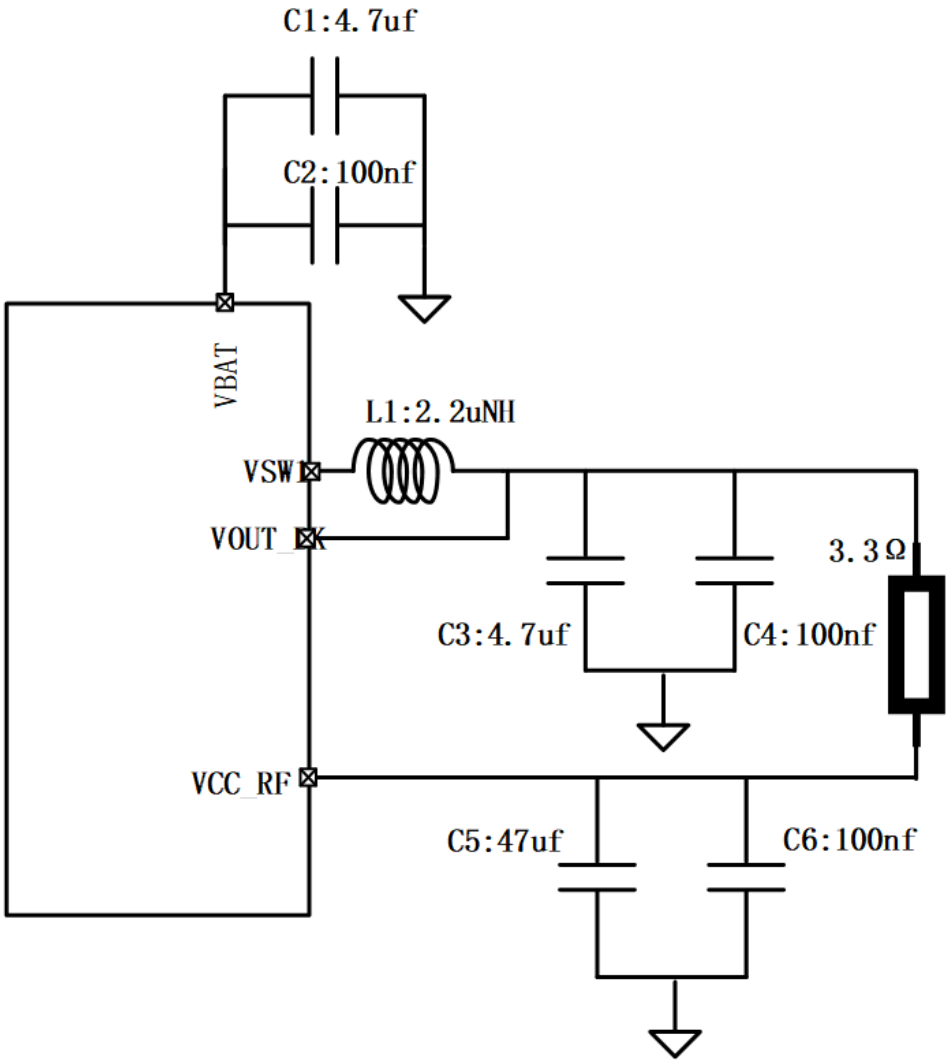


图 11: DC-DC 外围电路

3.2.2 DC-DC 参考设计

- VSW1 管脚与电感 L1 距离尽量的短，且附铜面积尽量大。
- L1 与 C3&C4 之间的走线尽量短，且附铜面积尽量大。
- C1&C2 靠近 VBAT_BK 管脚位置摆放，走线尽量短
- VOUT 与 VCC_RF 之间增加磁珠（或小电阻 3.3Ω 左右），磁珠（或小电阻）靠近 C5&C6 摆放
- 电感 L1 尽量远离晶振 XTH
- 电感四周用 GND 隔离
- DCDC 的地不能与 XTH 共用，DCDC 的地通过 0Ω 电阻和其他地连接。DCDC 的地不能直连到 EPAD。

  总体原则为 DCDC **电流回路路径尽可能短；DCDC 地属于干扰源尽量隔离，避免干扰晶振**。

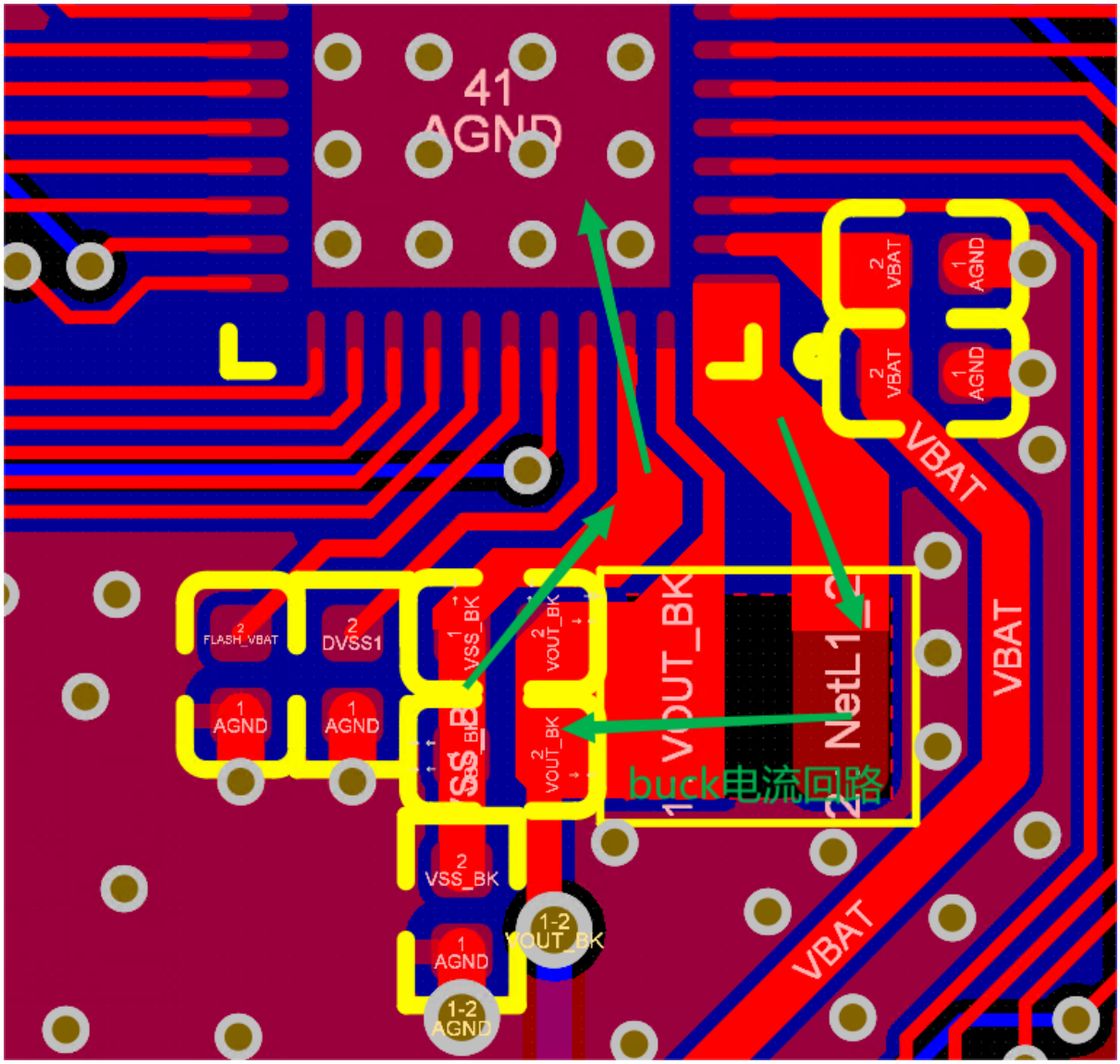


图 12: DC-DC Layout 示意图

3.3 射频走线注意事项

- 晶振尽量靠近芯片引脚摆放。

- 射频匹配链路按照 50Ω 走线，可以参考 TOP 和 BOTTOM 层的 GND 平面，RF 走线尽可能短，RF 线与焊盘宽度一致，天线的 型匹配并联元件焊盘和走线重合为佳。
- RF 线有完整的参考地，从 IC 端出来就进行包地处理，两边打 GND 过孔，底层地平面尽量宽，如**标签** 1 所指信号走线。
- IPEX-1 代天线端子信号引脚挖空，周围包地，尽量减小寄生电容导致阻抗突变，如**标签** 3。
- 芯片底部多打过孔，QFN 封装则打在 E-PAD 上，如**标签** 2。
- 晶振应远离天线，TOP 层挖空，周围包地，降低对电源和 RF 的干扰，需要挖空的部分如**标签** 3。
- 天线辐射区域不要摆放金属器件，净空区挖空处理。

