Panchip Microelectronics Co., Ltd.

PAN3020 SDK 用户指南

当前版本: 1.0

发布日期: 2021.09

上海磐启微电子有限公司

地址：上海张江高科技园区盛夏路 666 号 D 栋 3 楼

联系电话：021-50802371

网址：http://www.panchip.com

文档说明

由于版本升级或存在其他原因，本文档内容会不定期进行更新。除非另有约定，本文档内容仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标

磐启是磐启微电子有限公司的商标。本文档中提及的其他名称是其各自所有者的商标/注册商标。

免责声明

本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，磐启微电子有限公司对本文档内容不做任何明示或暗示的声明或保证。

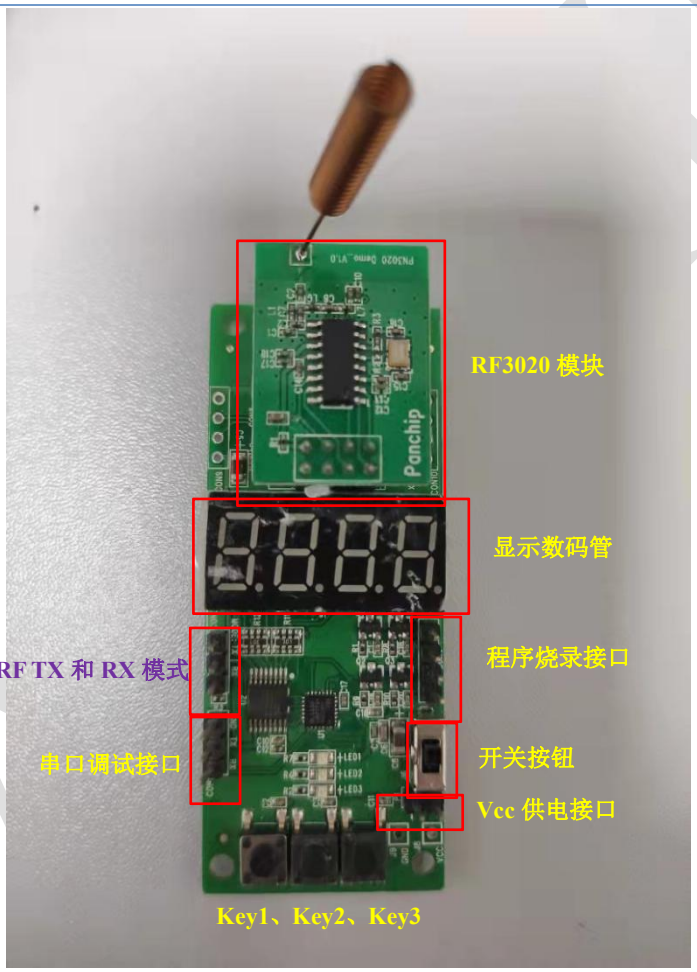
修订历史

版本	修订时间	描述
V1.0	2021.09.28	初始版本创建

目 录

1 概述

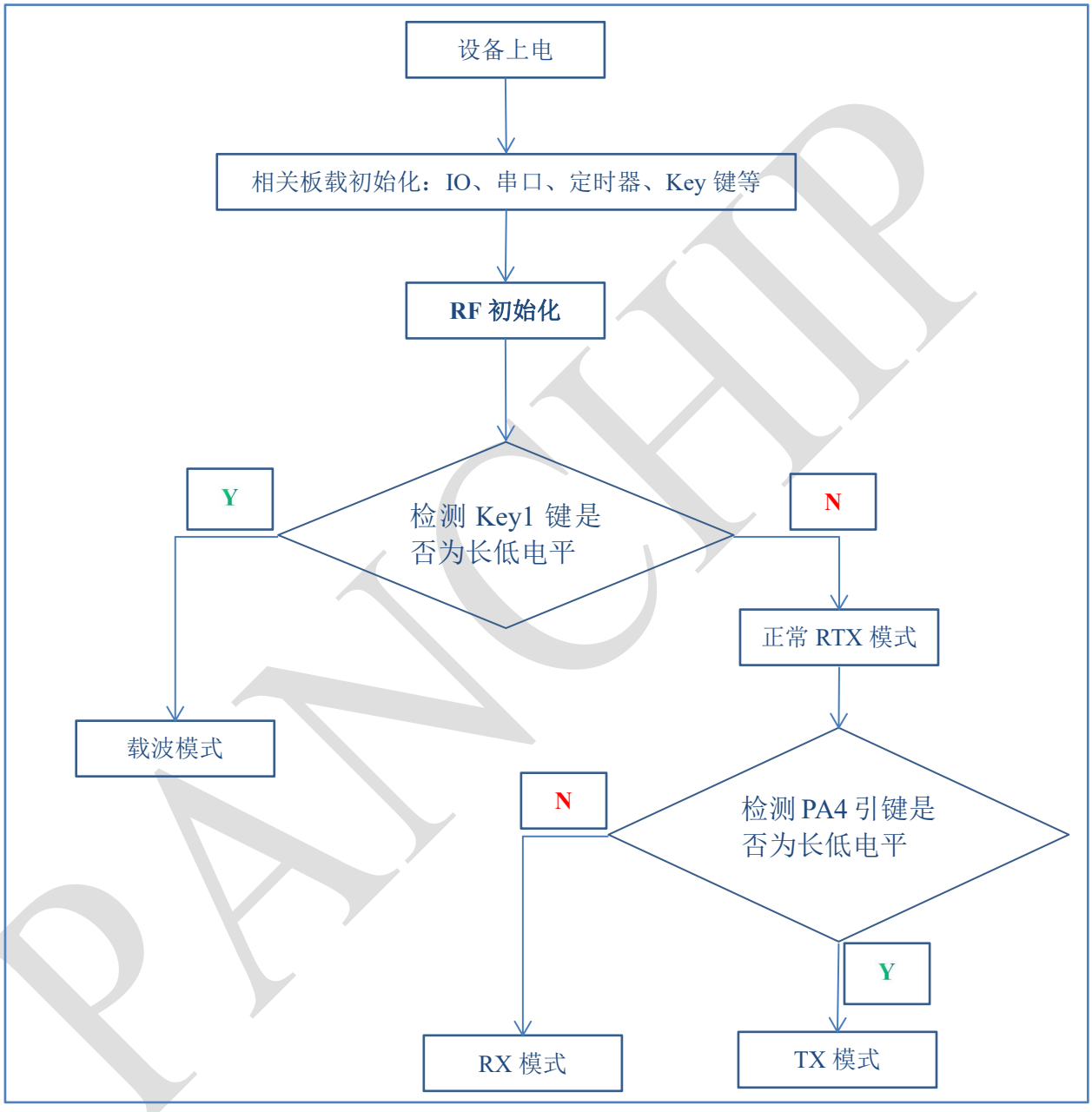
本文档主要基于 STM8 的 PAN3020 开发板的硬件环境介绍 SDK 接口函数的使用，RF 的初始化、Rx、Tx 的使用，该 SDK 中主要相关的文件有：RF.c、RF.h、RFAPI.H、RFAPI.c 等。



