

PAN1080 RADIO TX 使用说明

PAN-CLT-VER-A0, Rev 1.1

PANCHIP

PanchipMicroelectronics

www.panchip.com

修订历史

版本	修订日期	描述
V1.0	2021-02-19	初始版本创建

目录

目录

第 1 章 测试目的	4
第 2 章 测试内容	5
2.1 测试内容	5
2.2 环境配置	5
第 3 章 测试说明	6
3.1 环境说明	6
3.2 测试结果	6
3.3 开发说明	7
3.3.1 2.4G 发射初始化.....	7
3.3.2 2.4G 中断处理.....	9
3.3.3 2.4G 配置切换.....	11

第1章 测试目的

1. 测试 PAN1080 2.4G TX 功能。

第2章 测试内容

2.1 测试内容

1. 此项目演示了 2.4G 发射端功能：每隔 200ms 发送一次 2.4G 数据包，长度 5 个字节。
2. 接收端（参考 [PAN1080 RADIO RX 使用说明.docx](#)）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。

2.2 环境配置

a) 环境要求

- board: pan108xxb5_evb
 - uart: 显示串口输出 log
 - PC 串口工具：Panchip Serial Assistant V0.0.006.exe
- 需要搭配一个运行“prf_rx”的板子一起使用。

b) 编译和烧录

项目位置：“03_MCU\mcu_samples\PRF_TX”。

用 keil 编译程序，用 j-link 烧录编译后的 hex 文件到 pan108xxb5_evb 板子中。

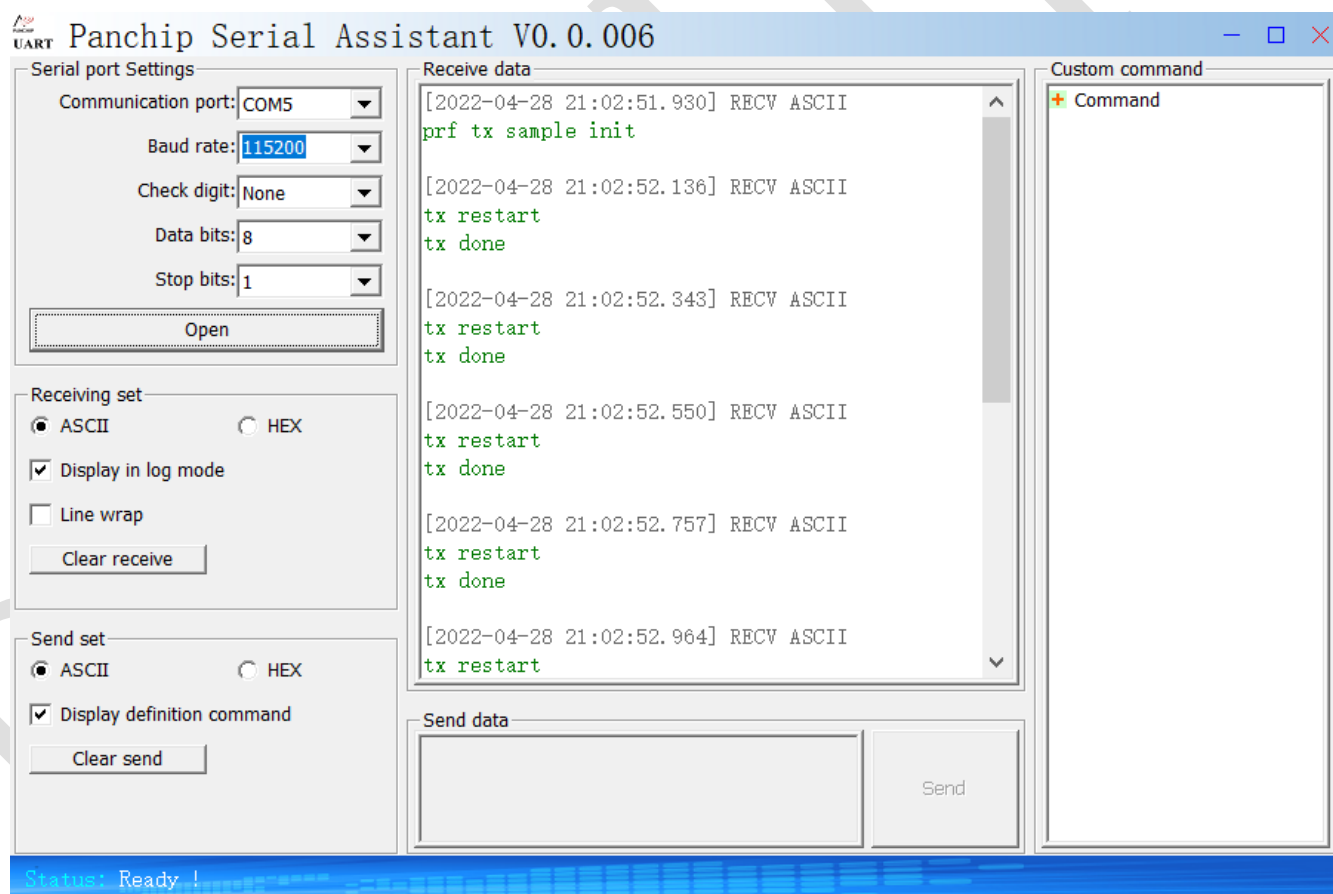
第3章 测试说明

3.1 环境说明

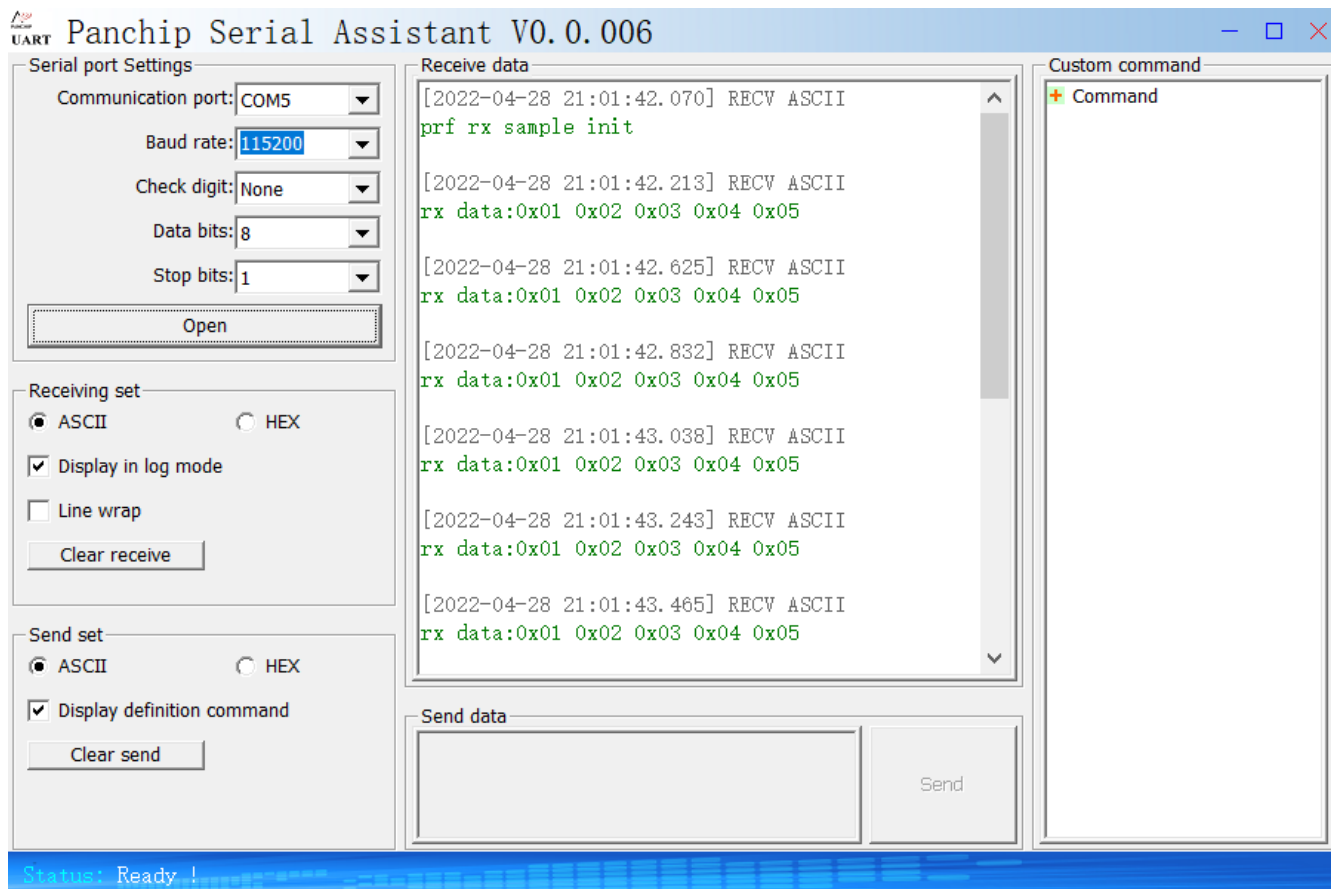
- a) 将接收端串口和发射端串口分别接到 PC 的 USB 端口上。
- b) 配置接收端和发送端（参考 [PAN1080_RADIO_RX 使用说明.docx](#)）。
- c) 观察 PC 串口工具的输出结果。

