TK8610 无线终端芯片

编程示例说明

V1.0

修订记录

修订时间	修订版本	修订描述
2023-03-28	V1.0	初始版本

重要声明

联系方式

地址：上海嘉定皇庆路 333 号上海智能传感器产业园区 4 幢 5 层

邮编：201899

电话：021-61519850

邮箱：info@taolink-tech.com

网址：www.taolink-tech.com

目录

1 概述 1
1.1 编写目的 1
1.2 参考资料 1
2 工程目录说明 2
3 例程说明 3
3.1 突发模式 P2P 主机通信 3
3.1.1 例程目的 3
3.1.2 程序设计 3
3.1.3 工程名称 3
3.1.4 程序流程图 3
3.1.5 程序分析 3
3.1.6 运行结果 6
3.2 突发模式 P2P 从机通信 6
3.2.1 例程目的 6
3.2.2 程序设计 6
3.2.3 工程名称 6
3.2.4 程序流程图 7
3.2.5 程序分析 7
3.2.6 运行结果 9
3.3 带 LBT 功能的突发模式 P2P 通信主机 10
3.3.1 例程目的 10
3.3.2 程序设计 10
3.3.3 工程名称 10
3.3.4 程序流程图 10
3.3.5 程序分析 11
3.3.6 运行结果 14
3.4 带 LBT 功能的突发模式 P2P 通信从机 14
3.4.1 例程目的 14
3.4.2 程序设计 14
3.4.3 工程名称 14
3.4.4 程序流程图 14
3.4.5 程序分析 15
3.4.6 运行结果 15
3.5 时隙模式 P2P 通信主机 15
3.5.1 例程目的 15
3.5.2 程序设计 15
3.5.3 工程名称 15
3.5.4 程序流程图 16
3.5.5 程序分析 16
3.5.6 运行结果 18
3.6 时隙模式 P2P 通信从机 19
3.6.1 例程目的 19
3.6.2 程序设计 19

3.6.3 工程名称 19
3.6.4 程序流程图 20
3.6.5 程序分析 20
3.6.6 运行结果 22
3.7 P2P 通信无线唤醒主机 23
3.7.1 例程目的 23
3.7.2 程序设计 23
3.7.3 工程名称 23
3.7.4 程序流程图 24
3.7.5 程序分析 24
3.7.6 运行结果 27
3.8 P2P 通信无线唤醒从机 27
3.8.1 例程目的 27
3.8.2 程序设计 28
3.8.3 工程名称 28
3.8.4 程序流程图 28
3.8.5 程序分析 28
3.8.6 运行结果 32
3.9 突发模式 P2P 通信带传感器主机 32
3.9.1 例程目的 32
3.9.2 程序设计 32
3.9.3 工程名称 32
3.9.4 程序流程图 33
3.9.5 程序分析 33
3.9.6 运行结果 37
3.10 突发模式 P2P 通信带传感器从机 37
3.10.1 例程目的 37
3.10.2 程序设计 37
3.10.3 工程名称 37
3.10.4 程序流程图 37
3.10.5 程序分析 38
3.10.6 运行结果 38
3.11 P2P 通信 IO 唤醒 38
3.11.1 例程目的 38
3.11.2 程序设计 38
3.11.3 工程名称 38
3.11.4 程序流程图 38
3.11.5 程序分析 39
3.11.6 运行结果 42

图形目录

图 2-1 工程目录介绍 2
图 3-1 突发模式主机通信程序流程图 3
图 3-2 突发模式从机通信程序流程图 7

图 3-3 串口调试助手打印信息 10
图 3-4 带 LBT 功能的突发模式通信程序流程图 11
图 3-5 时隙模式 P2P 通信主机程序流程图 16
图 3-6 P2P 通信无线唤醒从机程序流程图 28
图 3-7 突发模式 P2P 通信带传感器主机程序流程图 33

