E73-TBA/E73-TBB 使用手册

nRF52810/nRF52832 测试套件

目录

1. 开发板硬件介绍

1.1. 概述

随着智能穿戴、IOT 物联网的持续火热，越来越多的公司和个人始了穿戴式产品、物联网产品的研发。由于官方的开发资料都是英文版，中文资料特别缺乏，而且工程师在开发过程中会碰到一些问题由于语言沟通障碍也一时无法找到答案。所以为方便广大研发人员、创客、学生学习基于 nRF52832/ nRF52810 系列多协议 SOC 的开发，EBYTE 特别组织工程师开发了 nRF52832/ nRF52810 测试板。该开发主要有以下特点：

* 板载低功耗、高性能、支持多协议的 nRF52832/ nRF52810 芯片
* 支持 MESH 组网
* 支持蓝牙 5
* 集成高性能 USB 转串口芯片
* 二个用户按键
* 二个用户 LED
* 一个 NFC 天线接口
* 做工精美、尺寸 28*52.5MM、体积小巧、方便携带
* IPEX 接口，可外接天线
* 预留 SWD 接口，可外接 JLINK 调试器

1.2. 原理说明

E73-TBX 原理方框图

1.2.1. 芯片介绍

nRF52832 芯片为挪威 NORDIC 公司新推出的采用 32 位 CORTEX-M4 内核的支持蓝牙 BLE、ANT+ 、2.4G 私有协议的多功能 SOC 芯片。 该芯片跟上一代芯片 nRF51 相比，有着速度更快、功耗更低、功能更强等特点。

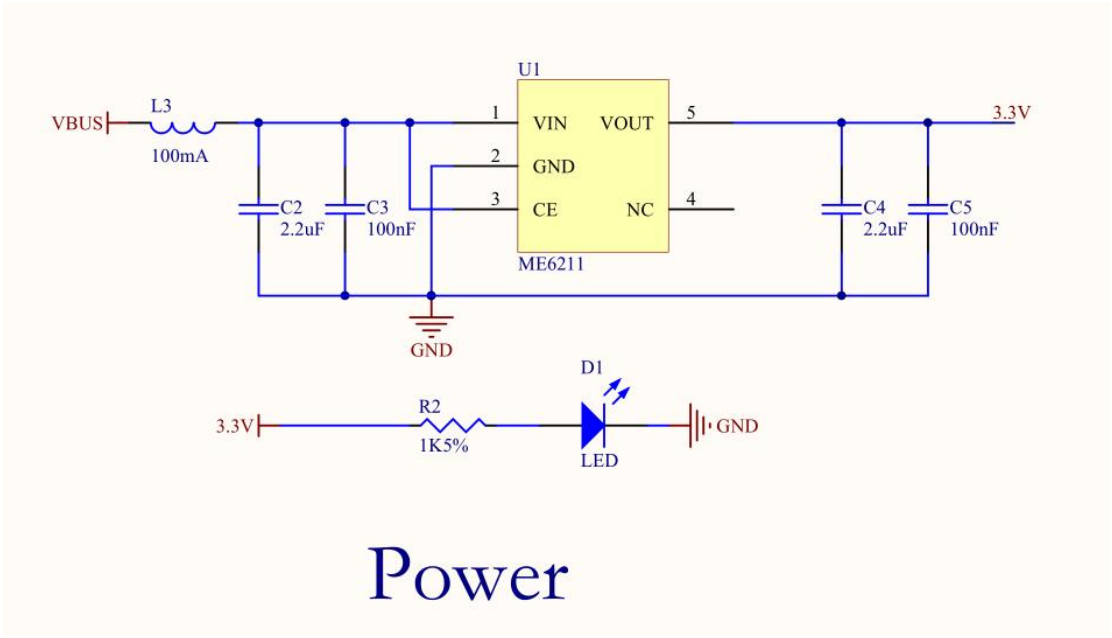
以下为 nRF52832 芯片的主要特征：

* 集成支持多协议的高性能 2.4G 射频收发器
* 32/64MHZ 外部时钟
* 片上集成 512KB FLASH,64 KB RAM
* 7.7mA TX at +4dBm
* 空中兼容 NRF24L 系列、NRF24AP 系列、NRF51 系列 SOC
* 20 个 PPI 通道
* 可设定发射功率：-20dBm-4dBm
* 超低待机电流：400nA
* 接收灵敏度 -96dBm (BLE)
* 集成 NFC，支持近场检测与唤醒，支持 OOB 配对
* 集成 BALUN
* 集成 PDM 、IIS 接口
* 3X4 通道 PWM 输出
* 12BIT 200KSPS ADC

原厂考虑到有些客户对产品的成本比较敏感，特推出了一颗 COST DOWN 的版本--- nRF52810。这颗 IC 的 RAM 和 FLASH 是 192K/24K,相比 nRF52832 是小了一些。但是应付一般的应用是没有问题的。