2 SDK使用流程

本章节基于 PAN3020 开发板，并基于此开发板的硬件环境，描述 SDK 的基本使用流程，包括初始化及流程图的介绍。

2.1 工作模式初始化



根据以上流程图，可判断，在当前的 PAN3020 中存在 3 中工作模式，通过按键、控制 PA4 的状态来控制或进入到不同的模式中。

2.2 RF 初始化

2.2.1 接口函数

```
RF_Init;
```

2.2.2 实现功能

1、 清空 TX、RX FIFO；

2、 清除 IRQ、RX、TX 状态位；

3、 设置 RX 和 TX 的收发地址；

4、 设置速率和频点；

5、 设置功率；

……

2.2.3 使用方法

芯片初次上电时调用接口函数。

2.3 参数配置流程

2.3.1 接口函数

```
RF_SetFreq_Datarate
RF_SetPower
```

2.3.2 实现功能

设置频率、Code_Rate、带宽、发射功率

2.3.3 使用方法

芯片初始化时调用接口函数，已封装进 RF_init 函数中。

2.4 载波流程

2.4.1 接口函数

```
RF_Carrier
```

2.4.2 实现功能

正常模式下，发射数据 PA 打开，数据发完后，PA 关闭，进入载波状态后，保持 PA 常开的状态；

2.4.3 使用方法

长按 Key1 键，并上电，确保 RF 进入到载波状态，可通过数码管显示“Ch.00”进行判断。

芯片初始化和配置参数完成后调用该接口函数。

完整应用流程参照第四章节 SDK 示例。

2.5 发送流程

2.5.1 接口函数

```
RF_TxMode();
```

2.5.2 实现功能

1、切换状态

通过 Status_Init()判断 RF 进入 Tx 模式；

```
Status_Init();
```

2、设置发射模式

```
RF_TxMode
```

3、TX 处理事件

```
APP_TX_Event();
```

4、设置 Tx 发包模式：单包、1000 包、连续；

```
APP_TXMODE_Swtich
```

5、准备发射数据包，发射数据；

```
RFAPI_StateMachine();
RFAPI_TxState();
RFAPI_PacketData();
RF_Tx_TransmintData();
```

2.5.3 使用方法

确认 RF 处于 TX 模式状态；

芯片初始化并且配置参数完成后调用接口函数。

调用相关接口函数，Tx 发包模式、准备数据、调用 Tx 发射数据；

完整应用流程参照第四章节 SDK 示例。

2.6 接收流程

2.6.1 接口函数

```
RF_RxMode();
```

2.6.2 实现功能

1、切换状态

通过 Status_Init()判断 RF 进入 Rx 模式；

```
Status_Init();
```

2、设置接收模式

```
RF_RxMode
```

3、RX 处理事件

```
APP_RX_Event();
```

4、进入接收状态，处理接收 Rx 数据；

```
RFAPI_StateMachine();
RFAPI_RXState();
RF_DumpRxData();
```

2.6.3 使用方法

确认 RF 处于 RX 模式状态；

芯片初始化并且配置参数完成后调用接口函数。

调用相关接口函数，使 RF 处于 RX 模式，并处理接收数据；

完整应用流程参照第四章节 SDK 示例。

3 SDK接口描述

本章节详细描述 SDK《RF.c》中提供的用户接口函数。

3.1 SPI_init

句法

```
void    SPI_init(void)
```

目的

用于 RF SPI 引脚的初始化；

参数

无

返回值

无

3.2 SPI_RW

句法

```
uint8_t SPI_RW( uint8_t   R_REG)
```

目的

SPI读写寄存器数据，写入一个字节的同时，读取一个字节的数据

参数

R_REG：SPI 操作的 RF 的寄存器；

返回值

返回读取到的一个字节的数据；

3.3 RF_WriteReg

句法

```
void RF_WriteReg( uint8_t reg,   uint8_t wdata)
```

目的

向寄存器中写入数据；

参数

reg:写入数据的寄存器；

wdata：写入的数据；

返回值

无

3.4 RF_ReadReg

句法

```
uint8_t RF_ReadReg( uint8_t reg)
```

目的

读取寄存器 reg 的数值

参数

reg：读取数据的寄存器

返回值

寄存器 reg 中读取到的数值

3.5 RF_WriteBuf

句法

```
void    RF_WriteBuf( uint8_t reg,   uint8_t *pBuf,   uint8_t length)
```

目的

往寄存器中写入指定 length 的 buffer

参数

Reg：写入数据的 reg 寄存器；

pBuf：要写入的数据；

Length：写入数据的长度；

返回值

无

3.6 RF_ReadBuf

句法

```
void    RF_ReadBuf( uint8_t reg,   uint8_t *pBuf,   uint8_t length)
```

目的

读取 RF 寄存器中的 Buffer

参数

Reg：存放 payload 的寄存器；

pBuf：read buffer

Length：buffer 的长度

返回值

无

3.7 RF_TxMode

句法

```
void    RF_TxMode(void)
```

目的

RF 的 TX 模式；

参数

无

返回值

无

3.8 RF_RxMode

句法

```
void    RF_RxMode(void)
```

目的

RF 的 RX 模式；

参数

无

返回值

无

3.9 RF_GetStatus

句法

```
uint8_t RF_GetStatus(void)
```

目的

获取 RF 的状态信息，主要用于 RF 的 IRQ 信号；

参数

无

返回值

无

3.10 RF_ClearStatus

句法

```
void    RF_ClearStatus(void)
```

目的

参数

清除 RF 的 IRQ；

返回值

无

3.11 RF_ClearFIFO

句法

```
void    RF_ClearFIFO(void)
```

目的

清除 RX、TX 的 FIFO；

参数

无

返回值

无

3.12 RF_SetPower

句法

```
void        RF_SetPower( uint8_t * p,uint8_t power)
```

目的

设置 RF 的功率（单位：dbm），可选参考数据有：

RF18DBM RF17DBM RF16DBM RF15DBM RF13DBM RF12DBM RF11DBM RF10DBM RF9DBM RF8DBM RF7DBM RF6DBM RF5DBM RF4DBM RF2DBM RF1DBM RF0DBM RFN1DBM RFN6DBM RFN15DBM