3.2 测试结果

- 1. 发射端输出结果：



2. 接收端输出结果:



3.3 开发说明

发射端启动了一个定时器,每隔 200ms 发送一包 2.4G 数据。发送成功串口输出 log“tx done”。

3.3.1 2.4G 发射初始化

设置发射频点 2450Mhz

```

1. pan_prf_config_t tx_config =
2. {
3.     .work_mode      = PRF_MODE_NORMAL,
4.     .chip_mode      = PRF_CHIP_MODE_SEL_XN297,
5.     .trx_mode       = PRF_TX_MODE,
6.     .phy            = PRF_PHY_1M,
7.     .crc            = PRF_CRC_SEL_NOCRC,
8.     .src            = PRF_SRC_SEL_NOSRC,
9.     .rx_timeout     = 50000,
10.    .rf_channel      = 2450,
11.    .tx_no_ack       = DISABLE,

```

```

12.     .nrf52_mode      = DISABLE,
13.     .rx_length      = 0,
14.     .addr_length     = 4,
15.     .crc_include_addr = ENABLE,
16.     .tx_trans_time   = 140,           //us
17.     .rx_trans_time   = 150,           //us
18.     .addr             = { 0x71, 0x76, 0x41, 0x76 },
19. };
20.
21. panchip_prf_payload_t tx_payload =
22. {
23.     .data_length      = 5,
24.     .data             = { 0x01, 0x02, 0x03, 0x04, 0x05 },
25. };
    
```

2. 4G 初始化配置说明如下：

初始化配置的结构体 “pan_prf_config_t”

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, no crc
prf_scramble_sel_t	src	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，配置范围 2402-2480Mhz
uint8_t	tx_no_ack	配置增强型模式下 tx 是否需要 ack
uint8_t	nrf52_mode	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	addr_length	接入地址长度配置，可配置为 3、4、5 字节
uint8_t	addr[5]	接入地址的内容
uint8_t	crc_include_addr	crc 是否包含接入地址（XN297 和 NRF24L01 必须使能）
uint16_t	tx_trans_time	增强型模式 tx 转 rx 的时间
uint16_t	rx_trans_time	增强型模式 rx 转 tx 的时间

prf_mode_t:

Type	Value	Description
PRF_MODE_NORMAL	0	普通型
PRF_MODE_ENHANCE	1	增强型

prf_chip_mode_sel_t:

Type	Value	Description
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_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_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

3.3.2 2.4G 中断处理

1. Tx 普通型触发的中断

Tx 完成后会触发 tx done 中断，其他中断不会执行，tx 中断执行完后 tx 流程就结束了。
Tx 中断接口如下：

```
1. void event_tx_fun(void)
2. {
3.     printf("tx done\n");
4. }
```

2. Tx 增强型触发的中断

Tx 完成后会触发 tx done 中断，tx 中断执行完后会进入 rx 模式，rx 有以下中断接口：

1) RX 收到数据

```
2) void event_rx_fun(void)
3) {
4)     panchip_prf_payload_t rx_payload;
5)
6)     rx_payload.data_length = panchip_prf_data_rec(&rx_payload);
7)     printf("rx data:");
8)     data_printk(rx_payload.data, rx_payload.data_length);
9) }
```

将收到的 ack 数据打印出来。

2) RX 接收超时

```
3) void event_rx_timeout_fun(void)
4) {
5)     printf("rx timeout\n");
6) }
```

超时时间内未收到数据则会触发该中断，超时时间就是初始化设置的时间。

3) RX 接收 CRC 错误

```
4) void event_crc_err_fun(void)
5) {
6)     printf("rx data crc err\n");
7) }
```

接收到数据但是 CRC 校验错误则会触发该中断，该中断也可将收到的 ack 数据打印出来。CRC 未使能该中断不会触发。

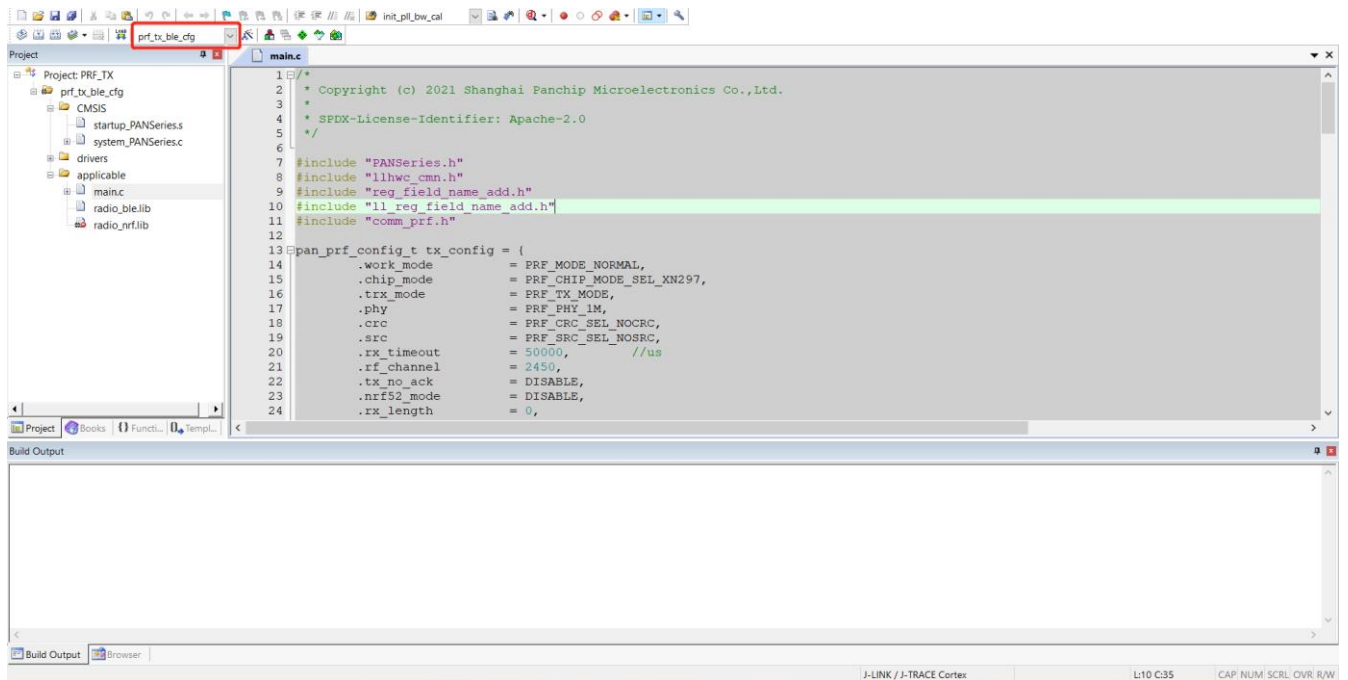
4) RX 接收 PID 错误

```
5) void event_pid_err_fun(void)
6) {
7)
8) }
```