射频链路走线参考如下：

- 天线匹配链路底层不要走线，保证地回路到芯片最短。天线匹配链路的地和芯片 EPAD 是一块完整连续的地。如**标签 "射频地"**。
- 芯片底层不要走线。

射频地线走线如下：

3.4 板载天线

PCB Layout 参考中 MIFA 天线尺寸如图所示。

天线设计尺寸参考

3.5 RF 网络匹配调整

- 长度切割是调整天线的快速方法。但如果客户的样板有足够的空间和成本预算来放置匹配的网络组件以及拥有天线调谐能力，我们还建议使用匹配网络。

1. 要进行 RF 匹配调试，建议在芯片引脚附近、靠近天线辐射端都各加入一个 形匹配，并且在两个 匹配之间串上一个 0Ω 电阻便于调试。如下图 RF **网络匹配原理示意图**。
2. 建议先调远端，先进行天线端匹配调试。断开两个 形匹配之间的串联电阻（如 RF **网络匹配原理示意图**中的**标记** 2），使用如矢量网络分析仪可观测天线阻抗、S11 参数的仪器接入到天线端 形匹配网络（如 RF **网络匹配原理示意图**中的**标记** 3 位置），调试天线端阻抗和 S11 驻波。
3. 之后进行芯片输出功率调试，断开两个 形匹配之间的串联电阻，使用如矢量网络分析仪等可观测天线阻抗、S11 参数的仪器接入到芯片端的 形匹配网络（如 RF **网络匹配原理示意图**中的**标记** 1 位置），调整芯片输出阻抗以及功率。
4. 若没有仪器观测阻抗参数，亦可断开两个 形匹配之间的串联电阻，将频谱仪或其他可以检测 RF 输出功率的仪器接入到芯片端的 形匹配网络，通过检测 2402、2440、2480 三个频点的功率值，来调整芯片的输出功率。
5. 最后还需要测试芯片灵敏度情况，因为发射机和接收机内部匹配不完全相等，所以当发射功率调好以后，最后需要确认接收灵敏度情况，如果发射功率调整好后发生接收机灵敏度下降，联系我司工程师修改内部匹配。

   注意：量产前一定要进行距离测试!