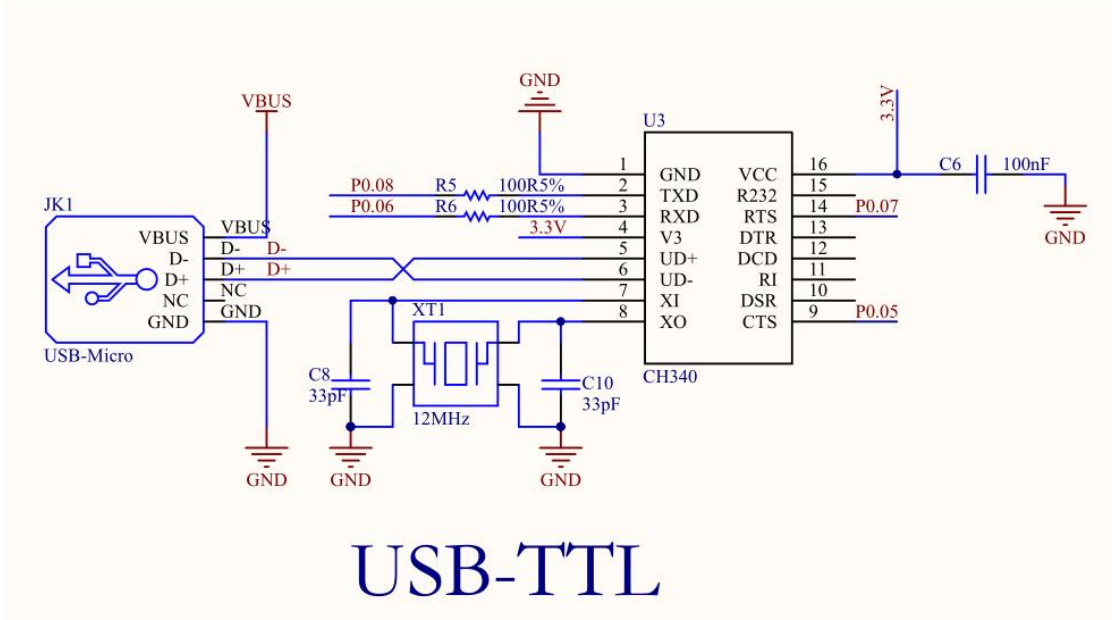
1.3. 开发板电路图介绍

1.3.1. 电源电路

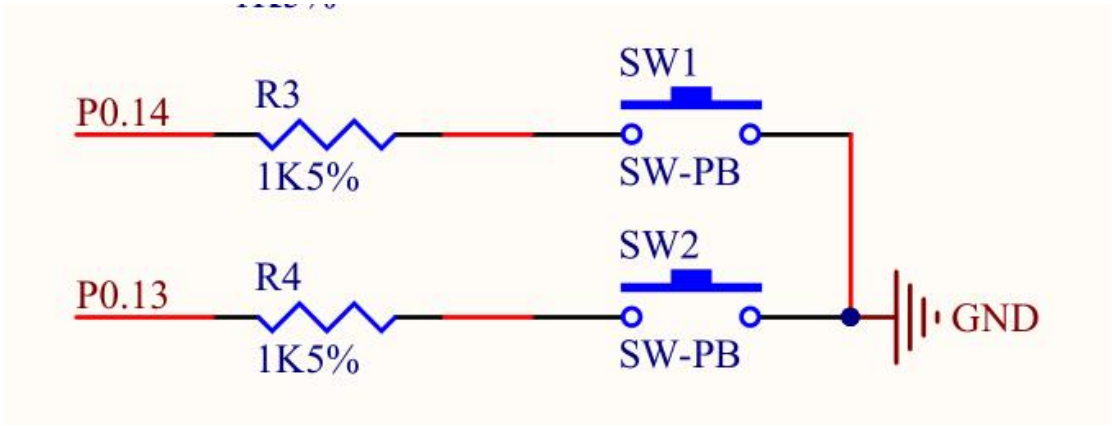


5V 电源从 USB 口进来以后，通过自恢复保险丝保护，再经过 ME6211 输出稳定的 3.3V 电压。

1.3.2. USB 转串口电路

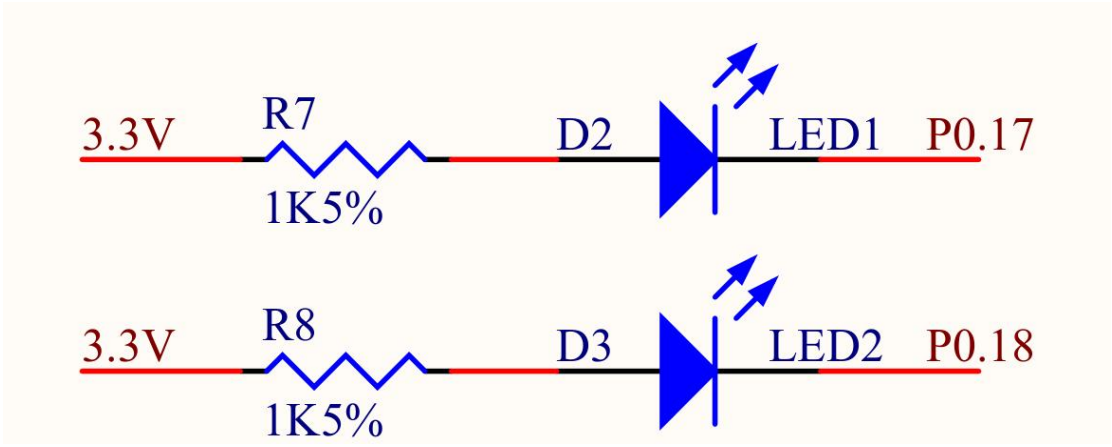


1.3.3. 用户按键



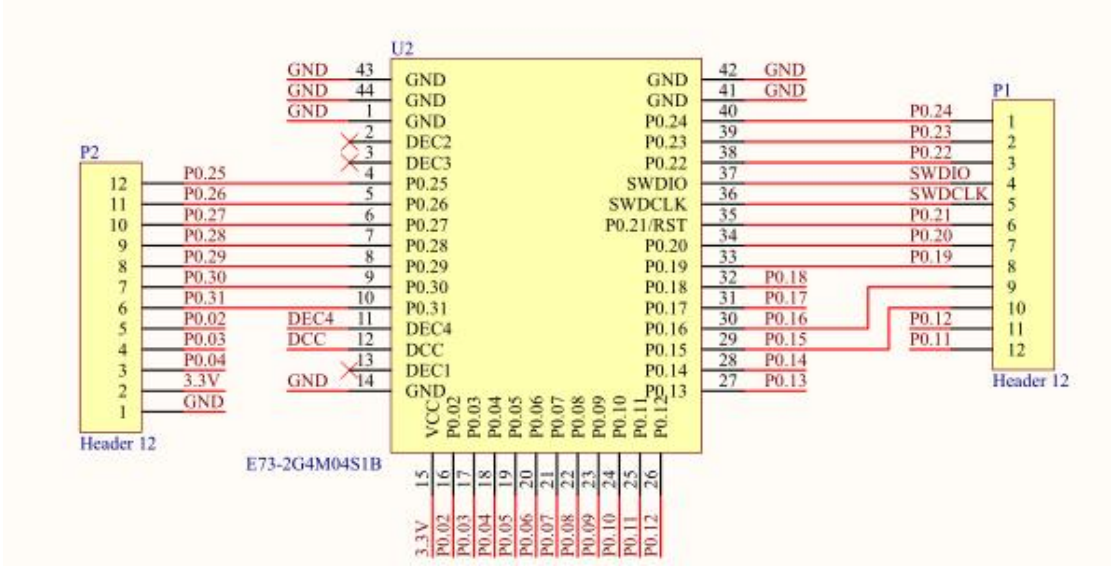
预留 2 个用户按键。

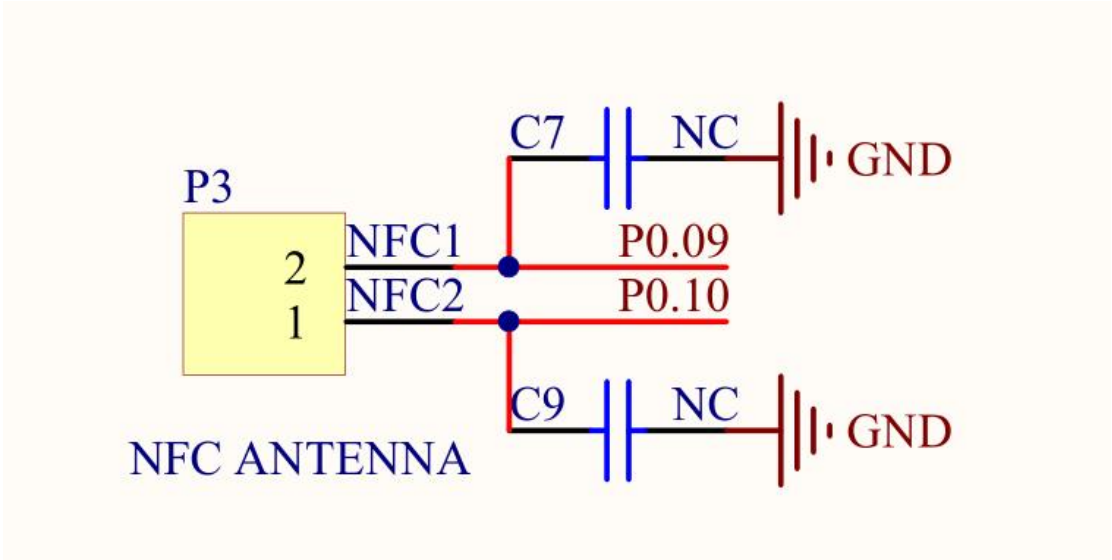
1.3.4. 用户 LED



预留 2 个用户 LED。

1.3.5. 扩展排针



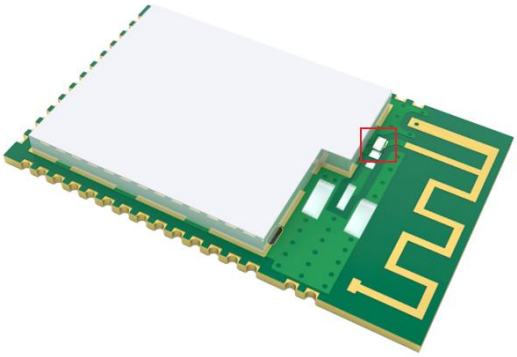


剩余的 GPIO 和 NFC 接口都引出来了。

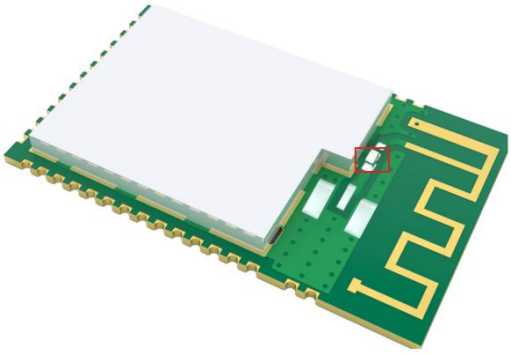
1.4. 开发板接口说明

1.4.1. 外接天线的接法

E73-TBX 上预留了 IPEX 天线座焊盘接口。假如用户需要外接天线的话，请把模块最后一颗匹配电容焊在下图圈内位置，然后再焊接一个 IPEX 天线座子，详细可见对应模块手册中天线选择部分。



选择为：板载 PCB 天线 （默认）



选择为：IPEX 接口

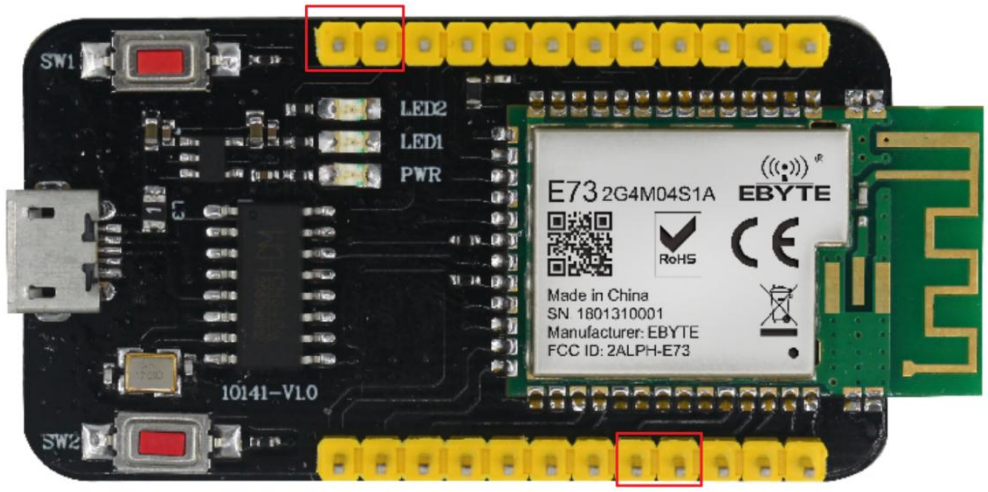
1.4.2. 供电和串口

开发板上的 USB 口为开发板提供 5V 电源，同时它也是板载 USB 转串口芯片 CH340G 跟 PC 的通讯口。开发板支持 USB 口和 JLINK 调试器供电。但是请不要二者同时供电。

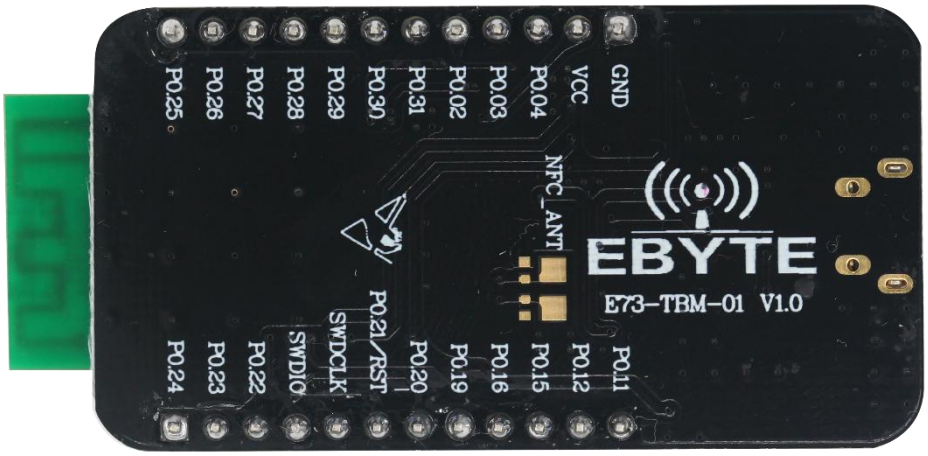
1.4.3. 程序下载接口

开发板预留了 4 线 SWD 接口，支持 JLINK V8,JLINK V9 等常见的调试工具.不建议用户使用 JLINK OB 调试，OB 对 NRF52 支持不好，有点不太稳定。

调试口位置如下：测试板背面有引脚定义，请查看，注意 VCC=3.3V



1.4.4. 模块引脚说明



1.5. 开发板开箱测试

1.5.1. 出货清单

E73-TBX 测试板 X 1（E73-TBA/E73-TBB 统称为 E73-TBX。）

MICRO USB 线 X 1

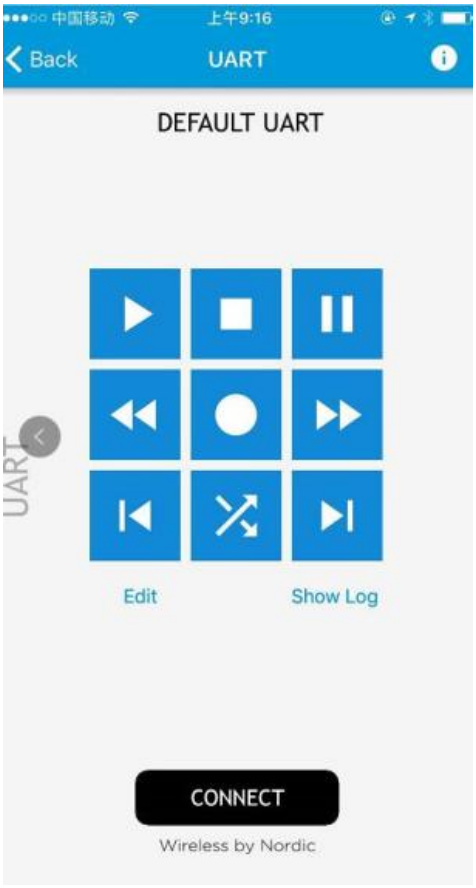
1.5.2. 开箱测试

当收到测试板的时候请先确认下货物是否完整。如果不完整的话请尽快联系淘宝客服处理。出货的时候，本开发板烧录的是蓝牙透传程序。本测试方法通过测试 PC 串口跟手机 APP 之间的数据传输功能来确定开发板是否完好。开箱测试方法如下：

1) 先在电脑上安装串口助手软件，手机端安装 nRF Toolbox 这个 APP。在手机上运行 nRF Toolbox 这个 APP。找到“UART”这个图标。

见下图。

2) 点“UART”进入后，会发现有 9 个按键。这 9 个按键的图标我们可以自行设定。而且，我们可以设定当点击这些图标的时候 APP 发送的数据内容。点“EDIT”，进入按键图标的编辑和发送内容的编辑。



3) 我们把第一个按键图标设置为“1”；当改按键被按下发送的内容设置为“12345678901234567890” 。

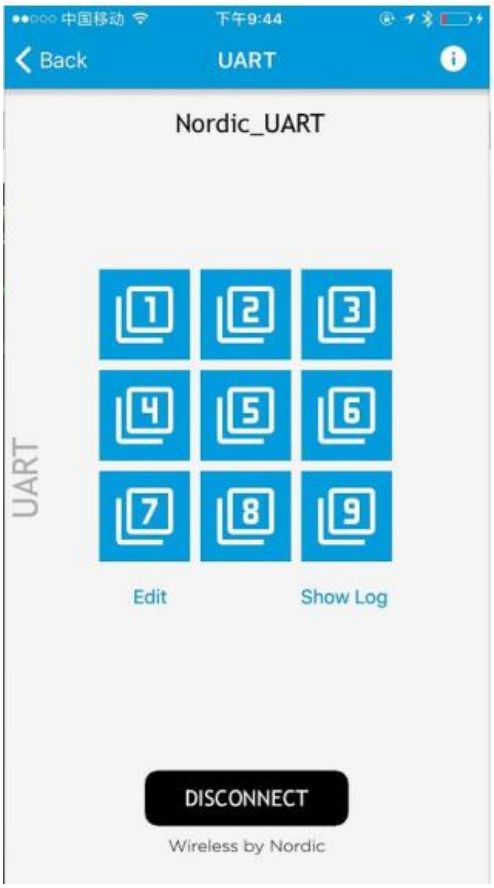
4) 我们把第二个按键图标设置为“2”；当该按键被按下的时候发送的内容设置为“09876543210987654321”

5) 我们再把第三个按键图标设置为“3”，然后把发送内容设置为“inforlink_nRF52832 E73TBM01”。

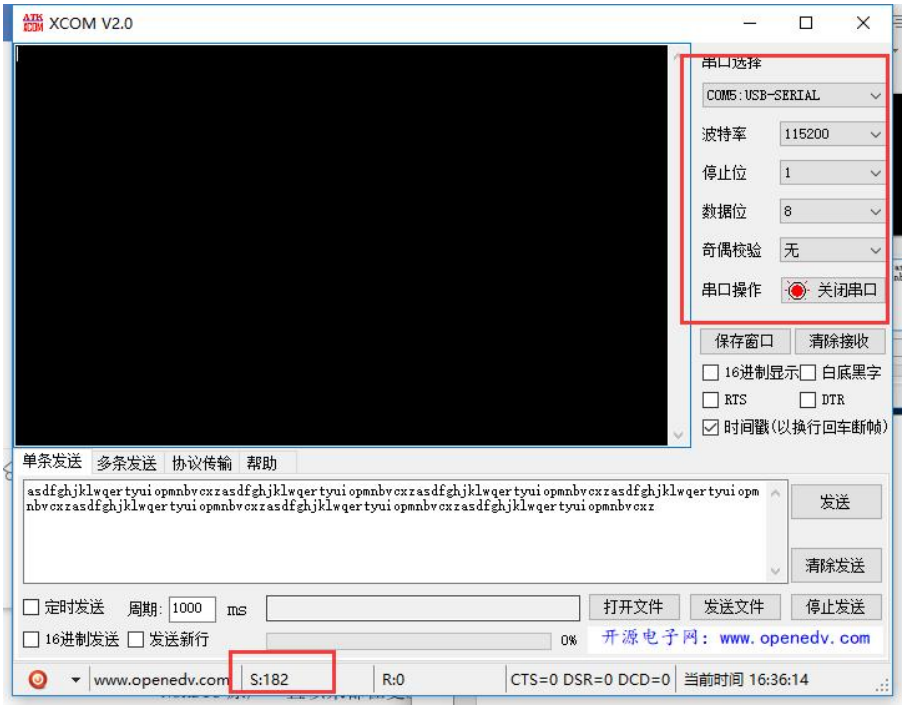
6) 编辑完以后，点“DONE”退出设置界面。

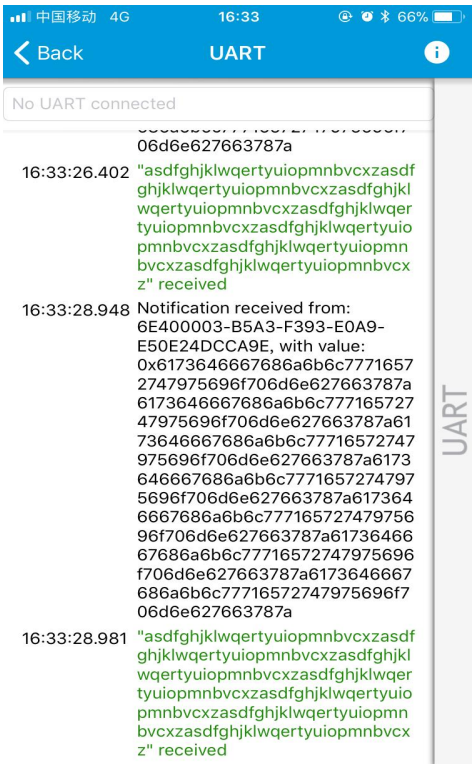
7) 用赠送的 USB 线把开发板和电脑连接上。此时红色的 LED 在闪烁，表示正在发送蓝牙广播。