参数

P：指向调用的 RF_call_data；

Power：需要设定的功率的参数；

返回值

无

3.13 RF_SetChannel

句法

```
void        RF_SetChannel(uint8_t Fb,uint8_t Fc )
```

目的

设置 RF 频点

参数

Fb：设置频率的整数位；

Fc：设置频率的小数位

返回值

无

3.14 RF_SetFreq_Datarate

句法

```
void RF_SetFreq_Datarate(double freq,uint8_t fre_band)
```

目的

设置 RF 的频率和速率，

参数

freq：RF 的频率值（单位 MHz，浮点型数据）

band：频点的 BAND，可选范围为 B315MHz、B433MHz、B868MHz、B915MHz

通过 freq 计算出在不同 BAND 下的 Fb,Fc 值，并传参至 RF_SetChannel 函数从而对频率进行设定；

返回值

无

3.15 RF_CalVco

句法

```
void RF_CalVco( uint8_t * ptr_Dem_cal1)
```

目的

参数

ptr_Dem_cal1：

返回值

无

3.16 RF_DumpRxData

句法

```
uint8_t RF_DumpRxData( uint8_t *ucPayload,    uint8_t length)
```

目的

读取接收到的 RX 数据

参数

ucPayload：存储读取到的数据的 Buffer

length：读取到的 Rx 数据长度

返回值

0：没有接收到数据

1：读取接收到的数据成功

3.17 RF_Tx_CheckResult

句法

```
void       RF_Tx_CheckResult(uint8_t *ucAckPayload,    uint8_t length)
```