表目录

表 2-1 工程目录介绍 2

1 概述

1.1 编写目的

介绍 TK8610 芯片编程例程实现方法，让用户快速理解掌握 TK8610 使用方法。

1.2 参考资料

《TK8610 无线终端芯片编程指南（SDK）》。

2 工程目录说明

以突发模式 P2P 工程为例来讲解，其他工程类似：

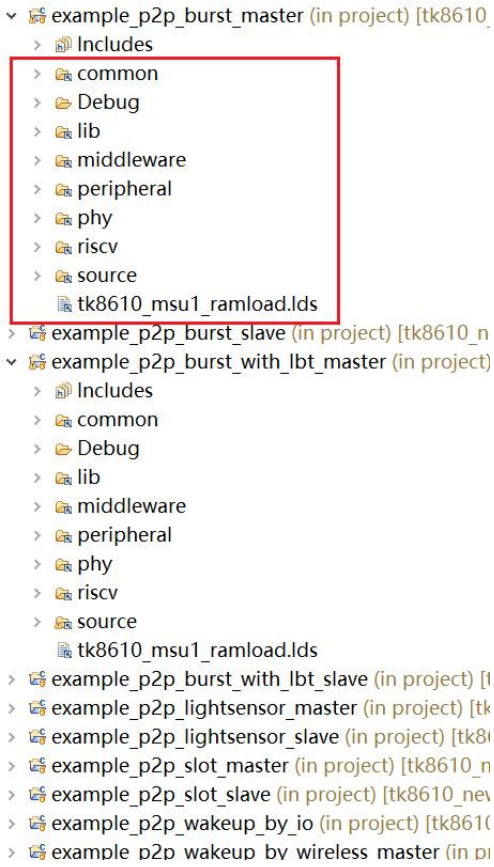


图 2-1 工程目录介绍

目录名	内容
Common	通用功能函数头文件
Debug	生成的一些中间文件以及编译生成的 HEX 文件
lib	SDK 库文件
middleware	中间件 API 的头文件
peripheral	外设的头文件
phy	物理层 API 的头文件
riscv	RISC-V 内核的启动代码
source	示例源代码
Tk8610_msu1_ramload.lds	链接文件

表 2-1 工程目录介绍

3 例程说明

3.1 突发模式 P2P 主机通信

3.1.1 例程目的

演示突发模式 P2P 通信，主机端发送数据。

3.1.2 程序设计

本示例设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），主循环轮询射频状态，并启动 1 秒周期上报数据，每次发送 10 字节数据，以 68 开头，后面四字节是包序号，并通过 UART1 打印发送的数据。

3.1.3 工程名称

example_ p2p_burst_ master

3.1.4 程序流程图

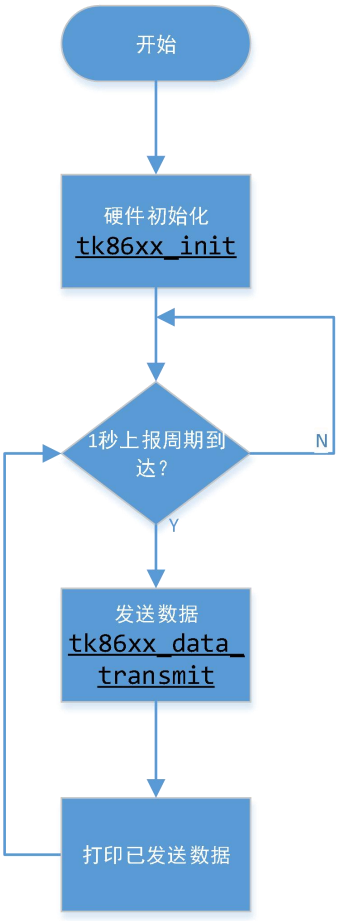


图 3-1 突发模式主机通信程序流程图

3.1.5 程序分析

- main() 函数

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.  /*调试串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p burst master ******\r\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
14. while (1)
15. {
16.  /* RF 系统轮询,用到RF 硬件资源，必须在主循环调用此接口，且主循环不能
有阻塞延时 */
17.  tk86xx_poll();
18.  /* 需要轮询的函数 */
19.  usr_proc_poll();
20. }
21.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，然后 1 秒周期上报数据。

main 函数中第一步对系统时钟初始化；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化定时器 1；第五步打印工程信息。

主循环驱动 RF 系统工作在突发模式，在此模式下，RF 始终处于空闲状态（即等待接收数据）。当用户有数据要发送的时候，会自动切换到发送状态，且在发送完成后，又会自动切换回空闲状态（等待接收数据）。用户层级轮询函数实现数据上报。