4 BOM

最小系统 BOM **参考下表，所有** PAN108x **系列通用：**

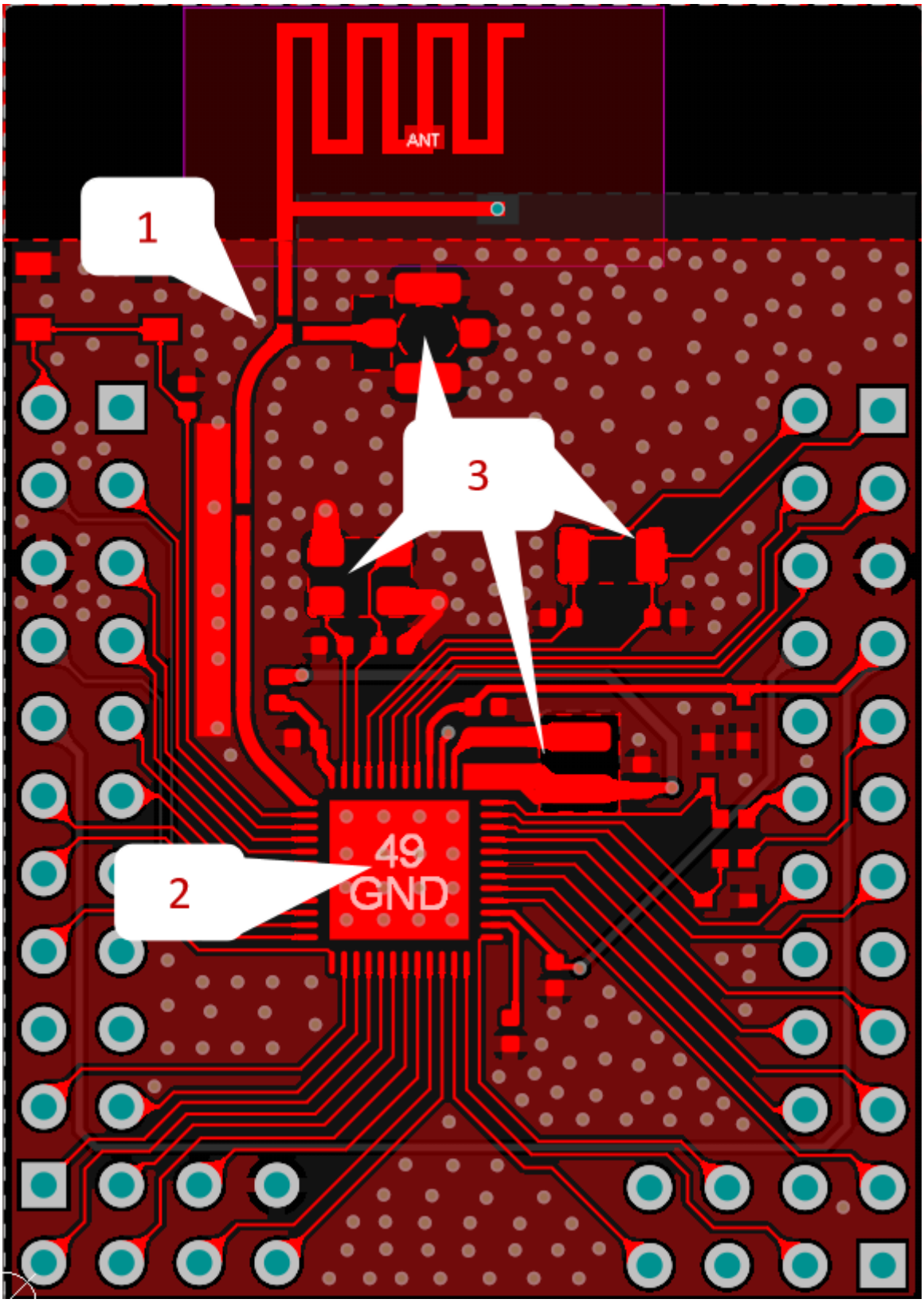


图 13: 射频链路走线示意图

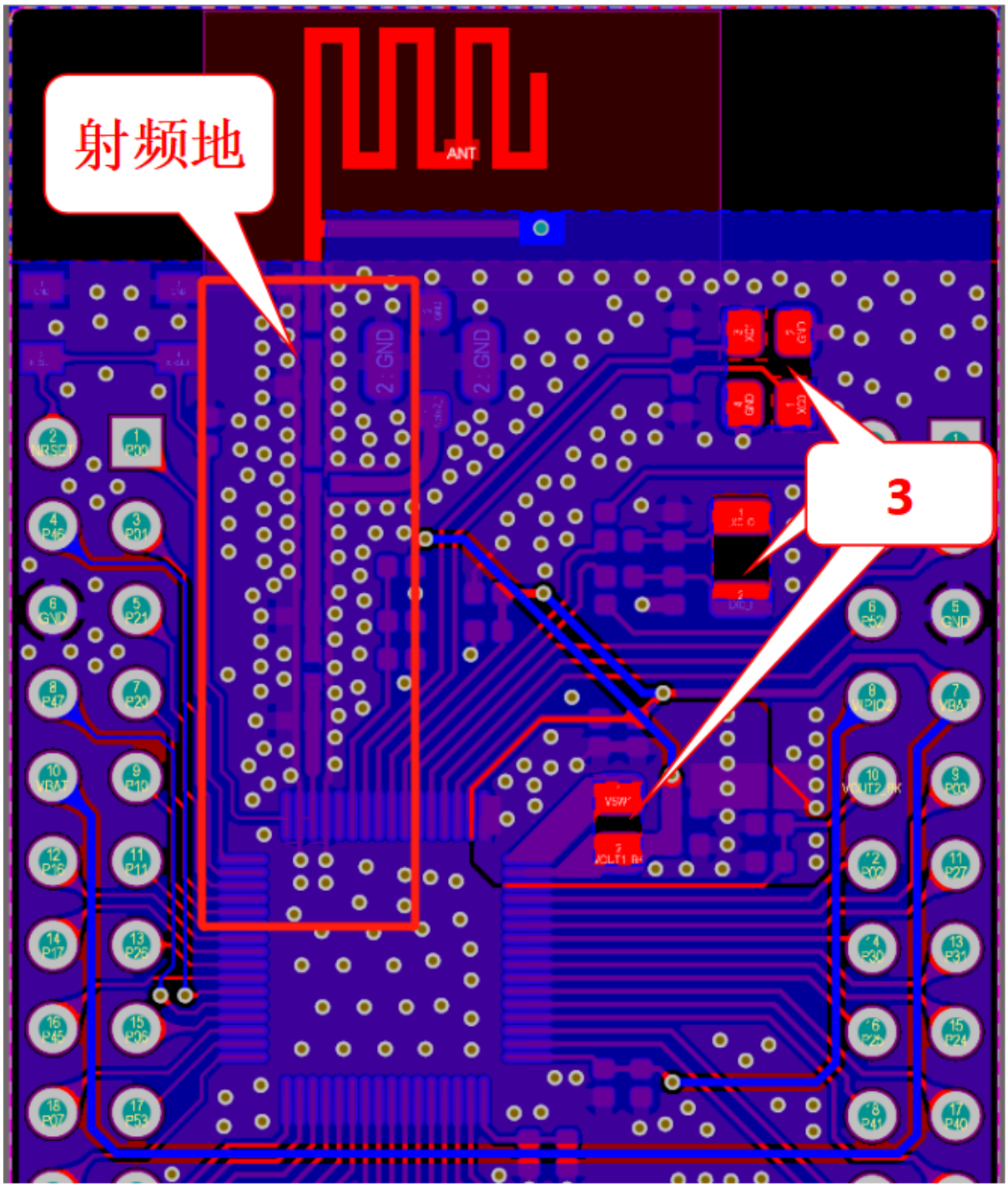


图 14: 射频地线走线示意图

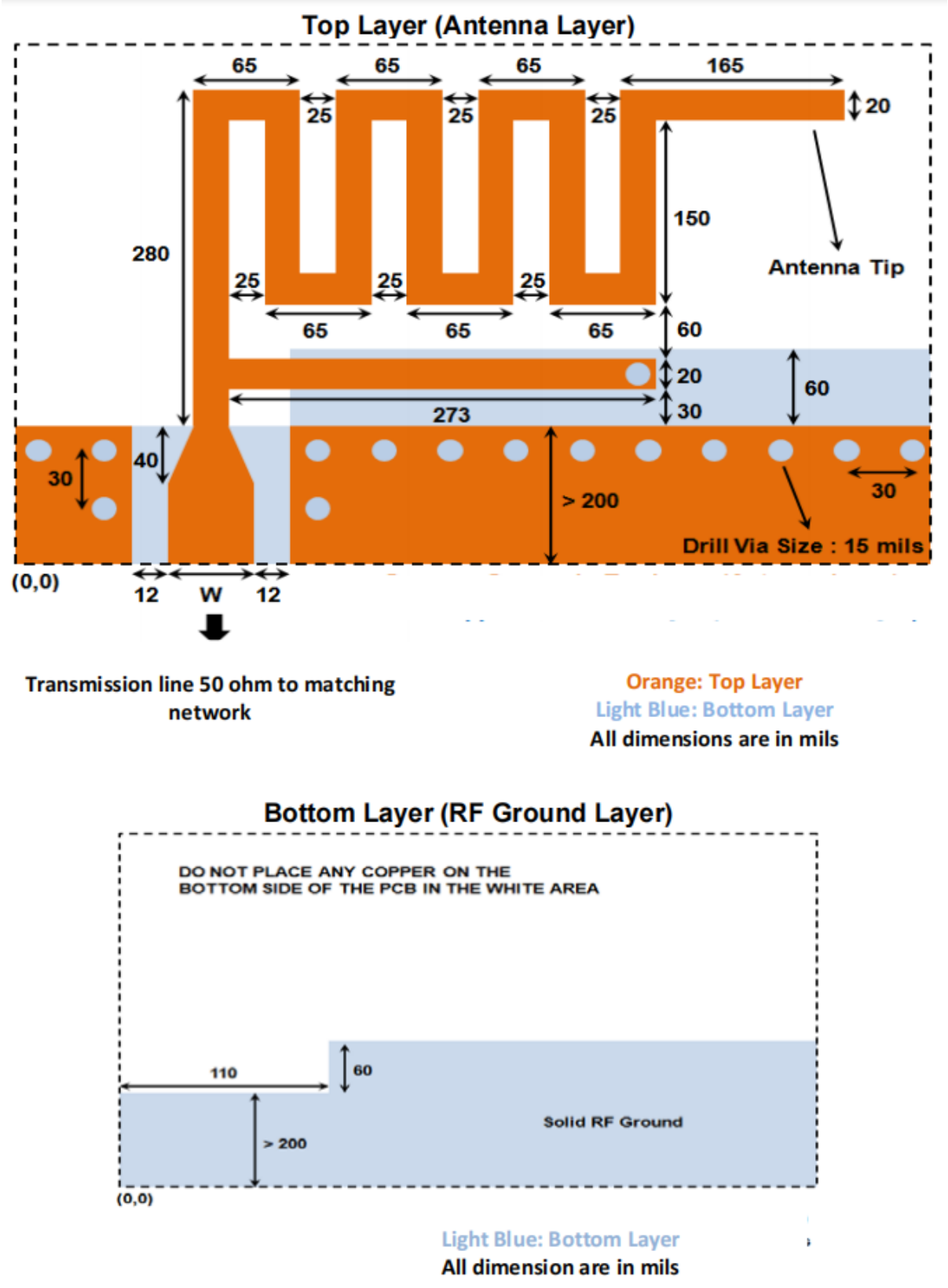
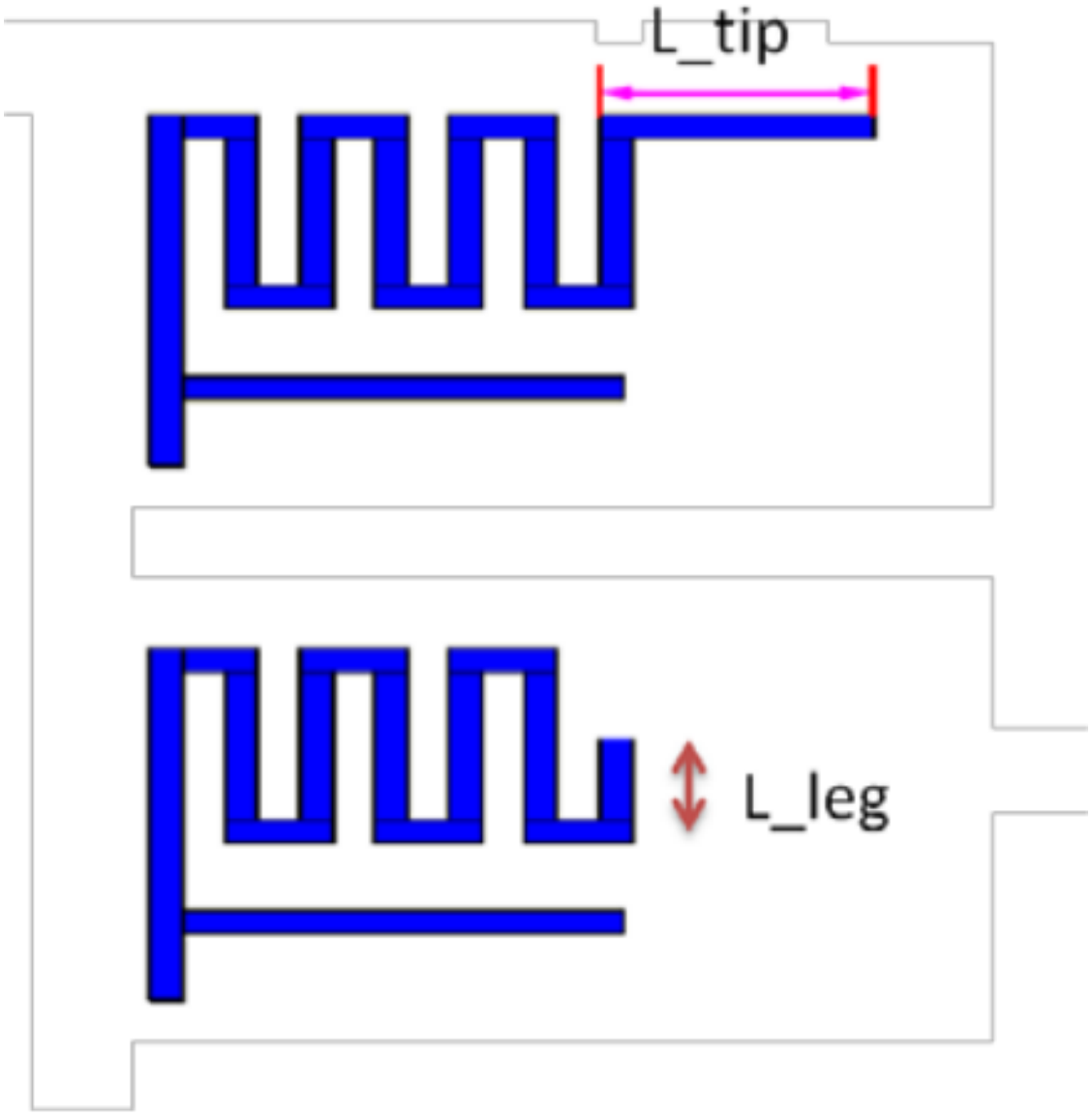