目的

用于检查 Tx 的结果：

1、普通型发送完成 或 增强型发送成功；

2、增强型发送成功且收到 payload；

3、增强型发送超时失败；

参数

ucAckPayload：用于增强型发送成功，读取 Payload 地址；

length：读取数据长度

返回值

无

3.18 RF_Tx_TransmintData

句法

```
void    RF_Tx_TransmintData( uint8_t *ucTXPayload,   uint8_t length)
```

目的

用于 Tx 数据的发射

参数

ucTXPayload：发送数据的地址；

length：发送数据的长度；

返回值

无

3.19 RF_Init

句法

```
void    RF_Init(void)
```

目的

RF 的初始化：包含 RF SPI 初始化、RF power 设置、RF 频点和速率初始化

参数

无

返回值

无

3.20 RF_Carrier

句法

```
void    RF_Carrier()
```

目的

用于载波模式下的相关配置；

参数

无

返回值

无

3.21 print_reg_val

句法

```
void    print_reg_val(void)
```

目的

打印、显示寄存器的数值

参数

无

返回值

无

3.22 print_RTX_buffer

句法

```
void      print_RTX_buffer(uint8_t *ucPayload,   uint8_t length)
```

目的

用于打印、显示 Tx 或者 Rx 的数据，

参数

ucPayload：用于接收到的数据或发射数据的地址；

length：接收或发射数据的长度；

返回值

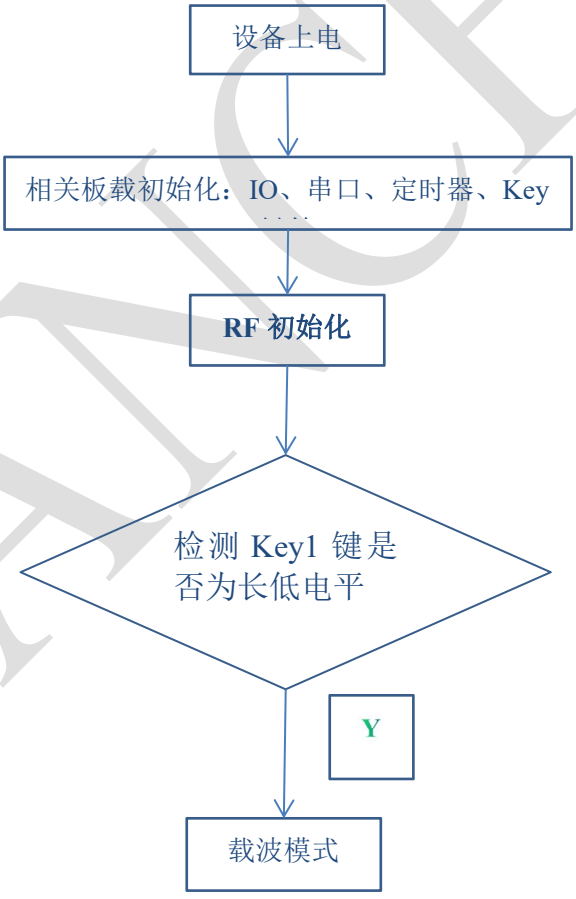
无

4 SDK示例

本章节详细描述 PAN3020 SDK 中几种模式的实现过程。

4.1 载波模式

4.1.1 载波模式实现流程图



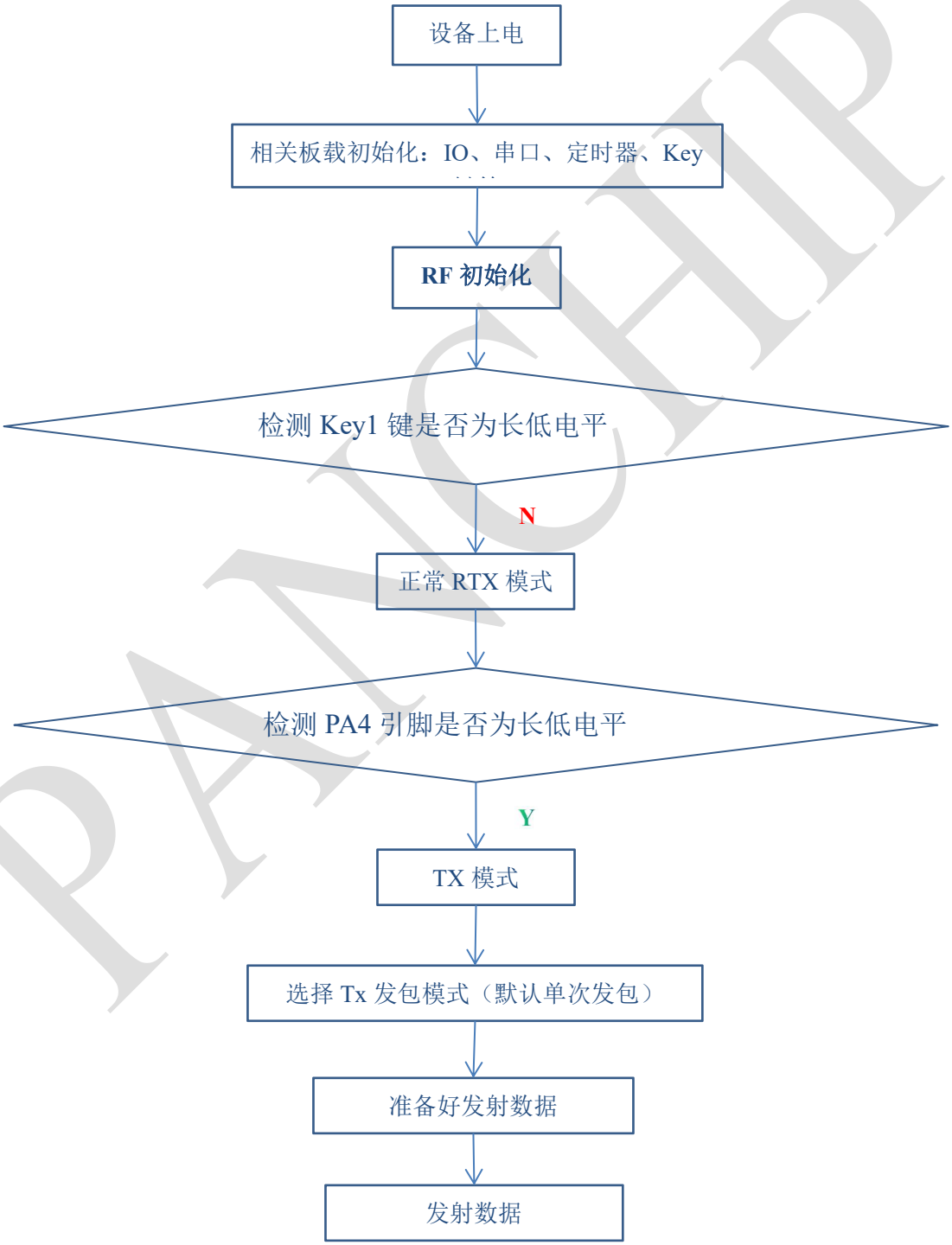
4.1.2 代码实现

```
BIOS_Init
{
    ……
    RF_Init();
    Status_Init();    //长按 Key1 时，会判断 TEST_CARRIER_MODE 状态
    ……
}
RFAPI_TestMode()
{
    ……
    RF_Carrier();
    While(1)
    {
        ……//处理按键控制跳频、LED 显示灯等
    }
    ……
}
```