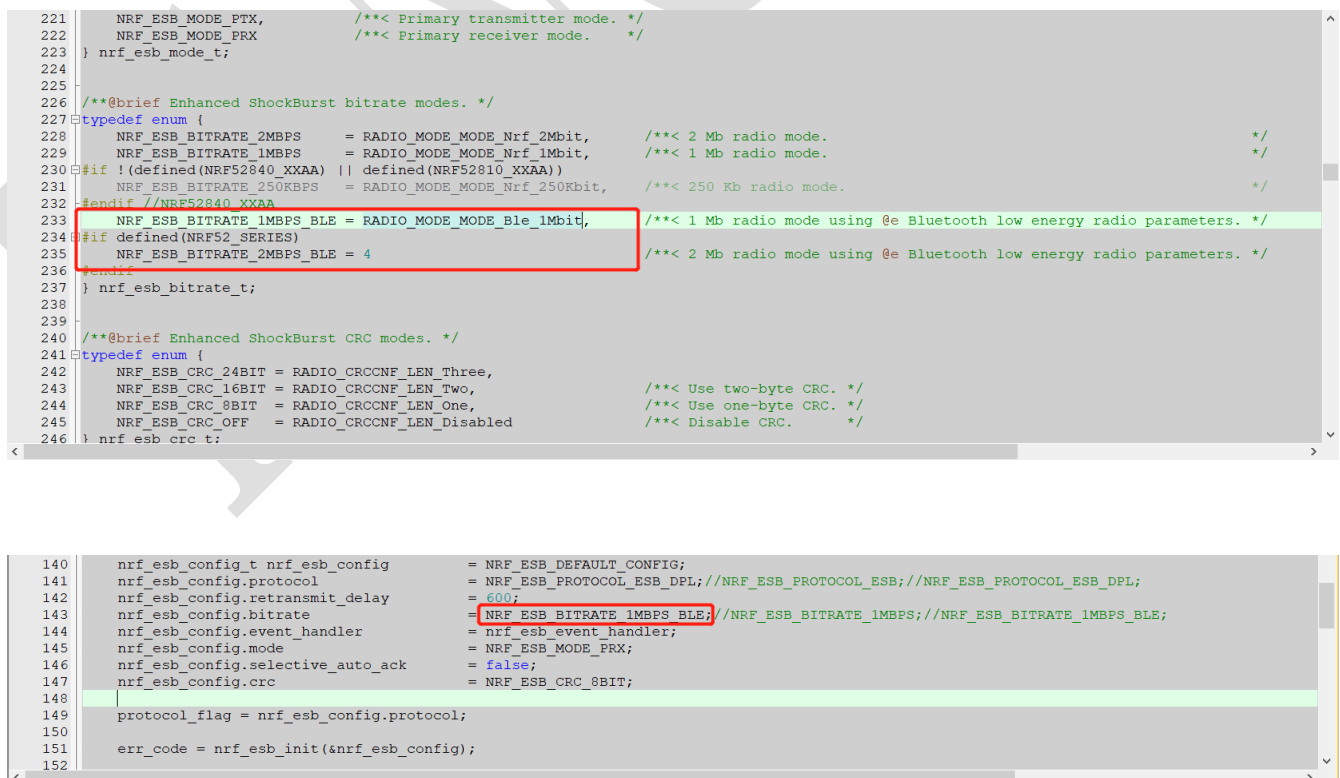
当接收重发包的 pid 异常时会触发该中断，该中断可将收到的 ack 数据打印出来。