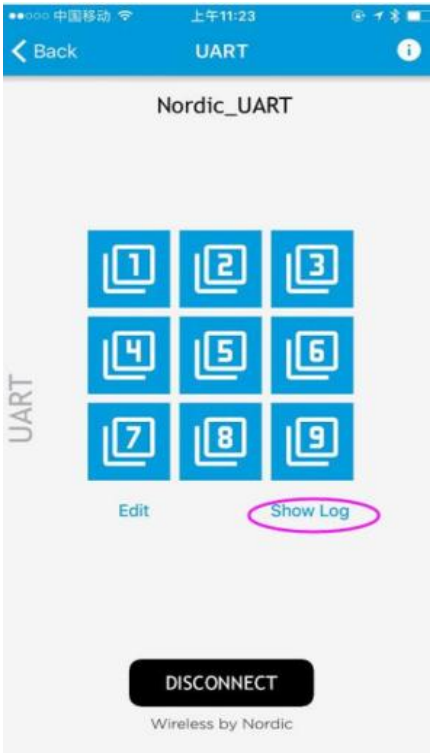
8) 点击 “CONNECT”连接，会搜到名为“Nordic UART”的蓝牙设备。点击后建立连接。



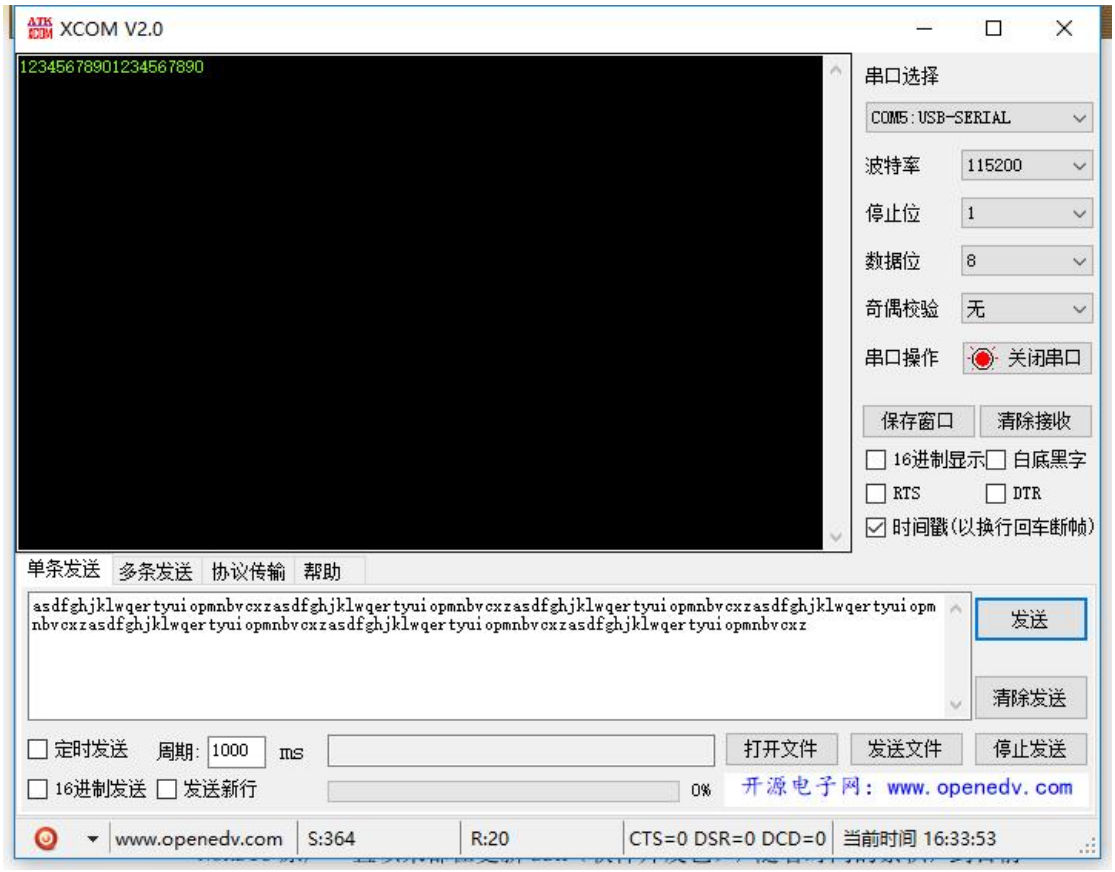
9) 运行 PC 端串口助手软件，按下图设置好各参数。在发送数据窗口输入我们要发送的数据“1234567890”，点击发送，我们可以在 APP 的 LOG 区看到 PC 端发过来的数据。