图 15: 天线设计尺寸参考示意图



PCB Thickness	Antenna L_Tip / L_leg
16 mils	L_tip= 353 Mils
31 mils	L_tip= 165 Mils
47 mils	L_tip= 125 Mils
62 mils	L_leg= 115 Mils

图 16: 天线设计匹配参考示意图

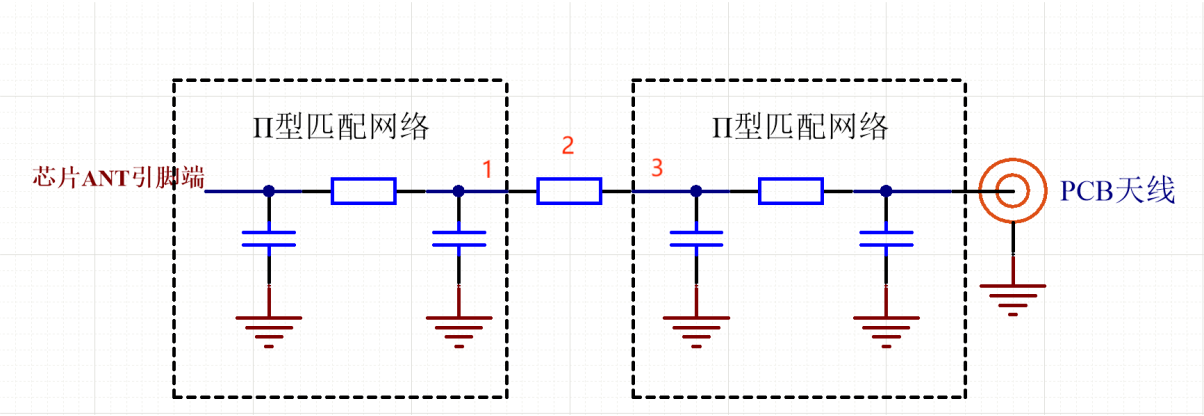


图 17: RF 网络匹配原理示意图

品种	参数	型号	品牌	立创编号	位号	封装	数量
贴片陶瓷电容	4.7uF	0402X475M6R3NT	广东风华高新科技股份有限公司	C168172	C1, C3, C5	0402_C	4
贴片陶瓷电容	100nF	0402B104K160NT	广东风华高新科技股份有限公司	C41851	C2,C4,C6,C7,C8	0402_C	1
贴片按键	4P-4.2mm x 3.25mm	K2-1808SN-A4SW-01	韩荣电子有限公司	C92589	K1	SW-SMD(4.2x3.25x2.5)	1
贴片功率电感	2.2uH	PIM252010-2R2MTS00	广东风华高新科技股份有限公司	C298679	L1	SMD-2520-1.0	1
贴片电阻	3.3Ω 1%	RC-02U3R30FT	广东风华高新科技股份有限公司	C321181	R1	0402_R	1
贴片电阻	0Ω 1%	RC-02000FT	广东风华高新科技股份有限公司	C140225	R2,R3,R4,R5	0402_R	2
贴片 4 脚晶振	32MHz 10ppm 12pF	X322532MOB4SI	深圳扬兴科技有限公司	C9009	Y1	SMD-3225_4P	1
贴片 2 脚晶振	32.768KHz 12.5pF	X321532768KGB2SI	深圳扬兴科技有限公司	C620155	Y2	SMD-3215_2P FC - 135	1
贴片 IC	PAN107BUA2A	\	上海磐启微电子有限公司	\	U1	QFN40	1

HDK **内容**:

硬件开发资料（HDK）位于: `<PAN107X-ZDK>\02_HDK`，其包含如下内容:

02_HDK 子目录	包含内容
PAN107MP_QFN40	PAN107B 芯片测试板硬件设计资料（原理图、PCB 文件等）和生产资料（BOM、gerber、坐标等文件）

2.3 NDK 演示例程

2.3.1 例程介绍

解决方案

源码路径: `<PAN1070-NDK>\01_SDK\nimble\samples\solutions`

- Electronic Shelf Label: 电子货架标签方案演示 Demo。

MCU Keil 例程

源码路径: `<PAN1070-NDK>\03_MCU\mcu_samples`

MCU 底层驱动（Low Level Driver）Keil 例程:

例程	说明
MCU Low Level ADC Driver Sample	MCU 底层 ADC 驱动例程演示说明
MCU Low Level CLKTRIM Driver Sample	MCU 底层 CLKTRIM 驱动例程演示说明
MCU Low Level CLK Driver Sample	MCU 底层 CLK 驱动例程演示说明
MCU DebugProtect Driver Sample	MCU Debug Protect 机制演示例程说明
MCU Low Level DMA Driver Sample	MCU 底层 DMA 驱动例程演示说明
MCU Low Level eFuse Driver Sample	MCU 底层 eFuse 驱动例程演示说明
MCU Low Level FMC Driver Sample	MCU 底层 FMC 驱动例程演示说明
MCU Low Level GPIO Driver Sample	MCU 底层 GPIO 驱动例程演示说明
MCU Low Level I2C Driver Sample	MCU 底层 I2C 驱动例程演示说明
MCU Panchip Private RF B250K Rx Sample	MCU 私有 2.4G 驱动 B250K Rx 例程演示说明
MCU Panchip Private RF B250K Tx Sample	MCU 私有 2.4G 驱动 B250K Tx 例程演示说明
MCU Low Level PWM Sample	MCU 底层 PWM 驱动例程演示说明
MCU Low Level SPI Sample	MCU 底层 SPI 驱动例程演示说明
MCU Low Level TIMER Sample	MCU 底层 TIMER 驱动例程演示说明
MCU Low Level UART Sample	MCU 底层 UART 驱动例程演示说明
MCU Low Level WDT Sample	MCU 底层 WDT 驱动例程演示说明
MCU Low Level WWDT Sample	MCU 底层 WWDT 驱动例程演示说明

2.3.2 例程列表

Solution: Electronic Shelf Label

1 功能概述 此 sample 为 `pan107x`(40pin 芯片) 在电子价签板下的应用。

具体支持的 feature 如下:

1. 外挂 spi flash: 外挂 flash 存储价签图案数据，每隔 45s 通过 dma 方式从 flash 读取一个图案
2. epd 墨水屏: 外挂 flash 读取的数据通过 3 线 spi 传输给墨水屏，启动墨水屏刷屏
3. 低功耗模式: 刷图完成后芯片及 epd 均进入休眠模式，模块进入超低功耗模式（standby），定时 15s 唤醒
4. RF 发送: 芯片唤醒进入 rf 发送流程
5. 每 3 次发送完成后，启动刷屏流程，1~4 步骤重复