3.3.3 2.4G 配置切换

软件默认使用的是 BLE 模式配置，如下图所示：

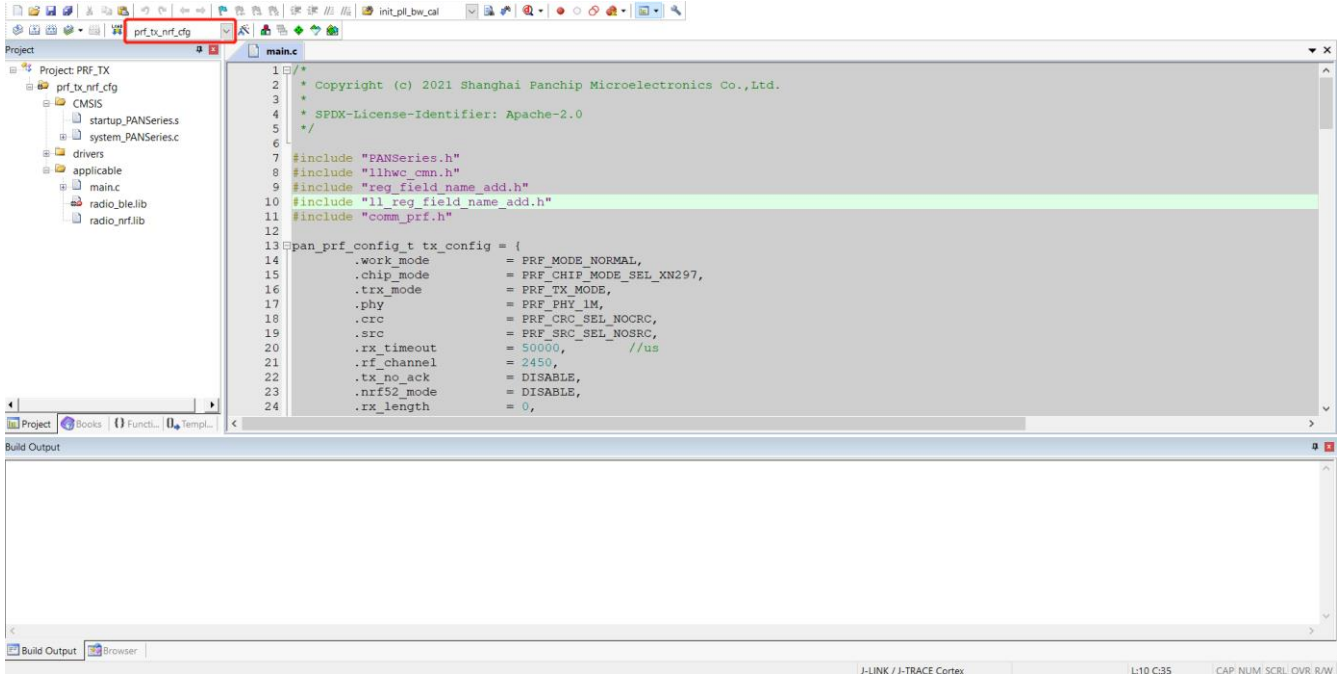


此模式配置下的 RF 参数中的 deviation 是 250K，此模式下可与 XN297、NRF52840 或 NRF52832 通信，NRF 代码中需要修改的配置如下图所示：