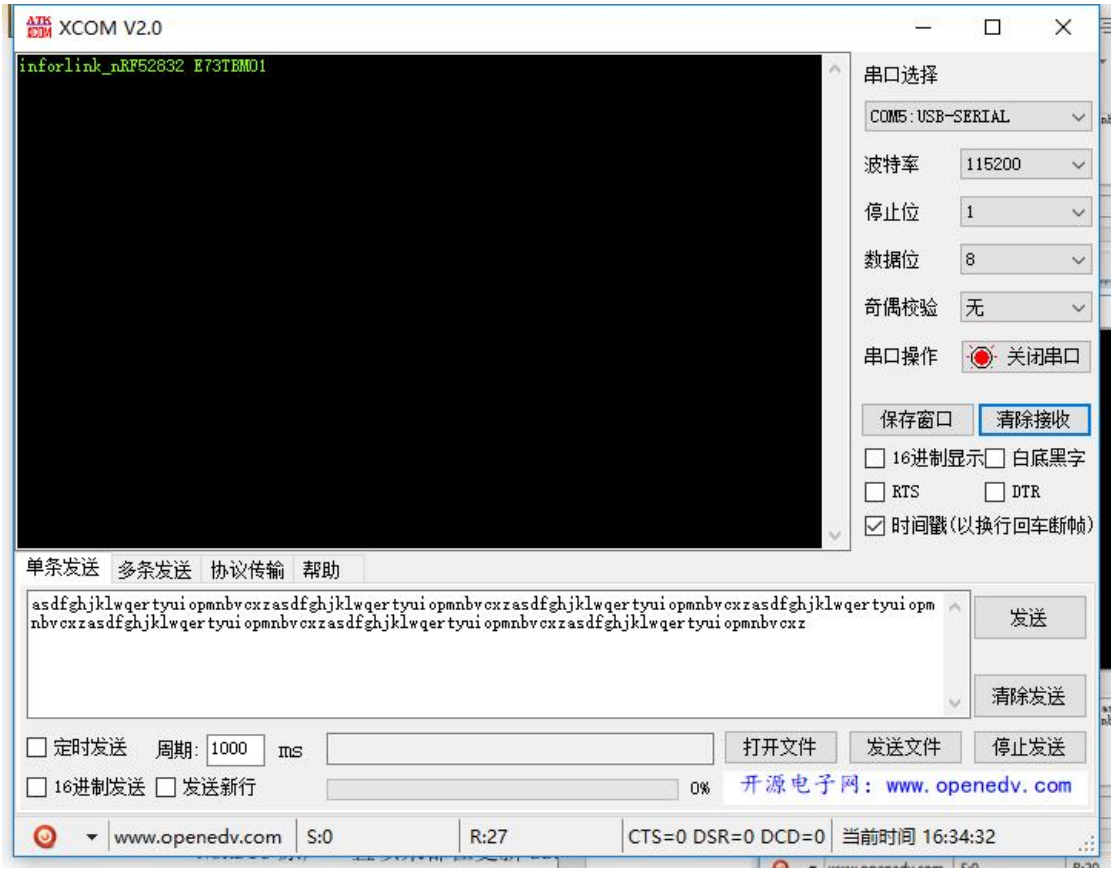


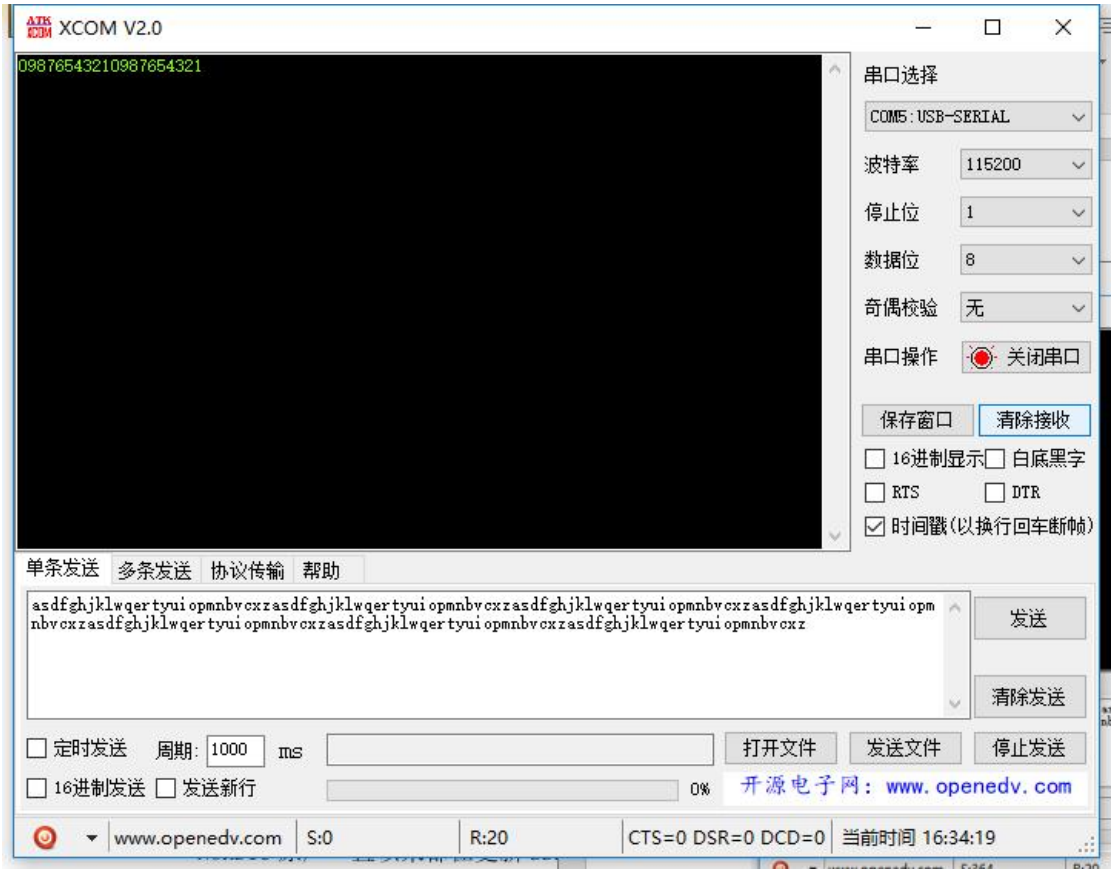




10） 我们在手机 APP 端点“1”键，“2”键，“3”键，PC 串口端会收到相应的数据。







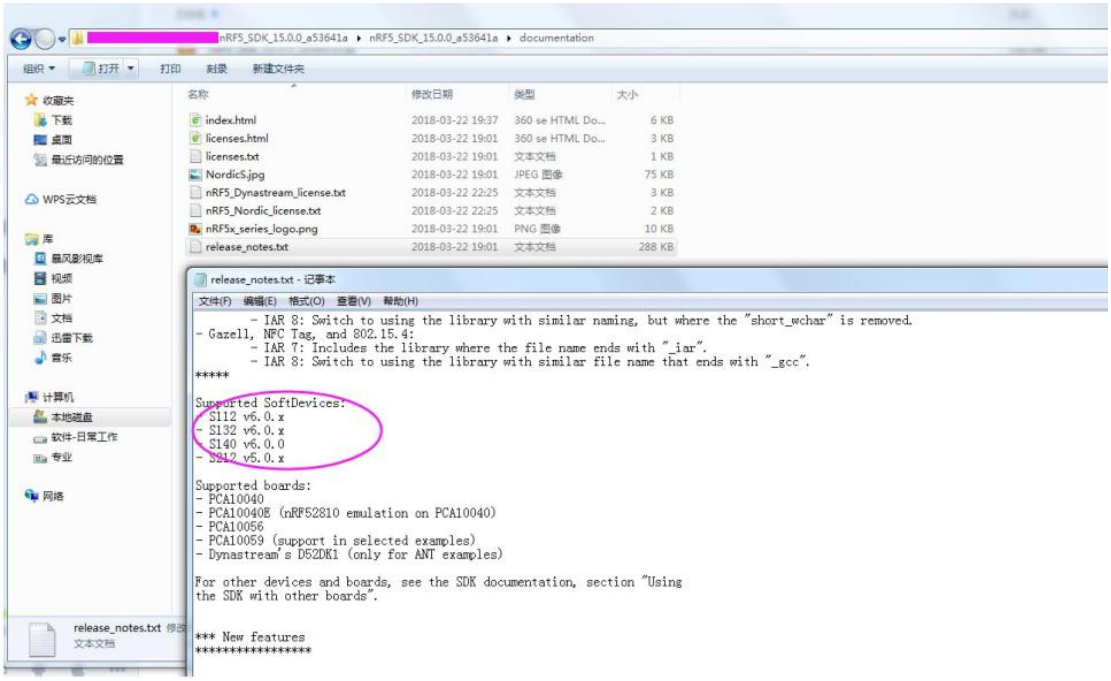
2. 开发工具介绍

2.1. 程序烧录的说明

NORDIC 原厂一直以来都在更新 SDK（软件开发包），随着时间的累积，到目前 SDK 版本有 15 个，相应的协议栈版本有几十个之多。而不同的 SDK 开发不同的功能需要唯一的协议栈版本与之匹配。这就给很多初学者带来很多困惑。为帮助初学者少走弯路，这里特别指出几点：

1) 不同的 SDK 版本只对应指定的协议栈版本

不同的 SDK 版本，对应的协议栈版本是不一样的，不能随便烧录。否则烧录会报错，或者烧进去板子根本跑不起来。SDK 跟协议栈之间的对应关系请查 SDK 附带的 RELEASE NOTE。见下图：

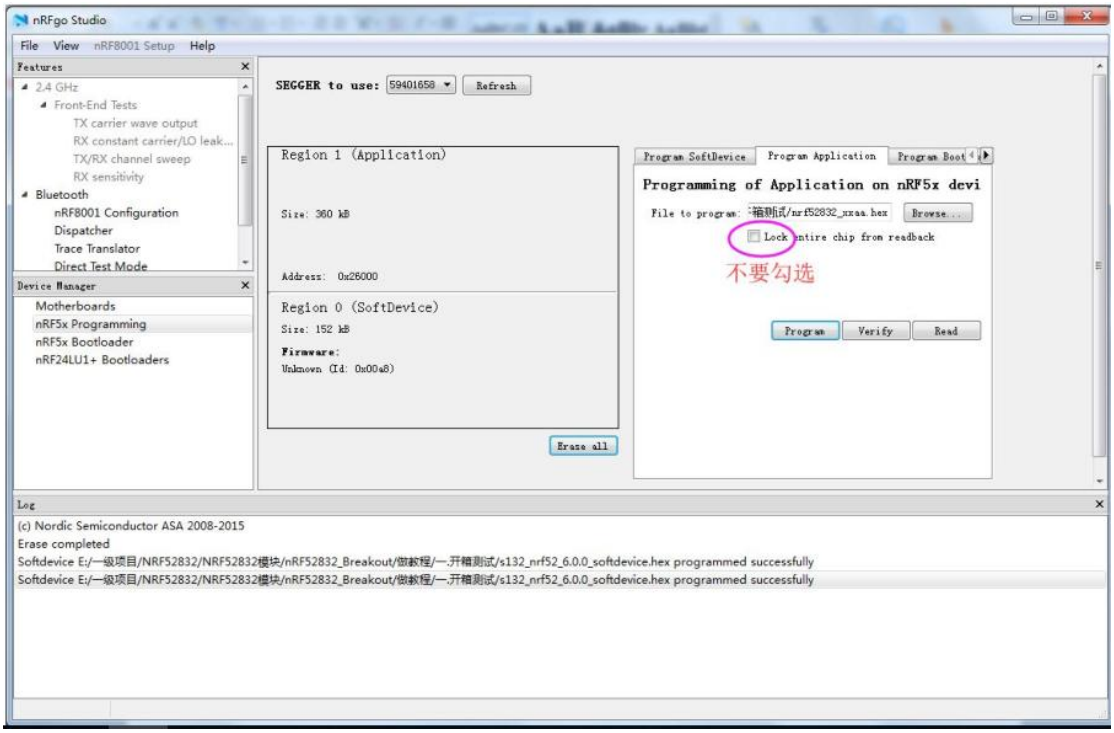


2) 什么情况下需要烧协议栈，什么情况下不需要

NORDIC 的 NRF5X 芯片有个特点，芯片可以跑蓝牙，作为一个蓝牙 SOC 使用；也可以不跑蓝牙，作为一个普通的 MCU 使用。可以简单地这样理解：比如我们写一个简单的跑马灯，无蓝牙功能，就不用烧协议栈进去。而我们开发蓝牙透传模块，就需要跑蓝牙了，那么就要烧协议栈进去。二者不能混用，否则烧录的时候会提示错误。

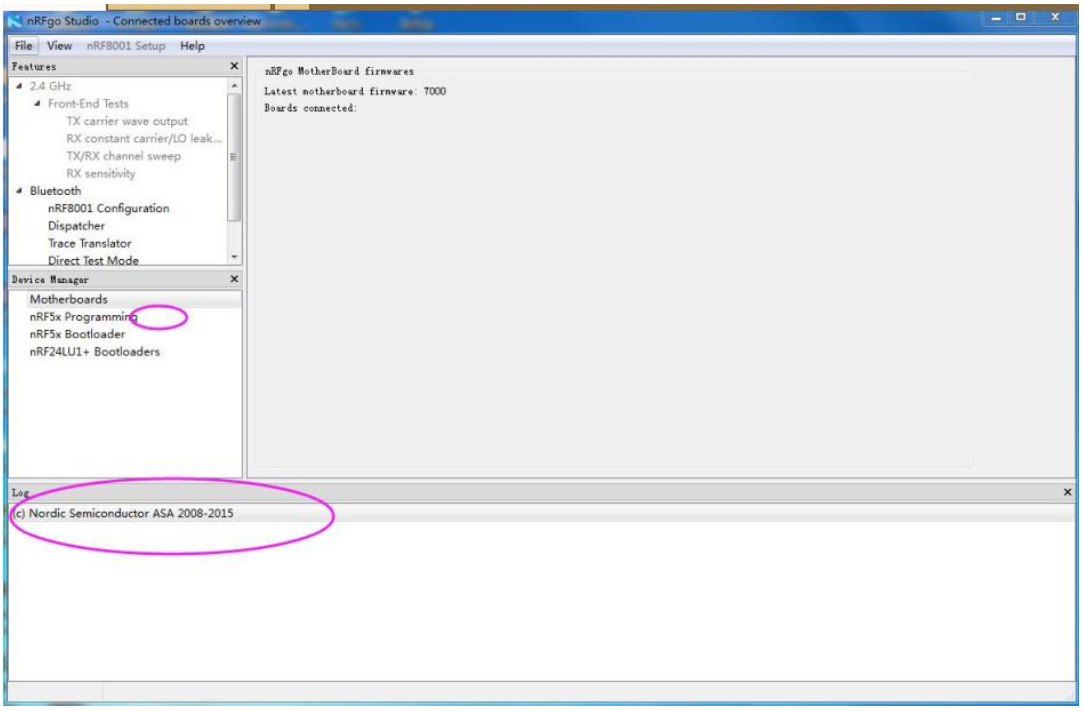
3) 开发阶段烧录程序的时候请不要勾选回读保护

用 nRFgo Studio 烧程序，不要勾选回读保护。否则，当你在 MDK 里面烧程序的时候会提示烧录失败。



2.2. NRFGO STUDIO 的使用说明

把 JLINK V9 一端插电脑 USB 口上，另外一端接在 测试板 的 SWD 烧录接口上。在开始菜单或者桌面双击运行 nRFgo Studio 图标以后，会出现以下界面：



注意看上图中的 2 个圈。第一个小圈位置如果显示黑色，说明 JLINK V9 烧录器已经被电脑找到，驱动已经安装好。如果显示灰色，说明 JLINK 没有被电脑找到。需要检查以下：