2 环境要求

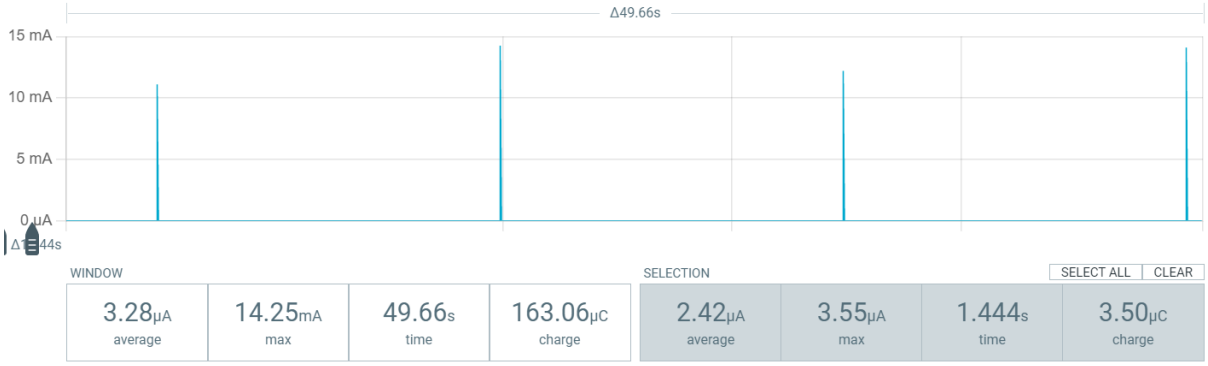
- board: ‘pan107x 40pin esl 价签板
- 外挂 flash、墨水屏
- 电流监测工具 nrf ppk

3 编译和烧录 例程位置: `nimble\pan107x_samples\solutions\esl`

使用 keil 工具可以对其进行编译、烧录、调试等操作。

4 演示说明

1. 准备 esl 价签板，107/FM/EPD 跳线帽短接
2. 插入 epd2266 墨水屏（SE2266JS0C5）
3. 打开 PPK 并使用其供电 3.3v
4. 观测 PPK 电流变化及墨水屏刷屏过程（45s 刷屏一次），电流每 15s 进入低功耗



5 主要数据结构说明

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，包括普通型和增强型
prf_chip_mode_sel_t	chip_mode	xn297 通信协议和 nordic 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 和 2M
prf_crc_sel_t	crc	数据包 CRC 配置，可配置为 crc 16bit，crc 8bit，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
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 (2402Mhz~2480Mhz)
uint8_t	tx_no_ack	配置增强型模式下 tx 是否需要 ack
prf_trf_t	trf_type	nordic 的长包模式配置，最大 payload 的长度为 255
uint8_t	rx_length	rx 接收数据包长度配置，增强型模式下可不配置
uint8_t	sync_length	接入地址长度配置，可配置为 3、4、5 字节
uint8_t	sync[5]	接入地址的内容 (xn297 模式下可白化地址，防止出现长 0 和长 1 的地址)
prf_dev_sel_t	dev	设置 deviation，可以选择 BLE 模式（1M 250k；2M 500k），NRF 模式（1M 160K；2M 320）
int8_t	tx_power	设置发射功率，范围 (-45dbm~7dbm)
uint8_t	pid_manual_flag	pid 手动配置的标志，使能后可以自定义 pid
uint8_t	crc_include_sync	crc 计算包含地址
uint8_t	src_include_sync	白化包含地址
uint16_t	tx_trans_time	发送传输时间设置
uint16_t	rx_trans_time	接收传输时间设置
prf_pipe_t	pipe	管道配置，可配置为 0~7

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	普通型 M1 模式

prf_chip_mode_sel_t：

Type	Value	Description
PRF_CHIP_MODE_SEL_BLE	1	蓝牙模式
PRF_CHIP_MODE_SEL_XN297	2	XN297 模式
PRF_CHIP_MODE_SEL_NORDIC	3	NORCDIC 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 模式
PRF_PHY_CODED	4	S2 模式
PRF_PHY_250K	5	250K 模式

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_dev_sel_t:

Type	Value	Description
PRF_DEV_NRF	1	NRF 模式 deviation 配置，1M 170k，2M 340K
PRF_DEV_BLE	2	NRF 模式 deviation 配置，1M 250k，2M 500K

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_PIPE3	1«3	管道 3
PRF_PIPE4	1«4	管道 4
PRF_PIPE5	1«5	管道 5
PRF_PIPE6	1«6	管道 6
PRF_PIPE7	1«7	管道 7

prf_trf_t：

Type	Value	Description
PRF_TRF_NORMAL	0	普通模式传输
PRF_TRF_NRF52	1	NRF 模式传输
PRF_TRF_BOE	2	250k 模式传输

6 补充说明 补充说明当前功耗测试情况，支持中遇到的问题（供参考）及已知仍可能存在的问题

6.1 功耗说明

2.4 NDK 开发指南

2.4.1 NDK 低功耗开发指南

本文主要通过一些示例，介绍**低功耗**各个模式、使用的方法以及可能遇到的问题。

1 低功耗模式

低功耗模式介绍如下表所示：

模式名称	进入	唤醒	1.2V 区时钟	时钟	1.2V 供电
STANDBY_M0	Sleep_mode = 3, Buck_en_ctrl=0, Buck_bp_ctrl=0, Flashldo_bp_en_ctrl =0, Flashldo_lp_en_en_ctrl =0, WFI	P00, P01, P02, BOD/LVR (可选, 需保证慢时钟开启), PIN RESET	全部关闭	全部关闭	断电
STANDBY_M1	Sleep_mode = 2, ldo_pwr_ctrl = 0, ldol_ pwr ctrl = 0/1, cpu pwr ctrl = 0/1, sram0/1 pwr ctrl = 0/1, ll_ram pwr ctrl=0/1, phy_ram pwr _ctrl=0/1, Buck_en_ctrl=0, Buck_bp_ctrl=0, Flashldo_bp_en_ctrl =0, Flashldo_lp_en_en_ctrl =0, WFI	所有 GPIO (边沿去抖), SLPTMR, WDT, BOD/LVR (可选), PIN RESET	CLK32K, 其他全部关闭	慢时钟	LPLDOL/H : LL_RAM (可选), PHY_RAM/PHY_REGS (可选), SRAM0/1 (可选), decrypt_ram(可选, cpu 模块不保电, 没办法做到只保少部分寄存器) LPLDOL/H: asnactrl_l (rcc 的寄存器) GPIOWDTBOD, LVR
DEEPSLEEP	Sleep_mode = 1, ldol_power_ctrl = 0/1, Buck_en_ctrl=0, Buck_bp_ctrl=0, Flashldo_bp_en_ctrl =0/1, Flashldo_lp_en_en_ctrl =1/0, WFI	所有 GPIO, SLPTMR, WDT, TIMER0/1/2, BOD/LVR (可选), PIN RESET	CLK32K, 其他全部关闭	慢时钟	LPLDOL/H: 其他数字模块, LPLDOL/H: WakeupGPIO, WDT, Timer0/1/2, (需要测试, 确认如何安全使用)
SLEEP	Sleep_mode = 0, WFI	所有外设中断, BOD/LVR (可选), PIN RESET	CLK32K, CPU_CLK 关闭, RCH、XTH、DPLL 根据软件配置选择打开	慢时钟快时钟	HP_LDO 供电

2 开发流程

2.1 低功耗使用流程

2.1.1 Sleep 模式

- **进入流程:**

1. 配置 sleep_mode 为 sleep 模式
2. 唤醒源配置
3. 设置 flash dp_en 为 0
4. 设置 CPU SLEEPDEEP 寄存器为 0; 地址: 0xE000ED10
5. 考虑安全, 建议进行一次手动 3V 同步操作

6. ___WFI();

- **退出流程：**

1. 唤醒源产生中断；
2. 处理中断，清除中断源；

- **备注：**

1. 支持 m0 调试模式，不支持 riscv 调试模式

2.1.2 Deepsleep 模式 进入流程：

1. 配置 sleep_mode 为 deepsleep 模式，配置各电压域的 power switch，配置 rcl_en_ctrl，xtl_pwr_ctrl，Buck 和 flashldo 的配置建议使用 driver 默认

2.