此模式配置下才能确保无干扰的环境下通信不丢包。

软件切换 NRF 模式配置如下图所示：

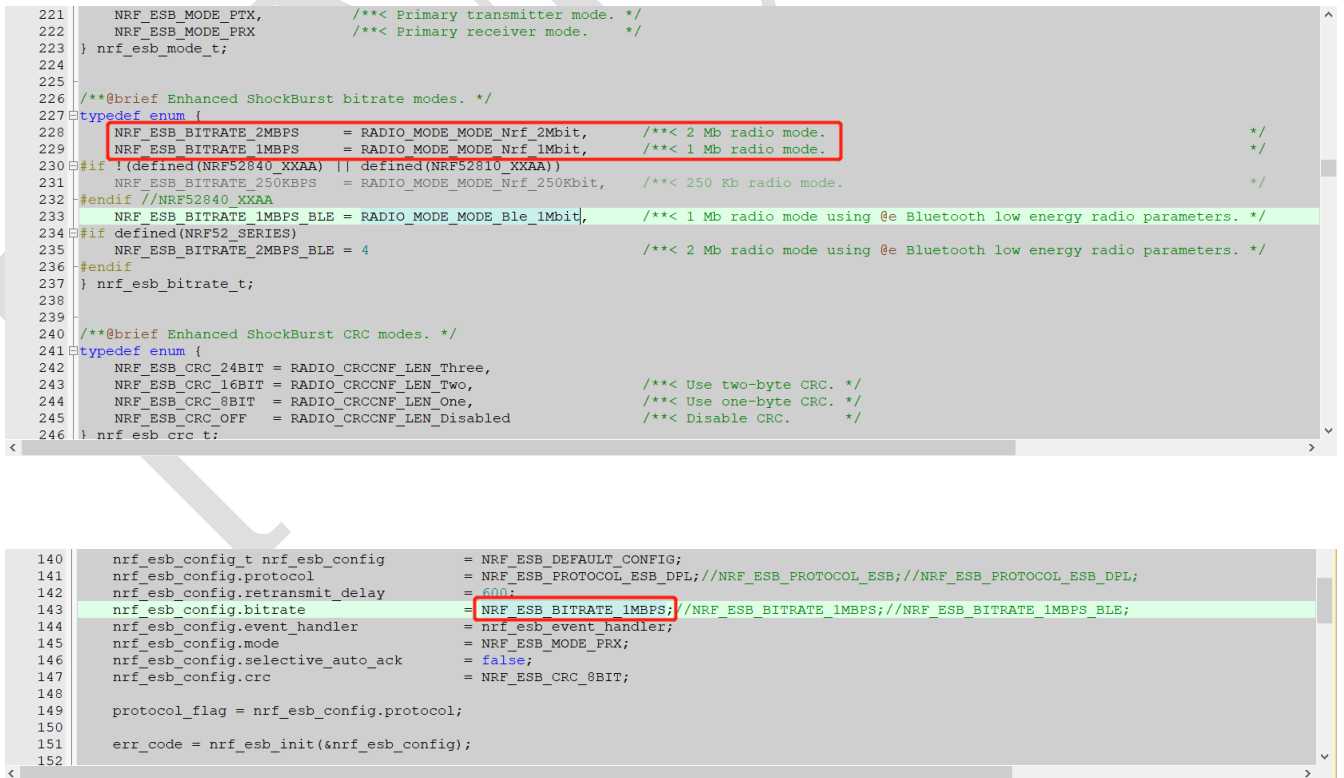


```

1 1/*
2 2 * Copyright (c) 2021 Shanghai Panchip Microelectronics Co.,Ltd.
3 3 *
4 4 * SPDX-License-Identifier: Apache-2.0
5 5 */
6 6
7 7 #include "PANSeries.h"
8 8 #include "lhwc_cmn.h"
9 9 #include "reg_field_name_add.h"
10 10 #include "ll_reg_field_name_add.h"
11 11 #include "comm_prf.h"
12 12
13 13 pan_prf_config_t tx_config = {
14 14     .work_mode           = PRF_MODE_NORMAL,
15 15     .chip_mode           = PRF_CHIP_MODE_SEL_XN297,
16 16     .trx_mode            = PRF_TX_MODE,
17 17     .phy                 = PRF_PHY_1M,
18 18     .crc                 = PRF_CRC_SEL_NOCRC,
19 19     .src                 = PRF_SRC_SEL_NOSRC,
20 20     .rx_timeout          = 50000,          //us
21 21     .rf_channel          = 2450,
22 22     .tx_no_ack           = DISABLE,
23 23     .nrf52_mode          = DISABLE,
24 24     .rx_length           = 0,