4) V9 是否完好

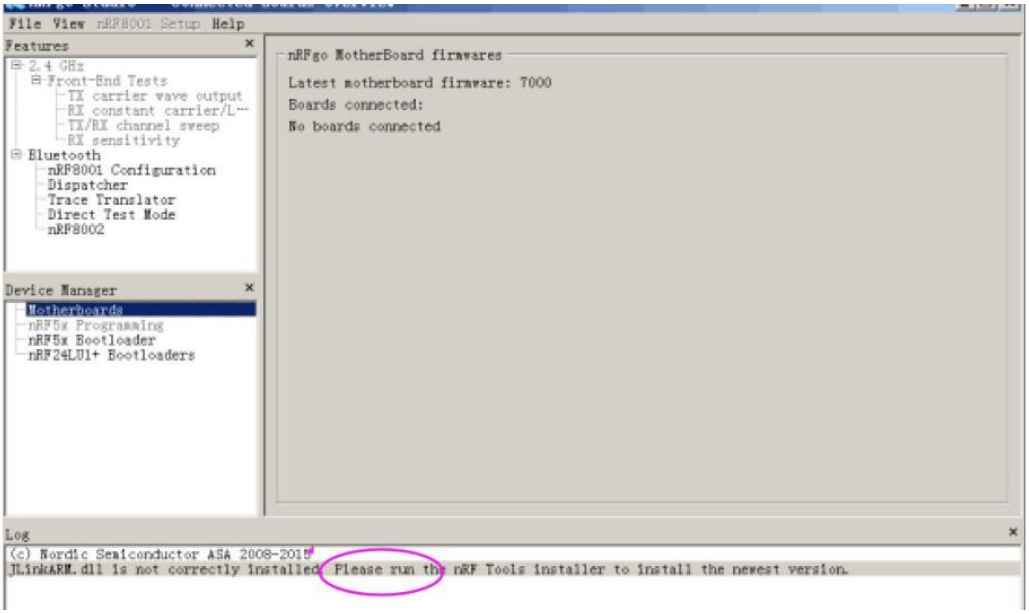
5) USB 线是否 OK

6) JLINK V9 到目标板的 VCC 跟 GND 是否接反，导致电流过大

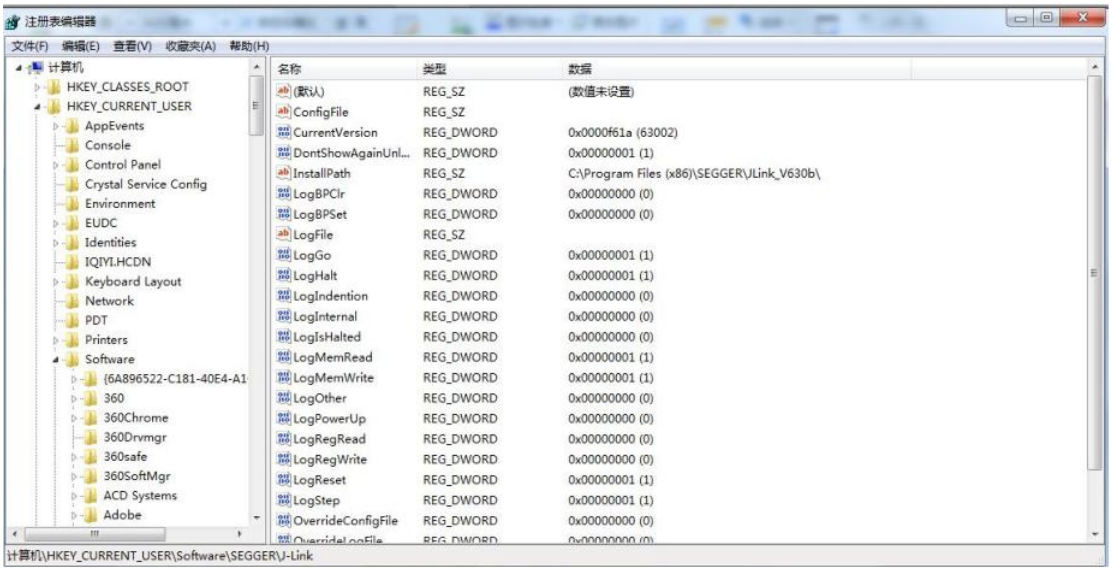
7) PC 的 USB 口是否松动

8) 第二个大圈位置是否有提示异常信息

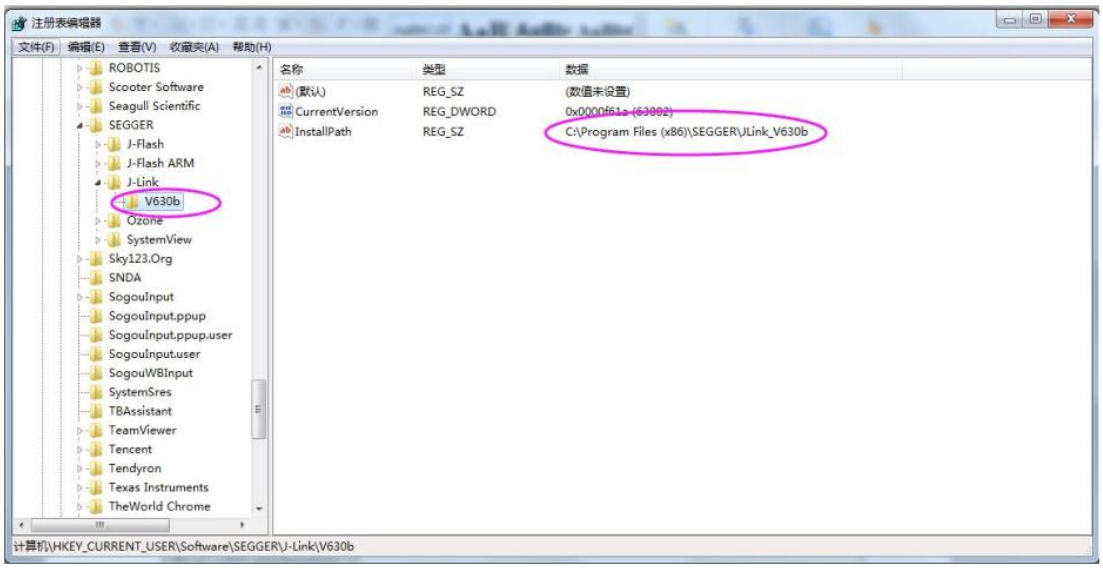
提示的异常信息主要是以下：



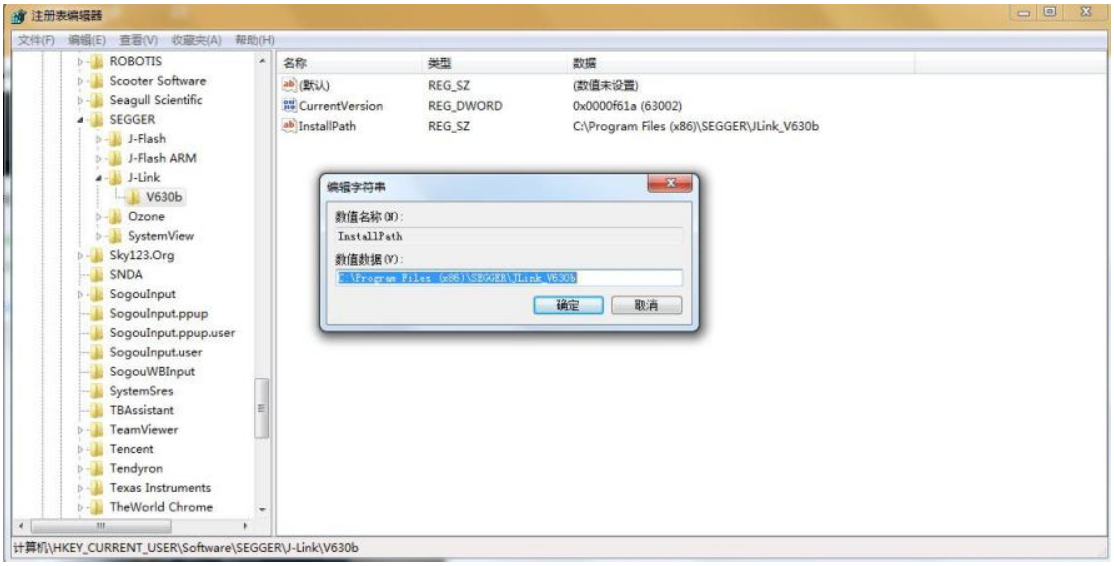
假如我们的电脑已经安装过 JLINK 驱动，而我们重新安装了另外一个版本的 JLINK 以上问题就容易出现。知道了原因，那解决办法就比较容易了。我们只要手动更新一下注册表，问题就可以解决。