Power Switch	模式 1（推荐）	模式 2	模式 3
lpldoh_en	1	1	0
lpldol_en	1	0	1
Ldo_pwr_ctrl	1	1	1
Ldol_pwr_ctrl	0	1	1
Peri_pwr_ctrl(cpu/ll_sram/phy_sram/sram0/sram1)	1（ram 可配）	1（ram 可配）	1（ram 可配）
Lpldoh_iso_en	1（配置 0 待测试）	0	0

3. 唤醒源配置：所有 GPIO，SLPTMR，WDT，TIMER0/1/2，BOD/LVR（可选），PIN RESET。对于模式 1，如果 Lpldoh_iso_en 配置为 1，则不支持 TIMER0/1/2 唤醒；如果 Lpldoh_iso_en 配置为 0，则支持 TIMER0/1/2 唤醒和 PWM 输出（需要测试是否有漏电）。对于模式 2 或者模式 3，上述唤醒源都可以唤醒系统。BOD，LVR 唤醒需要开启 32K 时钟。
4. 对于模式 1，如果 Lpldoh_iso_en 配置为 1，不支持 PWM 输出；如果 Lpldoh_iso_en 配置为 0，则支持 PWM 输出（需要测试是否有漏电）。对于模式 2 或者模式 3，PWM 可以输出
5. Flash dp 设置为 1，并退出 enhance 模式。配置合适的 dp，和 rdp 时间
6. 建议 cpu 地址重映射功能开启，映射地址为 ram 保电区域，可加快唤醒后，程序执行速度
7. Dly_time2 需要根据测试结果重新配置，默认值比较大
8. 设置 CPU SLEEPDEEP 寄存器为 1；地址：0xE000ED10
9. Flash 控制器退出增强型模式；
10. 考虑安全，建议进行一次手动 3V 同步操作
11. _WFI();

退出流程：

1. 唤醒源唤醒，产生 lp 中断或者唤醒源相对应的中断
2. 清除相应 flag

备注：

1. 支持 m0 调试模式（低功耗期间会丢失），不支持 riscv 调试模式
2. gpio 唤醒电平，至少需要保持一个完整的 32K 时钟周期。如果需要读取的 gpio 的中断，需要 7 个 32K 时钟周期（需要 dly2 的延时决定）；如果 32K 时钟关闭，则唤醒需要的时间更久，和 32K 时钟的启动时间相关

2.1.3 Standby_m1 模式 进入流程：

1. 配置 sleep_mode 为 standby_m1 模式，配置各电压域的 power switch，配置 rcl_en_ctrl，xtl_pwr_ctrl，Buck 和 flashldo 的配置建议使用 driver 默认

2.

	模式 1（推荐，需测试）	模式 2	模式 3
lpldoh_en	1	1	0
lpldol_en	1	0	1
Ldo_pwr_ctrl	0	0	0
Ldol_pwr_ctrl	0	1	1
Peri_pwr_ctrl(cpu/ll_sram/phy_sram/sram0/sram1)	1(可配)	1(可配)	1(可配)
Lpldoh_iso_en	1	1	1

3. 唤醒源配置：所有 GPIO，SLPTMR，WDT，BOD/LVR（可选），PIN RESET。BOD，LVR 唤醒需要开启 32K 时钟。
4. flash 如果使用 4 线模式，建议开启 dp 模式，flash 两线切换四线的时间特别长，一般会有 8ms；如果 flash 使用 2 线模式，不建议开启 dp 模式，flash 直接掉电处理。
5. 建议 cpu 地址重映射功能开启，映射地址为 ram 保电区域，可加快唤醒后，程序执行速度
6. 根据需求，决定是否开启 cpu 保电功能，寄存器 LP_FL_CTRL[4]。可硬件恢复现场，代码实现有特定需求，参见说明部分
7. Dly_time2 需要根据测试结果重新配置，默认值比较大
8. 设置 CPU SLEEPDEEP 寄存器为 1；地址：0xE000ED10
9. 考虑安全，建议进行一次手动 3V 同步操作
10. __WFI();

退出流程：

1. 唤醒源唤醒，产生 lp 中断以及唤醒源相对应的中断
2. 清除相应 flag

备注：

1. 不支持 m0 和 riscv 调试模式
2. 支持 m0 的 cpu retention 功能（现场恢复），不支持 riscv 的 cpu retention 功能
3. gpio 唤醒电平，至少需要保持一个完整的 32K 时钟周期。如果需要读取的 gpio 的中断，需要 7 个 32K 时钟周期（需要 dly2 的延时决定）；如果 32K 时钟关闭，则唤醒需要的时间更久，和 32K 时钟的启动时间相关

2.1.4 Standby_m0 模式 进入流程：

1. 配置 sleep_mode 为 standby_m0 模式，配置 rcl_en_ctrl，xtl_pwr_ctrl
2. 唤醒源配置：所有 P00，P01，P02，BOD/LVR（可选），PIN RESET。
3. Flash dp 设置为 0
4. Dly_time1 根据需要决定是否配置，如果不在意 m0 的唤醒时间不建议去修改。此处的时间测试遍历会比较多，设置的值不好控制
5. 设置 CPU SLEEPDEEP 寄存器为 1；地址：0xE000ED10
6. 考虑安全，建议进行一次手动 3V 同步操作
7. __WFI();

退出流程：

1. 唤醒源唤醒，产生 lp 中断以及唤醒源相对应的中断
2. 清除相应 flag

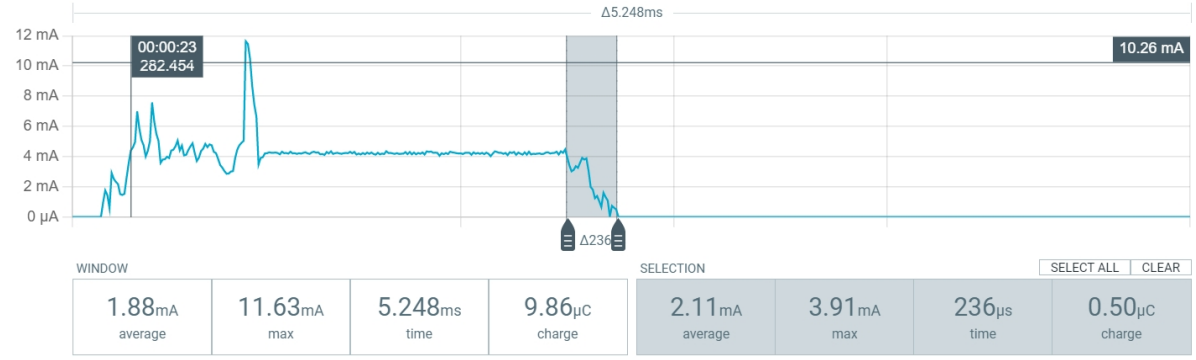
2.2 **参考相关例程** 当前 SDK 中提供了一些低功耗相关的例程，涵盖了章节 1 所述的所有低功耗模式等。

例程位置：03_MCU\mcu_samples\LP。

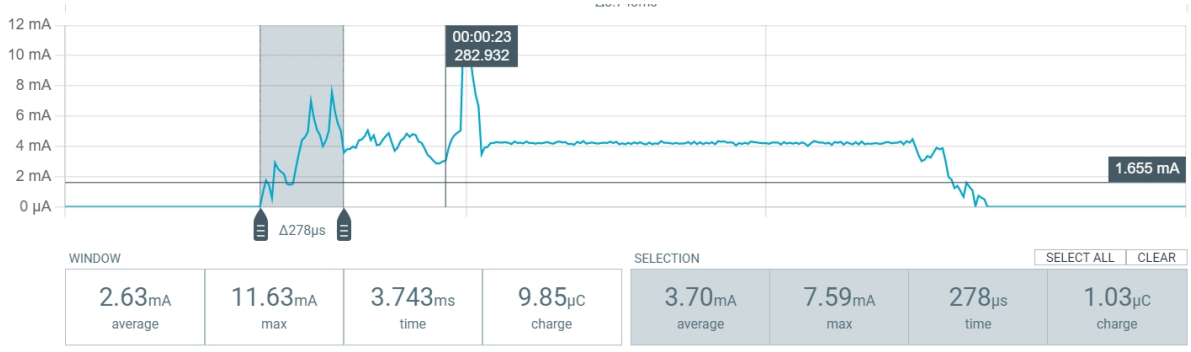
2.3 Standby_m1 **休眠唤醒** 以 standby m1 模式，cpu retention and cpu continue run 模式为基础介绍各个时间阶段 mcu 的动作。

阶段	说明	时 间 (us)	备注
进入休眠	从软件发送休眠指令至硬件完全休眠的时间	236	
硬件唤醒启动	唤醒源触发后硬件完全启动的时间	278	
Standby M1 retention 模式	continue run，唤醒至 rx ready 时间	465	此模式下**软件初始化和** RF **初始化步骤**可以省略
TX/RX(max 59B)	收发最大 payload 的时间，根据传输速率与字节数计算得出	1888	此处按最大数据计算
总计	休眠唤醒至 tx 完成/rx 完成的时间	2867	