- **rf_init() 射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.
8.void rf_init(void)
9.{
```

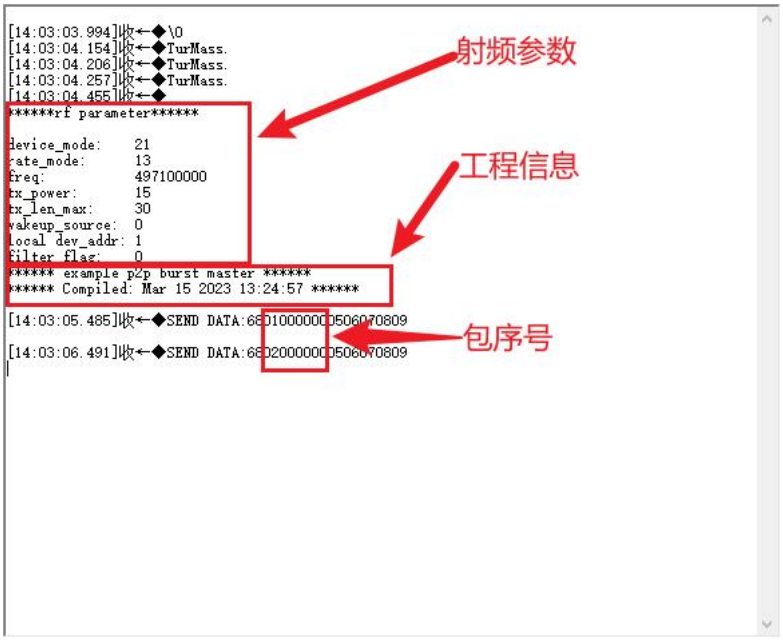
```
    /*参数配置*/
    static mid_config_u mid_config = {
        /*工作模式设置*/
  .p2p_config.work_mode                = P2P_BURST_NORMAL,    //突发模式
        /*共用参数设置*/
  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,         //速率模式
  .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,        //BCN 模式
  .p2p_config.common_config.bcn_id    = 1,                    //BCN 的 ID
  .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,         //BCN 频率
  .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,         //发送频率
  .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,         //接收频率
  .p2p_config.common_config.tx_power   = SYS_RF_PWR,          //发送功率
        /*突发模式参数设置*/
  .p2p_config.burst_config.dev_addr    = 1,                   //本地地址 启用地址过滤时使用
  .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      //不启用本地地址过滤功能   (本地地址过滤,与本地地址配合使用，仅用于工作模式 21 )
    };

    /*射频参数初始化*/
   tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);
```

```
38.#endif
39.}
```

此函数实现射频参数初始化，以及接收回调函数注册。

3.1.6 运行结果

编译并下载示例代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如下：



串口调试助手打印射频参数，工程信息和发送的数据包。

3.2 突发模式 P2P 从机通信

3.2.1 例程目的

演示突发模式从机端接收数据，并打印接收成功率。

3.2.2 程序设计

本实验设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），主循环轮询是否接收到数据，一旦接收到数据，通过 UART1 打印接收到的数据，并根据包序号计算接收成功率。

3.2.3 工程名称

example_p2p _burst _slave

3.2.4 程序流程图

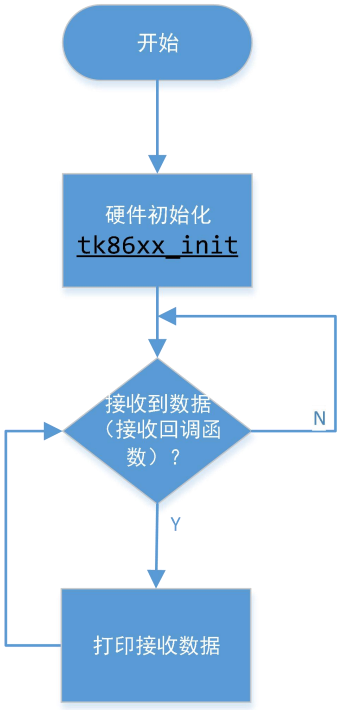


图 3-2 突发模式从机通信程序流程图

3.2.5 程序分析

- main()函数

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.  /*调试串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p burst slave ******\r\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
14. while (1)
15. {
16.  /* RF 系统轮询,用到RF 硬件资源，必须在主循环调用此接口，且主循环不能
有阻塞延时 */
17.  tk86xx_poll();
18.  /* 需要轮询的函数 */
19.  usr_proc_poll();
```

```
20. }
21.}
```

在 main 函数中完成硬件初始化，接收回调函数被注册，主循环中实现 RF 系统轮询，如果接收到数据，则接收回调函数被调用，打印接收到的数据以及接收成功率。

main 函数中第一步对系统时钟初始化；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步定时器 1 初始化；第五步打印工程信息。