按 WIN+R 键，在弹出的对话框输入 regedit,回车。会出现以下界面：

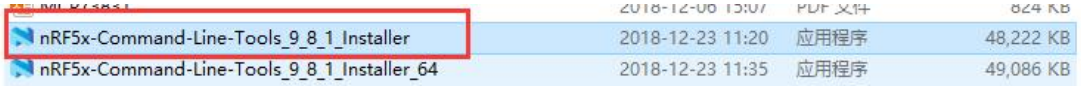


我们找到 HKEY_CURRENT_USER/Software/J-Link/V630b,然后修改 InstallPath 就可以了。



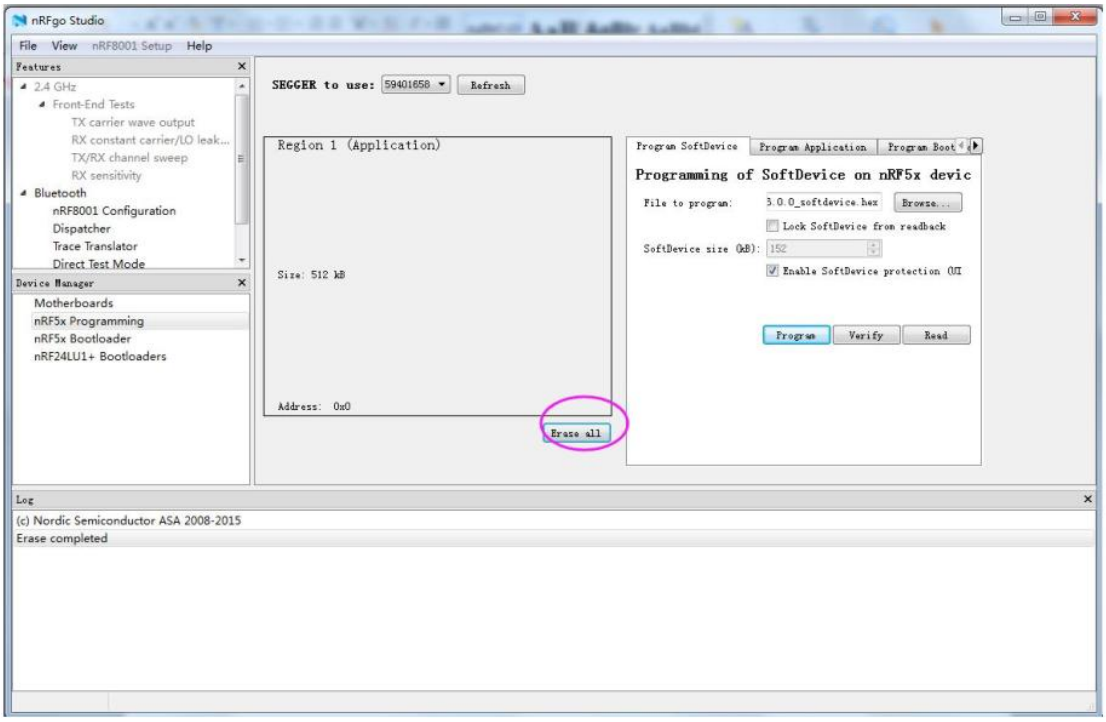
如果 64 位系统不能识别 NRF52810 芯片，直接安装即可解决

以上问题解决以后，我们来熟悉一下如何用 nRFgo Studio 来烧录程序。有以下步骤：



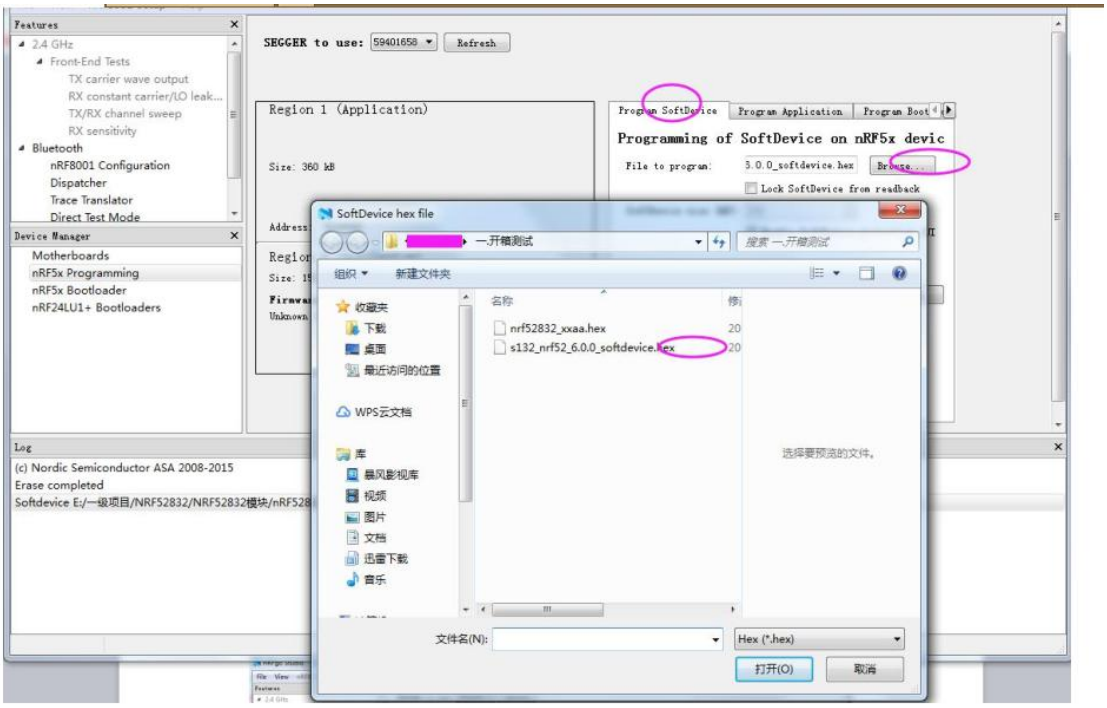
1) ERASE ALL

在烧录芯片之前，先擦除芯片。擦除成功后，最下方 Logo 提示框会提示 Erase complete。



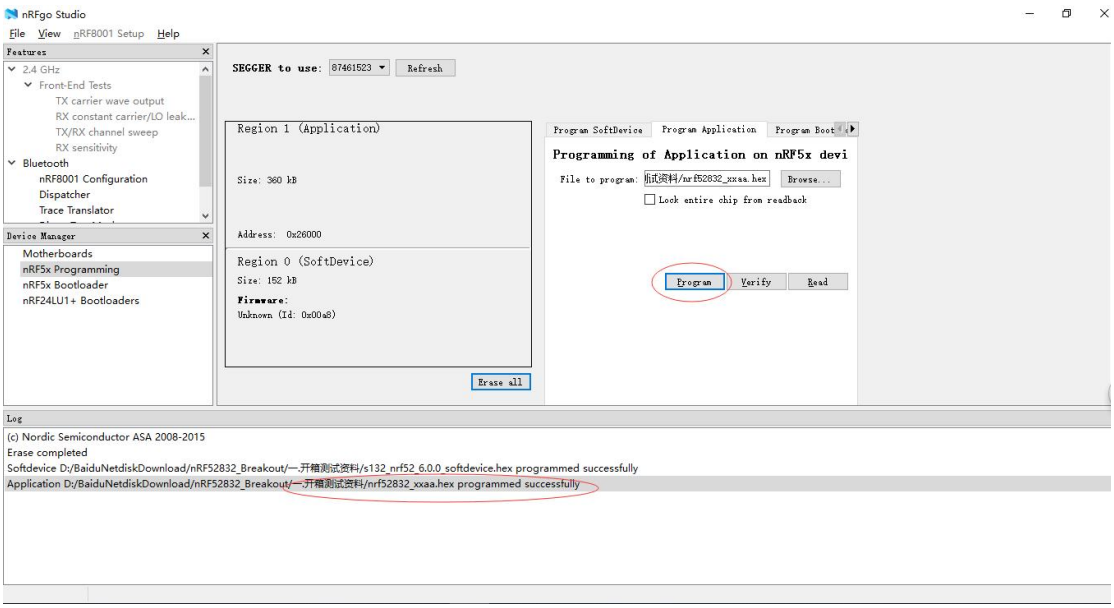
2) Program SoftDevice（烧录协议栈）

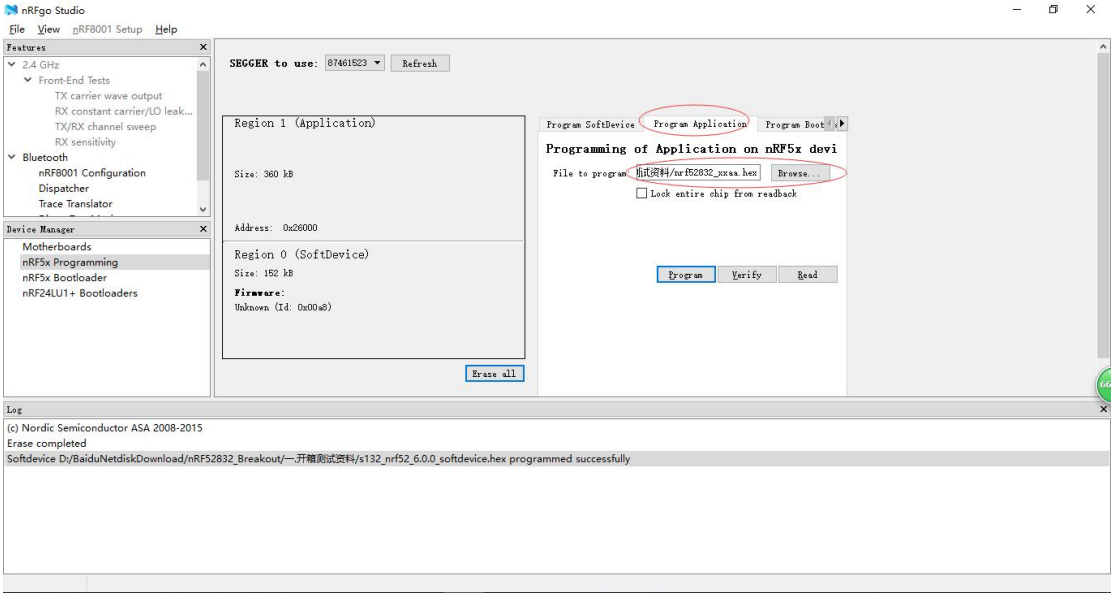
点 Browser，找到要烧录的 HEX 文件的路径，双击选中，然后点击 Program。



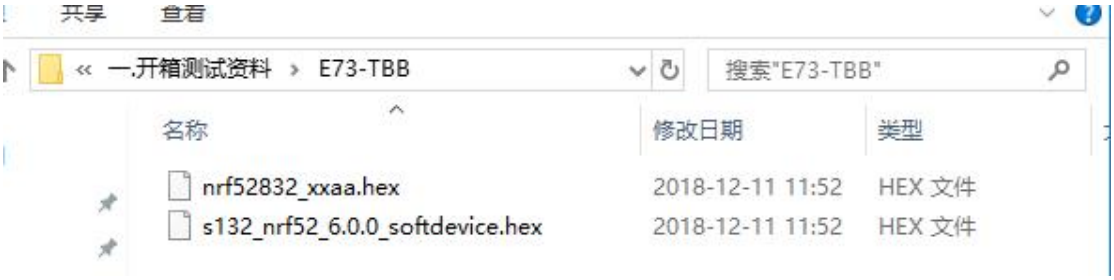
3) Program Application(烧录应用)

跟烧录协议栈的方法是一样的。



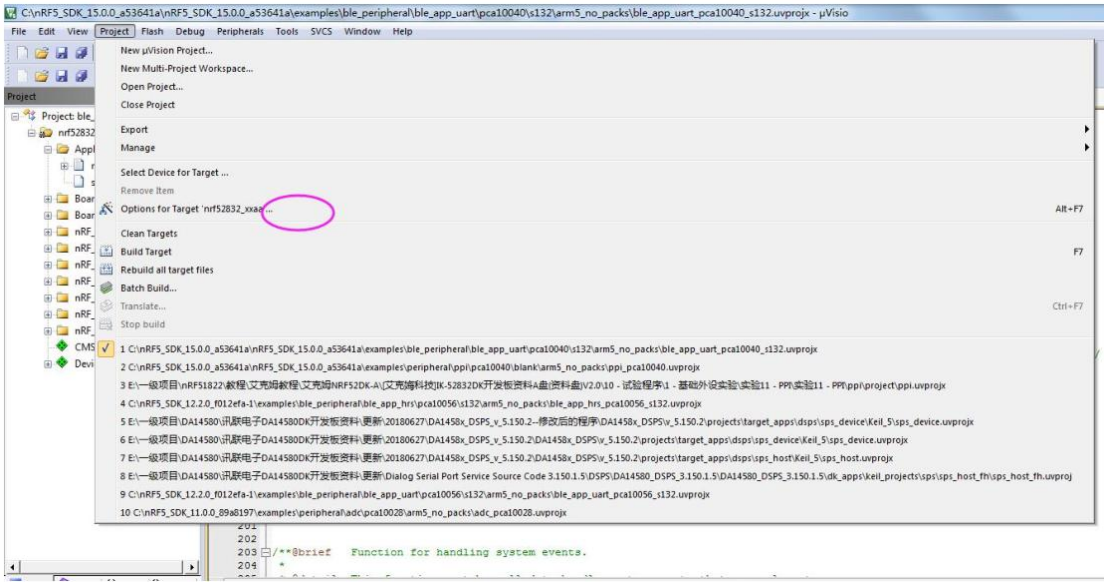


到此，如何用 NRFgo Studio 烧录协议栈和应用的方法介绍完了。上一次提到的关于回读保护的勾选，大家还是要留意一下，以免产生不必要的麻烦。另外要注意一下，协议栈和应用的烧录是有先后顺序的。大家可以把开箱测试这个文件夹的 HEX 烧到开发板上试下，熟悉下烧录过程。

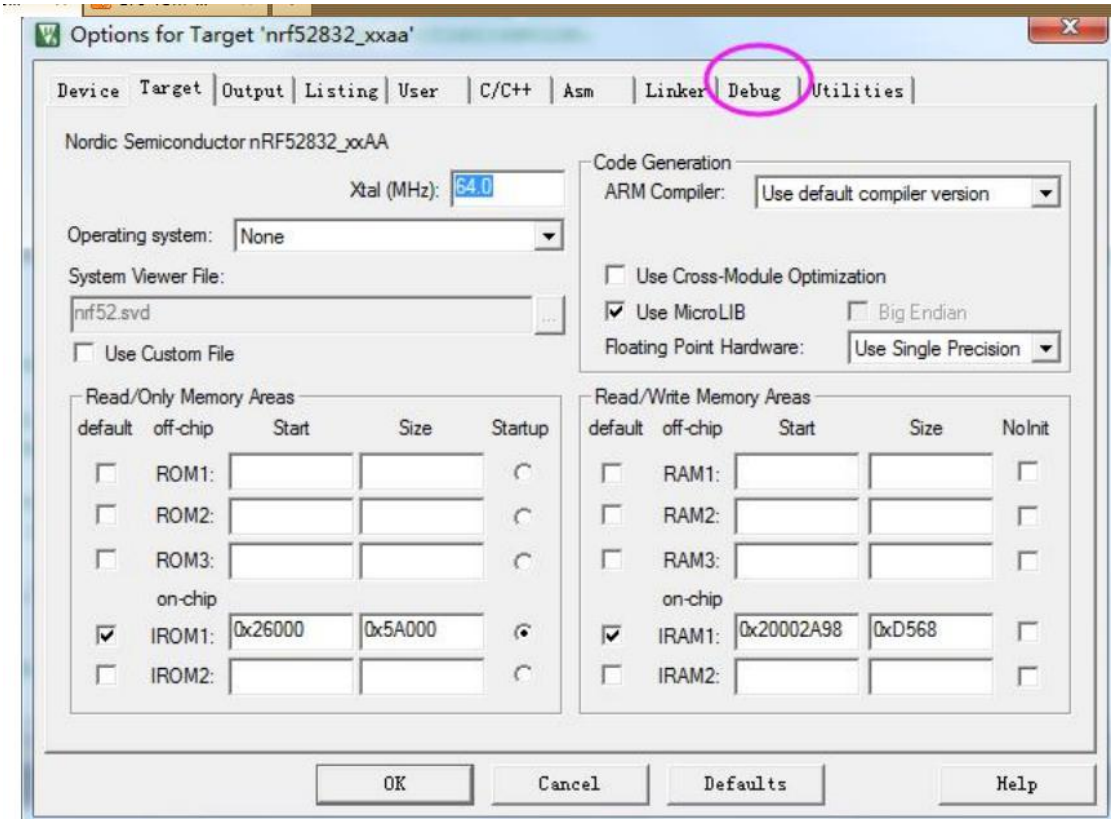


2.3. 调试仿真烧录工具的介绍

我们建议当用户调试 E73-TBX 测试板的时候选择 JLINK V9。跟 V8 相比，V9 的速度稳定性提高了不少，而且特别不容易丢固件，省去了调试器经常变砖的烦恼。我们用 JLINK 调试 E73-TBX 的时候需要进行相关的设置，请参考下图：



在 Project 下选 Option for Target xxxx。



注意选 SWD 接口。

3. 蓝牙 4.X BLE 的基础知识

3.1. 什么叫 BLE

BLE 是 Bluetooth Low Energy 的缩写。BLE 又常被称为 Bluetooth Smart ,是一种低功耗、低速率的短距离传输协议。由于它有着低功耗、方便易用、连接方便等特点，这几年在智能穿戴、智能家居、无线传感器网络等领域得到了越来越广泛的应用。

3.2. 传统蓝牙跟低功耗蓝牙的区别

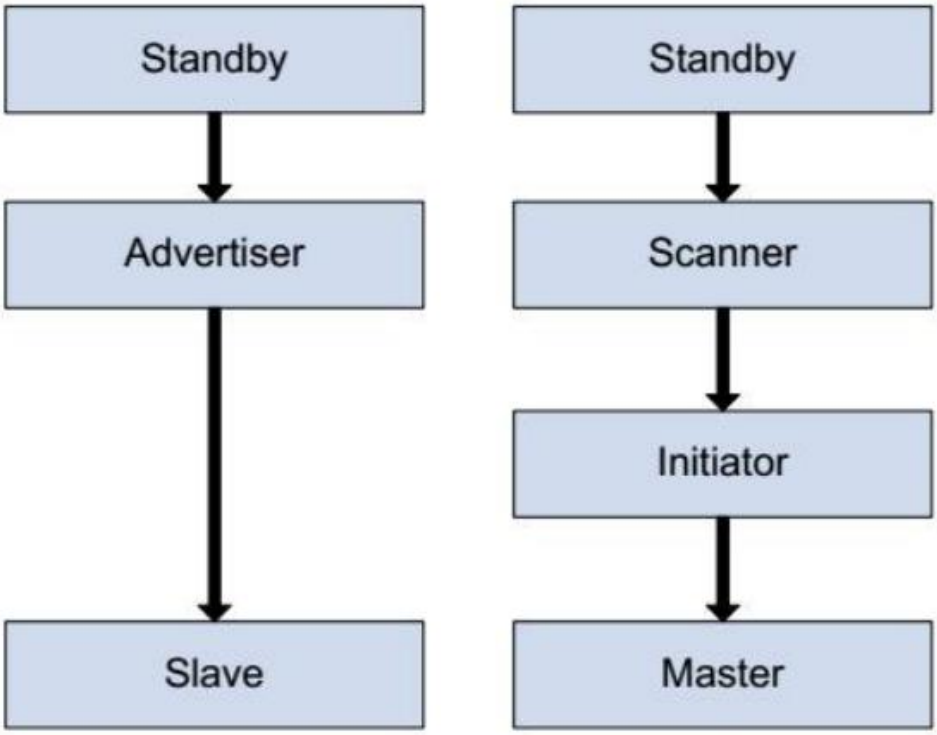
我们常把 Bluetooth Classic 称为传统蓝牙。传统蓝牙跟低功耗蓝牙的区别主要体现在下几个方面：

1) 发射功率
2) 通讯距离
3) 数据传输速率

传统蓝牙有 CLASS1、CLASS2、CLASS3 之分，他们的发射功率一般比较高，通讯距离也比较远，某些版本可以达到 100 米的距离。传输速率方面，传统蓝牙的传输速率也比较高，适合用于音频传输等场合。而低功耗蓝牙的特点确与之相反。它的发射功率比较小，一般为-30~+4dBm,适合短距离传输小数据。