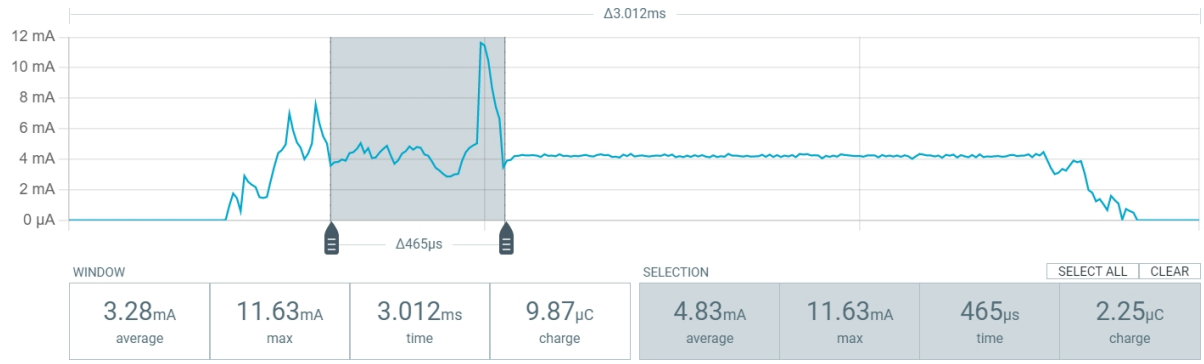
休眠时间



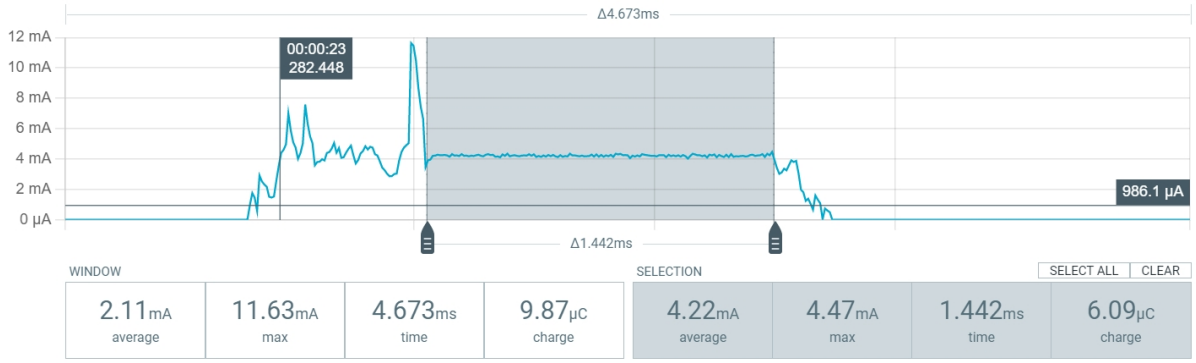
唤醒时间



软件运行至 RF ready 时间



RF 接收 32Byte 时间



3 低功耗注意事项

1. flash 运行在 4 线 enhance 模式，进入功耗前需要退出 enhance 模式。
2. 低功耗模式下供电分两部分 LPLDOL 和 LPLDOH，其中 LPLDOL 给 sram 供电，电压范围从 0.4~0.9v（未校准芯片有差异），LPLDOH 给 always on 区域部分供电，供电范围 0.5~1.2v（未校准芯片有差异），通常在进入低功耗在保证唤醒正常情况下尽量降低两个电压，常温下 LPLDOL/H 可设置未 0/1。
3. 为防止 LPLDOH 在电源抖动时出现不能唤醒的情况，增加了一个 LPLDOH_VREF_TRIM_AON（LP_LPLDO[23:21]）控制位，设置值的作用是弥补电源抖动，稳定电压，同时设置完成此值（例如设置为 2，LPLDOH 电压 0.7v）后再想拉低 LPLDOH 至 0.6v 是不能完成的，此属于正常现象。
4. DeepSleep 模式下外设 timer0/1/2 作为唤醒以及 PWM 在低功耗下输出需要将 deepsleep 低功耗模式设置为模式 2，即 dp_mode= LP_DEEPSLEEP_MODE2.
5. Standby M1 cpu retention 模式唤醒后外设部分及 RF MAC 层寄存器需要重新初始化，PHY、保电的 sram、时钟等不需要重新初始化

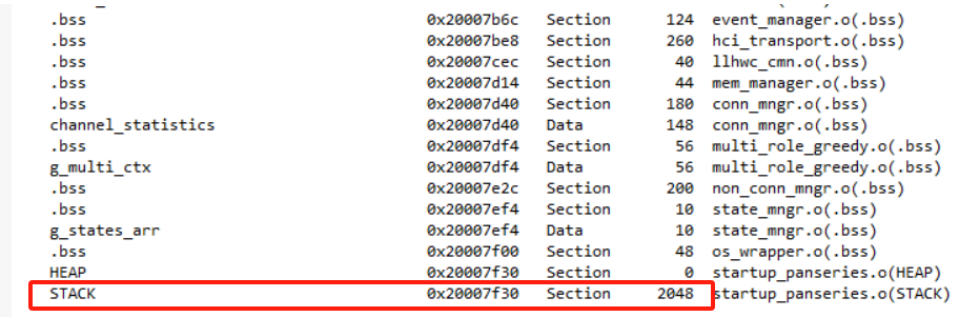
2.4.2 NDK RAM 使用情况分析以及优化指南

1 如何查看 KEIL 的 RAM 和 Flash 使用情况

因为当前工程中很多和 BLE 时间处理相关的函数为了提升处理速度使用了 RAM CODE，所以导致部分 CODE 会占用 RAM 空间，所以查看工程的实际占用空间要从 RAM 和 Flash 空间进行查看。

首先双击 keil 项目打开 map 文件

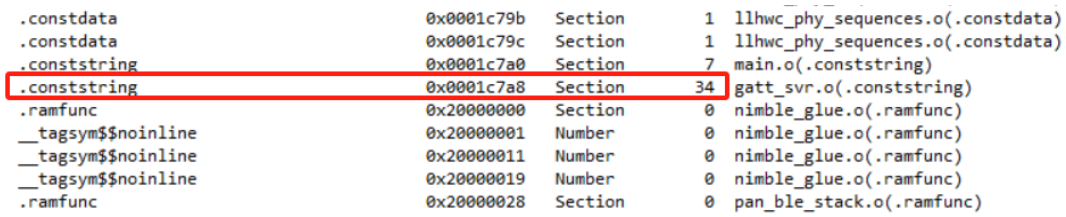
1.1 **如何查看 RAM 空间：** 因为 STACK 占用 RAM 边界位置，我们可以通过 STACK 占用可以知道 RAM 最多占用 `0x20007f30+0x800(2048) = 0x20008730` 的 RAM，因为 RAM 地址默认是 `0x20000000` 起始的，所以我们知道 RAM 占用了 `0x8730`（34608）字节的 RAM。



.bss	0x20007b6c	Section	124	event_manager.o(.bss)
.bss	0x20007be8	Section	260	hci_transport.o(.bss)
.bss	0x20007cec	Section	40	llhwc_cmn.o(.bss)
.bss	0x20007d14	Section	44	mem_manager.o(.bss)
.bss	0x20007d40	Section	180	conn_mngr.o(.bss)
channel_statistics	0x20007d40	Data	148	conn_mngr.o(.bss)
.bss	0x20007df4	Section	56	multi_role_greedy.o(.bss)
g_multi_ctx	0x20007df4	Data	56	multi_role_greedy.o(.bss)
.bss	0x20007e2c	Section	200	non_conn_mngr.o(.bss)
.bss	0x20007ef4	Section	10	state_mngr.o(.bss)
g_states_arr	0x20007ef4	Data	10	state_mngr.o(.bss)
.bss	0x20007f00	Section	48	os_wrapper.o(.bss)
HEAP	0x20007f30	Section	0	startup_panseries.o(HEAP)
STACK	0x20007f30	Section	2048	startup_panseries.o(STACK)