4.1.3 注意事项

4.2 Tx 模式

4.2.1 代码流程图(以默认的单包发送为例)



4.2.2 代码实现

```
BIOS_Init
{
    ……
    RF_Init();
    Status_Init();    //确认为 Tx 状态，硬件上短接 TX 与 COM 口引脚
    ……
}
Int main()
{
    BIOS_Init();
    While(1)
    {
        APP_StateMachine();
        RFAPI_StateMachine();
    }
}

APP_StateMachine()
{
    ……
    APP_TX_Normal()
    {
        ……
        APP_TX_Event()
        {
            ……
```

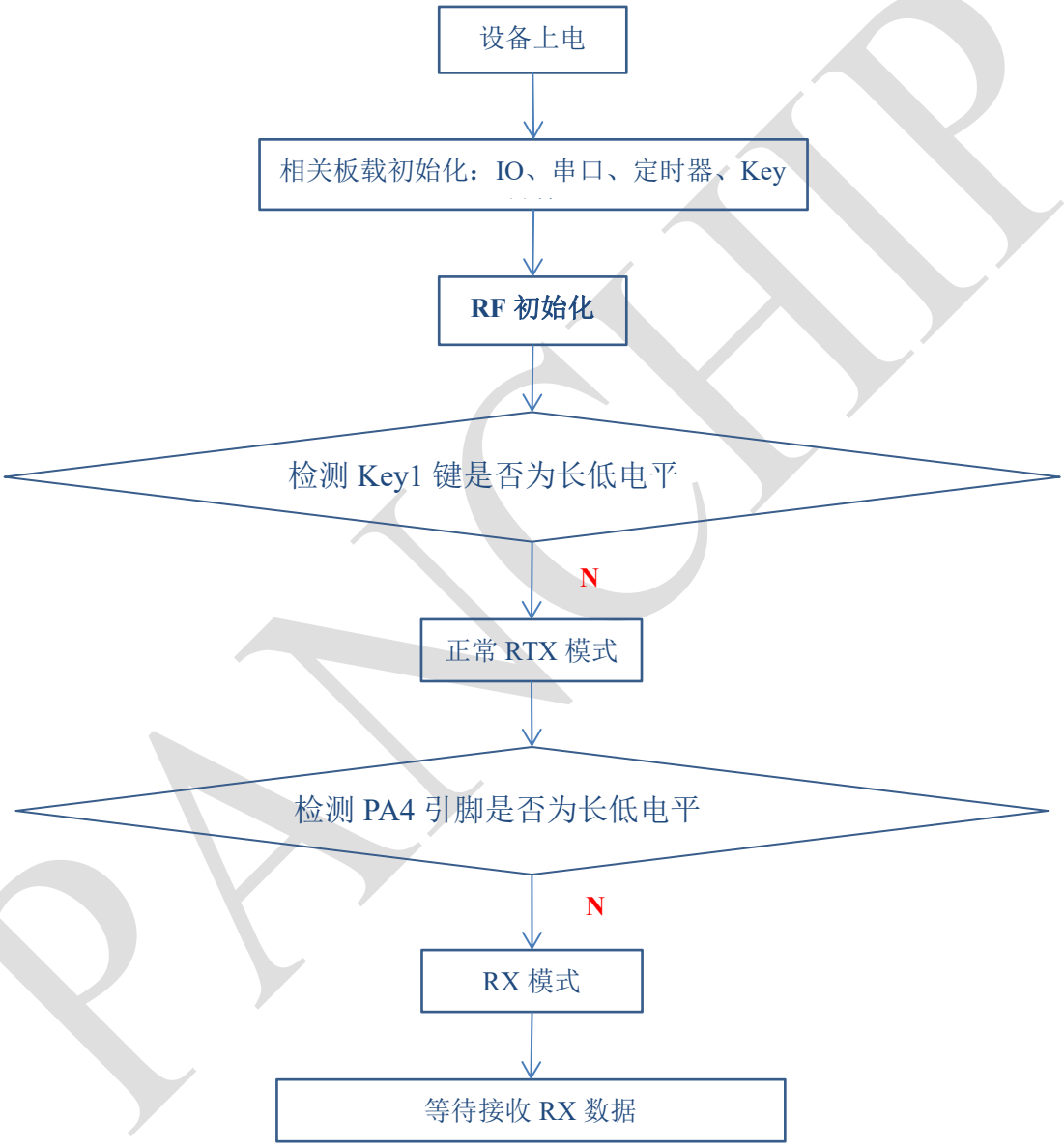
```
                APP_TXMODE_Swtich();//设置发包模式
                ……
                //Key scan  处理，主要用于频点的切换
            }
        }
}

RFAPI_StateMachine()
{
        ……
        RFAPI_TxState()
        {
            ……
            RFAPI_PacketData();
            RF_Tx_TransmintData(RF_Payload.ucPayload,TxPayloadLength);
        }
}
```