3.3. 设备的工作状态和蓝牙设备的种类

BLE 设备有以下几个工作状态：



当 BLE 设备没有跟任何设备建立连接且没有广播的时候我们称之为就绪态。假如设备从就绪态通过发送蓝牙广播跟跟主机相连，我们称之为从机；假如设备从就绪态通过扫描蓝牙外设的广播然后跟从机相连，我们称之位主机。

3.4. 蓝牙广播的解析

3.4.1. 前言

报文由数据字节组成同时是按比特传输的，这就免不了牵涉到字节序的问题。对于各个字节的传输，总是从最低位开始传输。如 0x80 是按 00000001 发送的，0x01 是按 10000000 发送的。同时大多数字节域又是从低字节开始发送的。如 0x010203 发送序列为 110000000-100000010000000。之所以说大多数，是因为并不是所有的数据都会从低字节发送从后面的抓取的广播报文中也能看不来。另外由于抓包软件可能并不一定能完全知道哪些数据时从低字节开始发的，抓取的广播数据可能有一些需要按字节倒着读。

3.4.2. BLE 广播数据结构

在链路层，BLE 的广播报文分为如下几个部分：

| 前导| 存取地址 | 报头| 长度| 广播数据| 校验|

一般抓包工具抓取到广播数据后显示出来的都会分段显示，所以你很容易看出各个段的数据，本文重要是解析广播数据段中的内容。

前导（1 字节）：

不知道的可以理解为“同步头”，主要是用来配置接收机的自动增益控制。

存取 地址(4 字节)：

对于广播报文来说是固定的 0x8e89bed6（同时存取地址也决定前导的序列，在这里并是重点，不做过多介绍）

报头（1 字节）：

依次为广播报文类型(4bit),保留位(2bit),发送数据地址类(1bit),接收 地址类型(1bit)

长度（1 字节）：

指示广播数据的长度(广播地址 AdvA+数据 AdvData)

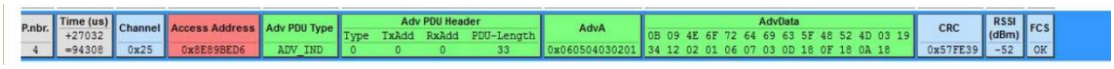
广播数据:

我们需要解析的数据段，后面会详细说明。

校验(3 字节:):

CRC 校验

3.4.3. 抓包分析



从中可以明显的看出各个段的内容:

红色字段存取地址就是广播的固定接入地址 0x8e89bed6。(抓包软件未转换字节序)、

报头字段中：

广播类型是通用可连接广播（TYPE 为 0）

地址类型为公共地址，TxAdd 和 RxAdd 都为 0

长度字段指示 adv+AdvData 长度和广播设备地址都是我自己设定的

ble_gap_addr_t add={.addr_type=BLE_GAP_ADDR_TYPE_PUBLIC,

.addr={0x01,0x02,0x03,0x04,0x05,0x06}}

因为地址是 48-bit address, LSB format.所以真实地址为 0x060504030201.AdvDara 中一大堆数据就是我们需要解析的。下面是详细信息，其中有抓包软件加的帧头数据。

```
+-------------------------------------------------+----------------- - - -
|     Packet sniffer frame header                 |
+----+-------------+-------------------------+-------+
|info| Packet nbr. | Time stamp              | Length|  Packet data
+----+-------------+-------------------------+-------+----------------- - - -
| 01 | 04 00 00 00 | EB 01 B9 02 00 00 00 00 | 2D 00 | 2C D6 BE 89 8E 00 21 01 02 03 04
+----+-------------+-------------------------+-------+----------------- - - -
```

从 Packet data 开始

2C 为 packet data 总字节数

D6 BE 89 8E 为接入地址(字节序问题)

00 为报头字段(通用广播类型，public 地址)

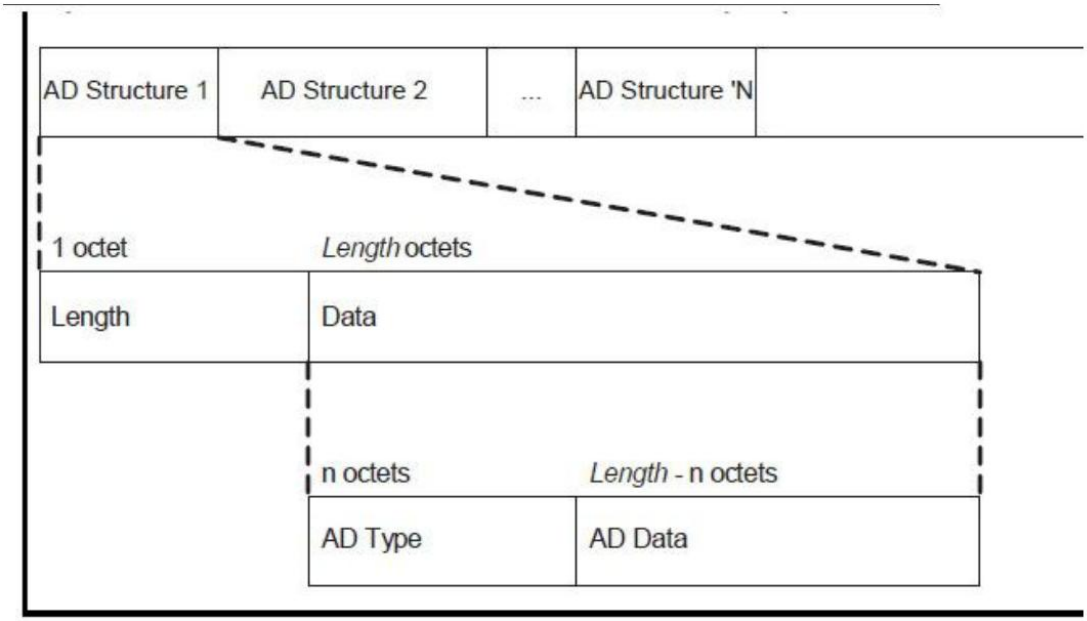
21 为长度字段的值，指示 adv+AdvData 的总字节数

01 02 03 04 05 06 为广播设备地址

之后变为重点需要解析的 AdvData 部分。首先还是需要知道这一部分的数据格式。

蓝牙说明书 4.1 中说明 数据格式为

|length|AD type|AD data|



即 Advdata 都是由这种格式的数据段组成。

Length 即为一小段数据的长度

AD type 指示 AD Data 数据的含义

AD type 的定义可以为下图中的一种

Macros

```
#define BLE_GAP_AD_TYPE_FLAGS 0x01
#define BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_MORE_AVAILABLE 0x02
#define BLE_GAP_AD_TYPE_16BIT_SERVICE_UUID_COMPLETE 0x03
#define BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_MORE_AVAILABLE 0x04
#define BLE_GAP_AD_TYPE_32BIT_SERVICE_UUID_COMPLETE 0x05
#define BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_MORE_AVAILABLE 0x06
#define BLE_GAP_AD_TYPE_128BIT_SERVICE_UUID_COMPLETE 0x07
#define BLE_GAP_AD_TYPE_SHORT_LOCAL_NAME 0x08
#define BLE_GAP_AD_TYPE_COMPLETE_LOCAL_NAME 0x09
#define BLE_GAP_AD_TYPE_TX_POWER_LEVEL 0x0A
#define BLE_GAP_AD_TYPE_CLASS_OF_DEVICE 0x0D
#define BLE_GAP_AD_TYPE_SIMPLE_PAIRING_HASH_C 0x0E
#define BLE_GAP_AD_TYPE_SIMPLE_PAIRING_RANDOMIZER_R 0x0F
#define BLE_GAP_AD_TYPE_SECURITY_MANAGER_TK_VALUE 0x10
#define BLE_GAP_AD_TYPE_SECURITY_MANAGER_OOB_FLAGS 0x11
#define BLE_GAP_AD_TYPE_SLAVE_CONNECTION_INTERVAL_RANGE 0x12
#define BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_16BIT 0x14
#define BLE_GAP_AD_TYPE_SOLICITED_SERVICE_UUIDS_128BIT 0x15
#define BLE_GAP_AD_TYPE_SERVICE_DATA 0x16
#define BLE_GAP_AD_TYPE_PUBLIC_TARGET_ADDRESS 0x17
#define BLE_GAP_AD_TYPE_RANDOM_TARGET_ADDRESS 0x18
#define BLE_GAP_AD_TYPE_APPEARANCE 0x19
#define BLE_GAP_AD_TYPE_MANUFACTURER_SPECIFIC_DATA 0xFF
```

后面的 0B 说明这一段数据的长度为 11 个字节，即 09 4E 6F 72 64 69 63 5F 48 52 4D 查上面的表 09 指明 AD type 为完整的本地名称。 我定义的为”Nordic_HRM”十六进制就是 4E 6F 72 64 69 63 5F 48 52 4D。

后面 03 代表接着的一小段数据位三个字节 那么后面的 19 34 12 都属于这一小段 19 表示”外观” 34 12 代表外观。（外观特性是 SIG 定义的一组值，用来表示设备是普通手机，手环什么的）。

再后面是 02 则 01 06 属于这段 01 代表 FLAG ，flag 说明了物理连接功能，比如有限发现模式，不支持经典蓝牙等。

bit 0: LE 有限发现模式

bit 1: LE 普通发现模式

bit 2: 不支持 BR/EDR

bit 3: 对 Same Device Capable(Controller) 同时支持 BLE 和 BR/EDR

bit 4: 对 Same Device Capable(Host) 同时支持 BLE 和 BR/EDR

bit 5…7: 预留

即 06 数据说明了设备的连接功能为普通发现模式并且不支持经典蓝牙。

07 表明 03 0D 18 0F 18 0A 18 属于这一段 03 查上面的表知道后面的数据表示完整 16bit uuid 列表

```
ble_uuid_t adv_uuids[] =
{
{0x180D, BLE_UUID_TYPE_BLE},
{0x180F, BLE_UUID_TYPE_BLE},
{0x180A, BLE_UUID_TYPE_BLE}
};
```

就是我设置的广播 UUID， 39 FE 57 表示 CRC 校验。

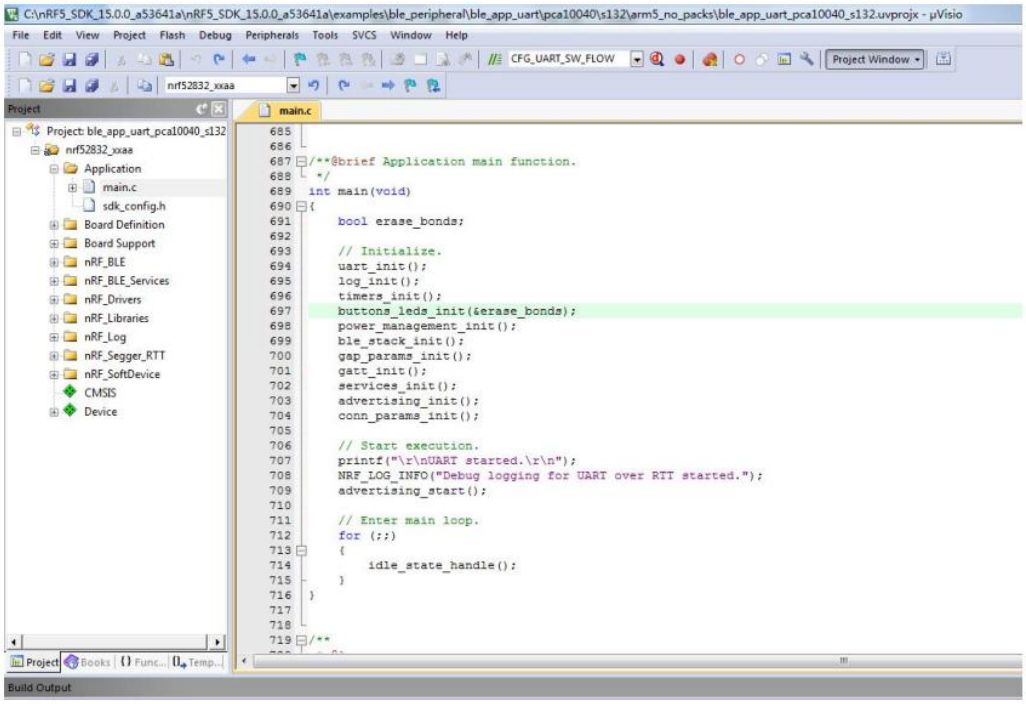
4. SDK 中蓝牙部分的程序结构

NORDIC 的官方 SDK 没有框架依耐性，这个跟 TI 不一样。 我们知道，TI 的 BLE SDK 中有一个操作系统抽象层(OSAL)，工程师在开发的时候需要按照他的方式去创建任务，否则可能会产生一些未知