图 18: 通过 STACK 查看 RAM 占用

1.2 **如何查看 Flash 空间** 从 RAM 空间往上查找 flash 边界位置，0x20000000 前面的即是 flash 最大地址



.constdata	0x0001c79b	Section	1	llhwc_phy_sequences.o(.constdata)
.constdata	0x0001c79c	Section	1	llhwc_phy_sequences.o(.constdata)
.conststring	0x0001c7a0	Section	7	main.o(.conststring)
.conststring	0x0001c7a8	Section	34	gatt_svr.o(.conststring)
.ramfunc	0x20000000	Section	0	nimble_glue.o(.ramfunc)
__tagsym$$noinline	0x20000001	Number	0	nimble_glue.o(.ramfunc)
__tagsym$$noinline	0x20000011	Number	0	nimble_glue.o(.ramfunc)
__tagsym$$noinline	0x20000019	Number	0	nimble_glue.o(.ramfunc)
.ramfunc	0x20000028	Section	0	pan_ble_stack.o(.ramfunc)

图 19: 查看 flash 边界

通过 map 文件找到边界地址 0x1c7a8+0x22(34) = 0x1c7ca(116682) bytes。

2 关于堆空间的使用说明

2.1 **蓝牙 controller 的堆** 蓝牙 controller 的堆默认是用于 controller 的广播，连接等各种数据包动态分配使用的。而且是从 host 定义来进行分配的，但是不同的参数可能导致堆的需求空间不一致。

我们可以通过打开 `app_config.h` 或者 `app_config_spark.h` 中的 `BT controller Memory Pool usage print` 选项（对应的宏 `CONFIG_CNTRL_MEM_POOL_PRINT`）显示的输出底层 controller 所需要的内存。

正常分配时如下：

```
[19:32:53.628]收 ←LL Controller Version:bd5923c

[19:32:53.665]收 ←BT controller memory pool used: 400 bytes, remain bytes: 8496, total:8896
BT controller memory pool used: 764 bytes, remain bytes: 8132, total:8896
BT controller memory pool used: 3784 bytes, remain bytes: 5112, total:8896
BT controller memory pool used: 4840 bytes, remain bytes: 4056, total:8896
BT controller memory pool used: 5164 bytes, remain bytes: 3732, total:8896
BT controller memory pool used: 6124 bytes, remain bytes: 2772, total:8896
BT controller memory pool used: 8896 bytes, remain bytes: 0, total:8896
app started
```

上面的 log 显示正常分配的对内存为 8896bytes，所以我们可以打开 `nimble_glue.c` 或者 `nimble_glue_spark.c` 找到堆分配的宏 `PAN_BLE_CTLR_BUFFER_ALLOC` 将其的值修改为 8896。

```
#define PAN_BLE_CTLR_BUFFER_ALLOC        (8896)
#define PAN_BLE_CTLR_BUFFER_SIZE        (((PAN_BLE_CTLR_BUFFER_ALLOC) + 3)& (~((uint32_
↪t)0x03))) /*4 bytes aligned*/

static uint32_t mem_buffer[PAN_BLE_CTLR_BUFFER_SIZE/4];
static uint32_t mem_pos;
```

我们也可以将堆修改为异常很小的值，比如此处设置为 4000，看下实际输出：

```
[19:38:01.115] 收 ← LL Controller Version:bd5923c

[19:38:01.151] 收 ← BT controller memory pool used: 400 bytes, remain bytes: 3600, total:4000
BT controller memory pool used: 764 bytes, remain bytes: 3236, total:4000
BT controller memory pool used: 3784 bytes, remain bytes: 216, total:4000
BT controller allocating 1056 bytes failed
```

此时分配失败后会触发断言在初始化的地方卡住。

我们可以一开始设置比较大的堆值，然后再调整为合适的值即可。

2.2 App 以及 host 使用的堆（应用层堆全局使用 freertos 的堆） 为了方便资源管理，我们 app 和 host 全局使用 freertos 的堆，相应堆的分配函数 `pvPortMalloc`。

当前 SDK 哪些资源默认使用了 freertos 的堆呢?

- app 中显式使用 `pvPortMalloc` 的地方
- freertos task 的栈
- 创建 freertos 定时器时的栈（定时器也是 task)、
- freertos 创建信号量等

2.3 如何查看 freertos heap 的使用 我们可以通过打开 `app_config.h` 或者 `app_config_spark.h` 中的 `FreeRTOS Heap Usage Print` 选项（对应的宏 `CONFIG_FREERTOS_HEAP_PRINT`）显示的输出底层 controller 所需要的内存。

freertos heap 的宏是 `FreeRTOSConfig.h` 中的 `configTOTAL_HEAP_SIZE`，我们可以通过修改 `configTOTAL_HEAP_SIZE` 的值来改变全局堆的大小。

启动时会输出如下:

```
[19:47:57.875] 收 ← total allocated bytes:216,remain:5920
total allocated bytes:304,remain:5832
total allocated bytes:392,remain:5744
total allocated bytes:480,remain:5656
total allocated bytes:536,remain:5600
total allocated bytes:704,remain:5432
total allocated bytes:792,remain:5344
LL Controller Version:bd5923c

[19:47:57.918] 收 ← app started
total allocated bytes:848,remain:5288
total allocated bytes:2864,remain:3272
total allocated bytes:2960,remain:3176
total allocated bytes:3232,remain:2904
total allocated bytes:3328,remain:2808
total allocated bytes:4368,remain:1768
total allocated bytes:4464,remain:1672
total allocated bytes:4520,remain:1616
total allocated bytes:4752,remain:1384
total allocated bytes:4776,remain:1360
total allocated bytes:4808,remain:1328
```

我们故意将 `configTOTAL_HEAP_SIZE` 设置为一个很小的值，分配失败是会有如下显示:

```
[19:50:28.736] 收 ← total allocated bytes:216,remain:2848
total allocated bytes:304,remain:2760
```

(下页继续)

(续上页)

```
total allocated bytes:392,remain:2672
total allocated bytes:480,remain:2584
total allocated bytes:536,remain:2528
total allocated bytes:704,remain:2360
total allocated bytes:792,remain:2272
LL Controller Version:bd5923c

[19:50:28.778] 收 ← app started
total allocated bytes:848,remain:2216
total allocated bytes:2864,remain:200
total allocated bytes:2960,remain:104
pvPortMalloc failed
allocate 272 bytes failed,remain:104
```

另外我们也要注意一点，堆定义的空间可以适当多分配一点，有些堆的分配是在运行时才会去调用。

2.4.3 NDK 常见问题（FAQs）

TBD

2.5 其他文档

PAN1070 SoC 相关的其他文档请参考：

- PAN107x BQB Test Report
- PAN1070 功耗测试报告

2.6 NDK 更新日志

2.6.1 PAN1070 NDK v0.1.0

PAN1070 Nimble DK `v0.1.0` (2023-10-24) 已发布：

1. SDK

NDK 软件开发框架基于 Keil + FreeRTOS + NimBLE，其中：

- Keil 是 SDK 支持的软件开发环境
- FreeRTOS 是一个开源实时操作系统（RTOS），用于配合 NimBLE 实现蓝牙应用
- NimBLE 是一个开源低功耗蓝牙（BLE）5.1 协议栈，其实际上是 Apache Mynewt 项目的一部分

解决方案

- `price_tag`: ESL 价签方案演示例程，支持外部 SPI Flash 存储、EPD 墨水屏、低功耗模式、RF 通信等功能。

2. HDK

目前版本提供了如下硬件相关资料：

- PAN107B QFN40 测试板图纸、设计源文件、生产文件

3. MCU

目前版本提供了如下 MCU 裸机 Keil 例程及相关文档:

- ADC
- CLK
- CLKTRIM
- DebugProtect
- DMA
- EFUSE
- FMC
- GPIO
- I2C
- LP
- PRF_B250K_RX
- PRF_B250K_TX
- PWM
- SPI
- TIMER
- UART
- USB_HID
- WDT
- WWDT

4. DOC

目前版本提供了如下文档:

- NDK 快速入门指南
- NDK 开发环境介绍
- NDK 整体框架介绍
- Nimble 简介
- PAN107x 硬件参考设计指南
- Price Tag 价签方案例程说明
- MCU 底层外设驱动例程说明
- 低功耗开发指南
- NDK RAM 使用情况分析以及优化指南

5. TOOLS

目前版本提供了如下工具:

- 串口工具（PC 工具）
- Air Sync Debugger（手机测试软件安卓 APK）

- Google Home（手机测试软件安卓 APK）
- nRF Connect（手机测试软件安卓 APK）
- nRF Mesh（手机测试软件安卓 APK）
- Siliconlabs Bluetooth Mesh（手机测试软件安卓 APK）

6. 已知问题

- MCU USB_HID 例程暂未通过测试