4.2.3 注意事项

4.3 Rx Demo

4.3.1 代码流程图



4.3.2 代码实现

```
    BIOS_Init
{
    ……
    RF_Init();
    Status_Init();    //默认为 RX 状态；
    ……
}
Int main()
{
    BIOS_Init();    //默认设置为 RX 模式，
    While(1)
    {
        APP_StateMachine();
        RFAPI_StateMachine();
    }
}

APP_StateMachine()
{
    ……
    APP_RX_Normal()
    {
        ……
        APP_RX_Event()
        {
            ……
```

```
                //Key scan  处理，主要用于频点的切换

            }
        }
}

RFAPI_StateMachine()
{
        ……
        RFAPI_RXState()
        {
            ……
            RF_DumpRxData()
}
```

4.3.3 注意事项

5 附件

5.1 发射功率表

寄存器值	发射功率 (dbm)	发射电流 (mA)
RF18DBM	18.12	78
RF17DBM	17.1	72.4
RF16DBM	16.7	64
RF15DBM	15.4	58
RF13DBM	13.2	48
RF12DBM	12.1	44.3
RF11DBM	11.08	42
RF10DBM	10.4	39.7
RF9DBM	9.4	37
RF8DBM	8.3	34.6
RF7DBM	7.2	32.8
RF6DBM	6.2	31
RF5DBM	5.4	29.8
RF4DBM	4.1	28
RF2DBM	2.3	26.2
RF1DBM	0.95	25.5
RF0DBM	0	24
RFN1DBM	-1	23.5
RFN6DBM	-6	21.4
RFN15DBM	-15	20.5

5.2 参考频点

在不同 BAND 下，频点参考值如下（目前尚未覆盖说明书中的全部频点）

```
#if(BAND == B315MHz)
/*BAND315MHz*/
    double const FREQ_CHANNEL_TABLE[]={315,317,319,321,323};
    double const CARRIER_FREQ_CHANNEL_TABLE[]={315,317,319,321,323};
```

```
#elif(BAND == B433MHz)
/*BAND433MHz*/
    double const FREQ_CHANNEL_TABLE[]={433,432,435,438,440};
    double const CARRIER_FREQ_CHANNEL_TABLE[]={433,432,435,438,440};

#elif(BAND == B868MHz)
/*BAND868MHz*/
    double const FREQ_CHANNEL_TABLE[]={868,868.5,869,869.5,870};
    double const CARRIER_FREQ_CHANNEL_TABLE[]={868,868.5,869,869.5,870};

#elif(BAND == B916MHz)
/*BAND915MHz*/
    double const FREQ_CHANNEL_TABLE[]={916,916.5,917,917.5,918};
    double const CARRIER_FREQ_CHANNEL_TABLE[]={916,916.5,917,917.5,918};
#endif
```