问题。而 NORDIC 的 SDK 本质上只是提供了各种调用的接口给工程师使用，而跟蓝牙相关的功能都在协议栈中运行。初始化协议栈、初始化硬件功能模块、广播、发起连接这些操作我们都可以自己控制。不过我们最好遵循官方的一些例程，开发自己的应用。

开发 nRF5X 系列 SOC,一般都有固定的流程,下面我们以官方例程中 BLE_APP_UART 为例,来说说 SDK 的程序结构。

main 函数都是类似如下的步骤：




```
power_management_init(); //功耗管理初始化，必须有
ble_stack_init() // 协议栈初始化，必须有
gap_params_init(); // GAP 参数的初始化，必须有
gatt_init(); // GATT 初始化，必须有
services_init(); // 蓝牙服务的建立，不同的服务细节不同但是大体过程一致。
advertising_init(); // 广播数据初始化，必须有
conn_params_init(); //连接参数初始化，如果连接上以后不需要更新连接参数，可
//不要
```

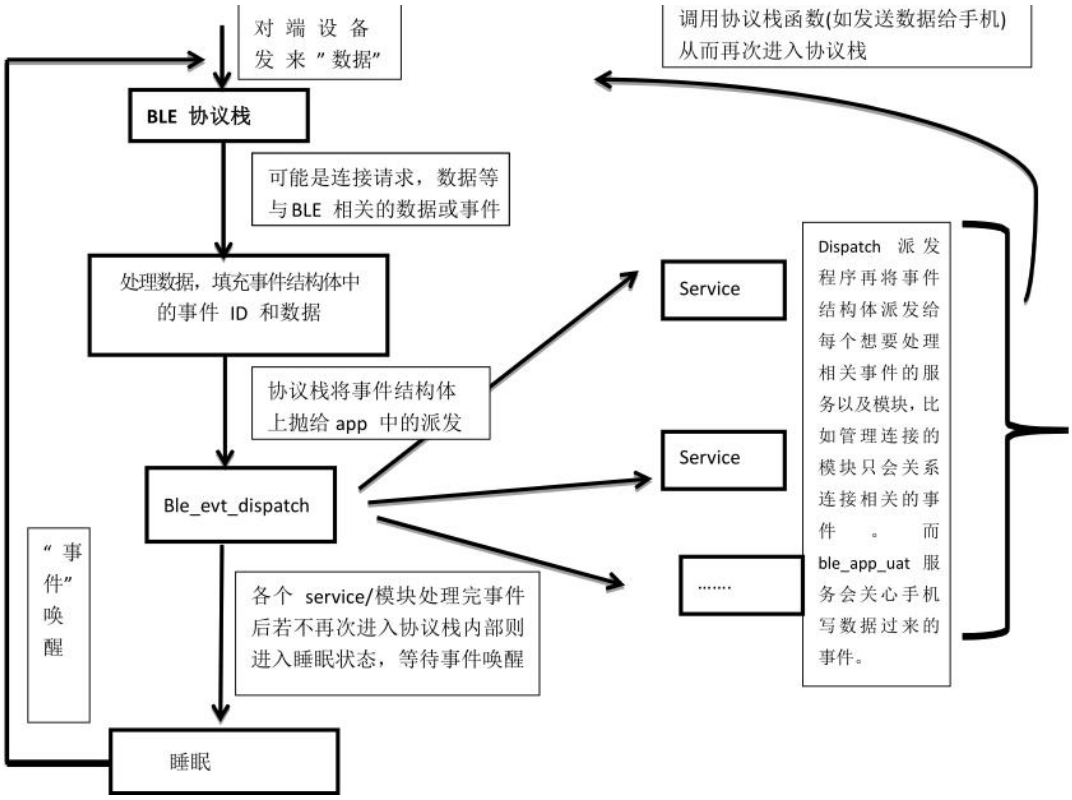
从以上可以得知，必须的函数只有 6 个而已。大家可以试下，将其他代码全都去掉，只要留这 6 个函数，设备一样可以运行；手机也能搜到这个蓝牙设备，并且可以正常通讯。

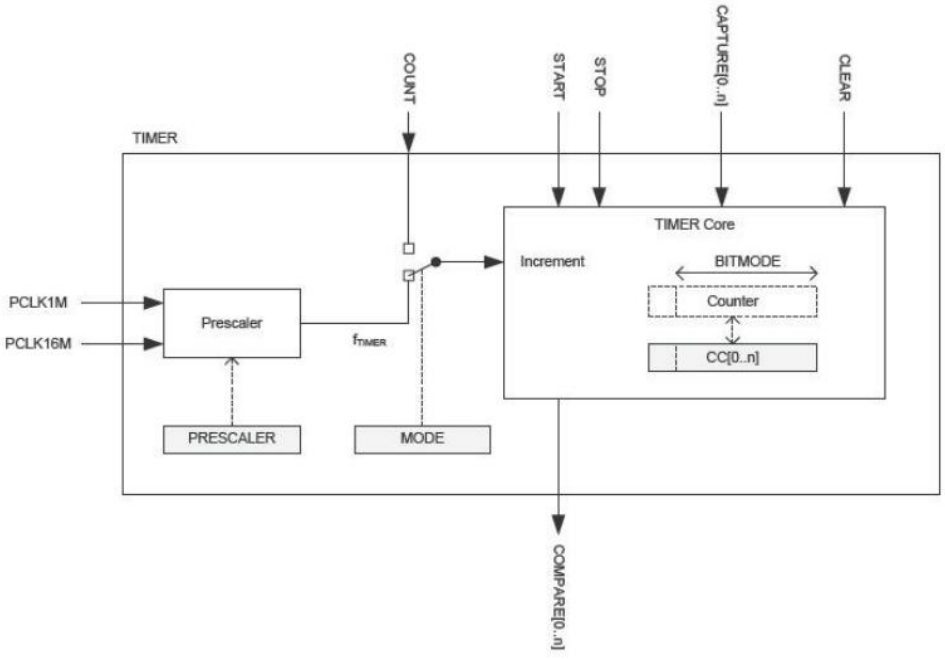
主程序进行过初始化之后，会进入一个死循环，执行性周而复始的事情。为降低功耗，SDK 在死循环中加入了低功耗处理的代码 idle_state_handle(),nRF5X 芯片在执行完工作以后，会自动进入休眠模式。

5. 外设 TIMER

nRF52832 系列 SOC 有五个 TIMER,对应的编号为 TIMER0-TIMER4。5 个 TIMER 都可以分别工作在定时模式和计数模式。

5.1. TIMER 的结构





定时器方框图

TIMER 有以下几部分组成：

1) 计数时钟源。有 1M、16M 2 种时钟源。

2) 时钟源分频器。用来设置分频，范

围是 2 的 0-9 次方。

3) 定时/计数模式选择。用来配置 TIMER 工作在定时器模式还是计数器模式。

4) 定时/计数器位数。8 位、16 位、24 位、32 位可选择。

5) CC[n]寄存器。CC 是捕捉、比较的缩写。CC 寄存器有 6 个。当执行 CAPUTURE TASK(捕捉任务)的时候，当前内部计数器的值将会立刻被拷贝到 CC 寄存器。

6) 各任务、事件等。

TIMER 有两种工作模式：定时模式和计数模式。当处于定时模式，我们称之为定时器；当处于计数模式，我们称之为计数器。2 种模式都可以通过 START 任务来启动；STOP 任务来停止。

执行 STOP 任务停止后的定时器，可以通过重新执行 START 任务来启动它。当 TIMER 处于定时模式的时候，TIMER 内部计数器在 FTIMER 时钟每个脉冲来临的时候计数一次，FTIMER 时钟的频率可以按以下公式计算：

$$f_{TIMER} = 16\ MHz\ /\ (2^{PRESCALER})$$

当内部计数器的值跟预先设置到 CC[n](n=1-5)里面的值相等的时候会触发相对应的 COMPARE[n]事件，如果我们使能了中断，则会产生中断。

当 TIMER 处于计数模式的时候，TIMER 内部计数器在 COUNT 引脚上每个脉冲来临的时候计数一次。

3 个任务：

1) START:启动定时/计数器

2) STOP:停止定时/计数器

3) SHUTDOWN:让定时/计数器掉电，后续无法通过 START 来启动该定时/计数器，除非重新复位。

注意事项：

1) 定时器/计数器只能在它们已经被停止的状态下才可以进行配置，否则会造成不可以预知的后果。

2) 当处于定时模式的时候，假如<=1MHZ，为节能，定时器会自动从 PCLK1M 时钟源获取计数脉冲，而不会使用 PCLK16M 时钟源。

5.2. 各寄存器的介绍

24.5.1 SHORTS

Address offset: 0x200

Shortcut register

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				L K J I H G F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0_CLEAR			Shortcut between COMPARE[0] event and CLEAR task
					See *EVENTS_COMPARE[0]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
B	RW	COMPARE1_CLEAR			Shortcut between COMPARE[1] event and CLEAR task
					See *EVENTS_COMPARE[1]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
C	RW	COMPARE2_CLEAR			Shortcut between COMPARE[2] event and CLEAR task
					See *EVENTS_COMPARE[2]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
D	RW	COMPARE3_CLEAR			Shortcut between COMPARE[3] event and CLEAR task
					See *EVENTS_COMPARE[3]* and *TASKS_CLEAR*
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
E	RW	COMPARE4_CLEAR			Shortcut between COMPARE[4] event and CLEAR task
					See *EVENTS_COMPARE[4]* and *TASKS_CLEAR*

该寄存器可以把 TASK 和 EVENT 联系起来。比如 COMPARE[n]_CLEAR 可以在内部计数器等于 CC[n]的时候清除内部计数器的计数值。

24.5.2 INTENSET

Address offset: 0x304

Enable interrupt

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0			Write '1' to Enable interrupt for COMPARE[0] event
					See *EVENTS_COMPARE[0]*
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	COMPARE1			Write '1' to Enable interrupt for COMPARE[1] event
					See *EVENTS_COMPARE[1]*
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	COMPARE2			Write '1' to Enable interrupt for COMPARE[2] event

中断允许设定寄存器，当设定为“1”的时候，对应的 COMPARE[n]事件产生的时候，会产生 COMPARE[n]中断。

24.5.3 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				F E D C B A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE0			Write '1' to Disable interrupt for COMPARE[0] event
					See *EVENTS_COMPARE[0]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	COMPARE1			Write '1' to Disable interrupt for COMPARE[1] event
					See *EVENTS_COMPARE[1]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	COMPARE2			Write '1' to Disable interrupt for COMPARE[2] event
					See *EVENTS_COMPARE[2]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	COMPARE3			Write '1' to Disable interrupt for COMPARE[3] event
					See *EVENTS_COMPARE[3]*
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
E	RW	COMPARE4			Write '1' to Disable interrupt for COMPARE[4] event
					See *EVENTS_COMPARE[4]*
			Clear	1	Disable

关闭 COMPARE[n]中断。

24.5.4 MODE

Address offset: 0x504

Timer mode selection

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				A A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	MODE			Timer mode
			Timer	0	Select Timer mode
			Counter	1	Select Counter mode Deprecated
			LowPowerCounter	2	Select Low Power Counter mode

定时/计数模式设定。

24.5.5 BITMODE

Address offset: 0x508

Configure the number of bits used by the TIMER

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id				A A
Reset 0x00000000				0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Id	RW	Field	Value Id	Value	Description
A	RW	BITMODE			Timer bit width
			16Bit	0	16 bit timer bit width
			08Bit	1	8 bit timer bit width
			24Bit	2	24 bit timer bit width
			32Bit	3	32 bit timer bit width

TIMER 位数设定。

24.5.6 PRESCALER

Address offset: 0x510

Timer prescaler register

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id	A A A A
Reset 0x00000004	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Id	RW	Field	Value Id	Value	Description
A	RW	PRESCALER		[0..9]	Prescaler value

预分频寄存器。

24.5.8 CC[1]

Address offset: 0x544

Capture/Compare register 1

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Id	A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
Reset 0x00000000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Id	RW	Field	Value Id	Value	Description
A	RW	CC			Capture/Compare value Only the number of bits indicated by BITMODE will be used by the TIMER.

捕捉/比较寄存器。

5.3. 程序的设计

通过以上的介绍，大家对 TIMER 应该比较了解了。我们用 TIMER 的一般步骤是：

1.设定工作模式

2.设定预分频（计数器不适用）

3.设定 CC[n]寄存器的值

4.使能中断（中断模式）

5.启动 START 任务

6.COMPARE EVENT 到来，清除内部计数器的值，清除中断（中断模式）

修订历史

版本	修订日期	修订说明	维护人
1.0	2019-3-5	初始版本	All
1.1	2023-08-09	错误更正	Bin
1.2	2023-09-05	错误更正	Bin

关于我们

销售热线：4000-330-990 公司电话：028-61543675
技术支持：support@cdebyte.com 官方网站：www.ebyte.com
公司地址：四川省成都市高新西区西区大道 199 号 B2、B5 栋

成都亿佰特电子科技有限公司
Chengdu Ebyte Electronic Technology Co.,Ltd.