```

此模式配置下的 RF 参数中的 deviation 是 175K，此模式下可与 NRF24L01、NRF52840 或 NRF52832 通信，NRF 代码中需要修改的配置如下图所示：



```

221 NRF_ESB_MODE_PTX,          /**< Primary transmitter mode. */
222 NRF_ESB_MODE_PRX         /**< Primary receiver mode. */
223 } nrf_esb_mode_t;
224
225
226 /**@brief Enhanced ShockBurst bitrate modes. */
227 #typedef enum {
228     NRF_ESB_BITRATE_2MBPS = RADIO_MODE_MODE_Nrf_2Mbit,          /**< 2 Mb radio mode. */
229     NRF_ESB_BITRATE_1MBPS = RADIO_MODE_MODE_Nrf_1Mbit,          /**< 1 Mb radio mode. */
230 } #if !(defined(NRF52840_XXAA) || defined(NRF52810_XXAA))
231     NRF_ESB_BITRATE_250KBPS = RADIO_MODE_MODE_Nrf_250Kbit,      /**< 250 Kb radio mode. */
232 #endif //NRF52840_XXAA
233     NRF_ESB_BITRATE_1MBPS_BLE = RADIO_MODE_MODE_Ble_1Mbit,      /**< 1 Mb radio mode using @e Bluetooth low energy radio parameters. */
234 #if defined(NRF52_SERIES)
235     NRF_ESB_BITRATE_2MBPS_BLE = 4                                /**< 2 Mb radio mode using @e Bluetooth low energy radio parameters. */
236 #endif
237 } nrf_esb_bitrate_t;
238
239
240 /**@brief Enhanced ShockBurst CRC modes. */
241 #typedef enum {
242     NRF_ESB_CRC_24BIT = RADIO_CRC_CFG_LEN_Three,
243     NRF_ESB_CRC_16BIT = RADIO_CRC_CFG_LEN_Two,                    /**< Use two-byte CRC. */
244     NRF_ESB_CRC_8BIT = RADIO_CRC_CFG_LEN_One,                     /**< Use one-byte CRC. */
245     NRF_ESB_CRC_OFF = RADIO_CRC_CFG_LEN_Disabled,                 /**< Disable CRC. */
246 } nrf_esb_crc_t;

```

```

140 nrf_esb_config_t nrf_esb_config = NRF_ESB_DEFAULT_CONFIG;
141 nrf_esb_config.protocol         = NRF_ESB_PROTOCOL_ESB_DPL; //NRF_ESB_PROTOCOL_ESB; //NRF_ESB_PROTOCOL_ESB_DPL;
142 nrf_esb_config.retransmit_delay = 600;
143 nrf_esb_config.bitrate          = NRF_ESB_BITRATE_1MBPS; //NRF_ESB_BITRATE_1MBPS; //NRF_ESB_BITRATE_1MBPS_BLE;
144 nrf_esb_config.event_handler    = nrf_esb_event_handler;
145 nrf_esb_config.mode              = NRF_ESB_MODE_PRX;
146 nrf_esb_config.selective_auto_ack = false;
147 nrf_esb_config.crc               = NRF_ESB_CRC_8BIT;
148
149 protocol_flag = nrf_esb_config.protocol;
150
151 err_code = nrf_esb_init(&nrf_esb_config);
152

```