主循环驱动 RF 系统工作，在接收到数据的时候，打印接收信号质量、接收数据以及接收成功率，请见下图 3-3。

- **rf_init()射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE          30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.
8.void rf_init(void)
9.{
10.    /*参数配置*/
11.    static mid_config_u mid_config = {
12.        /*工作模式设置*/
13.  .p2p_config.work_mode                = P2P_BURST_NORMAL,    //突发模式
14.        /*共用参数设置*/
15.  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,         //速率模式
16.  .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,       //BCN 模式
17.  .p2p_config.common_config.bcn_id    = 1,                  // BCN 的 ID
18.  .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,        //BCN 频率
19.  .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,        //发送频率
20.  .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,        //接收频率
21.  .p2p_config.common_config.tx_power   = SYS_RF_PWR,         //发送功率
22.        /*突发模式参数设置*/
```

```
    .p2p_config.burst_config.dev_addr     = 1,                    //本地地址 启用地址过滤时使用
    .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      //不启用本地地址过滤功能    （本地地址过滤,与本地地址配合使用，仅用于工作模式21）
     };

    /*射频参数初始化*/
     tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

3.2.6 运行结果

编译并下载代码到开发板中，打开串口调试助手，开发板上电，另外一块开发板运行 example_burst_p2p_master 例程。在串口调试助手显示如下：

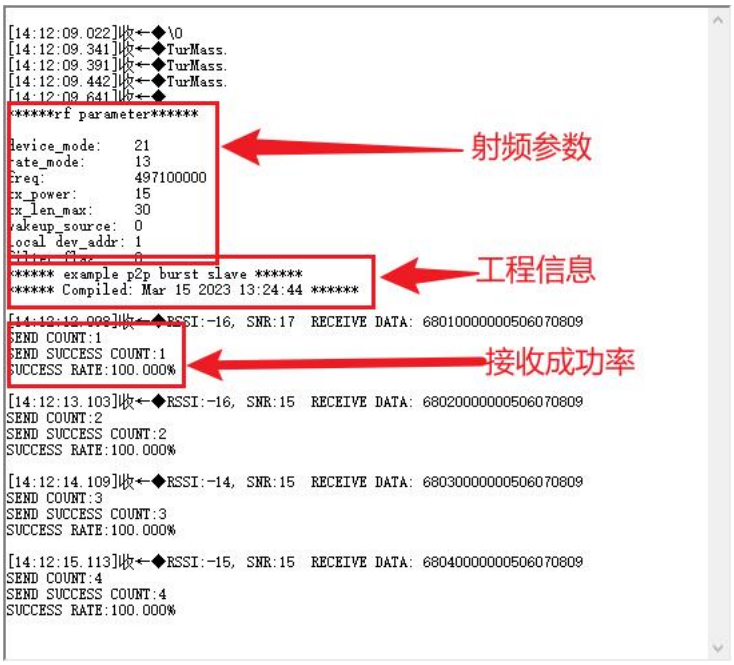


图 3-3 串口调试助手打印信息

串口调试助手打印射频参数，工程信息和接收到的数据包。

3.3 带 LBT 功能的突发模式 P2P 通信主机

3.3.1 例程目的

演示 LBT 功能，终端通过 LBT 功能提高通信成功率。

3.3.2 程序设计

本示例设计射频工作模式为突发模式（P2P_BURST_NORMAL），使用定时器 1 实现时间管理，主机 5 秒周期上报数据，上报数据前进行信道侦听，信道空闲或者侦听超时时，上报数据。

3.3.3 工程名称

example_p2p_burst_with_lbt_master

3.3.4 程序流程图

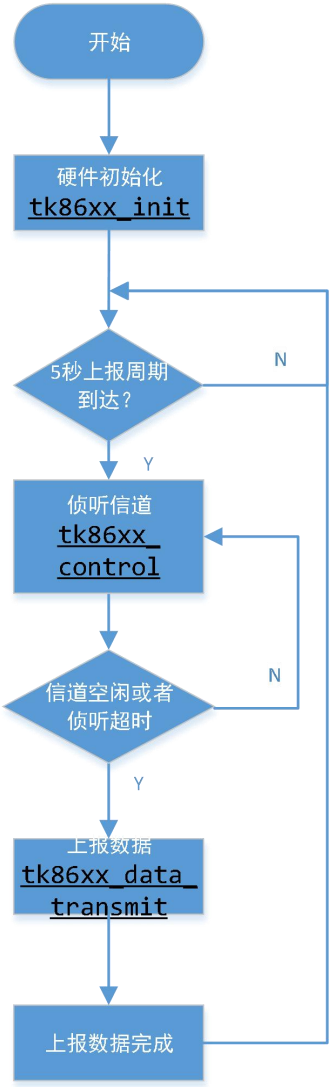


图 3-4 带 LBT 功能的突发模式通信程序流程图

3.3.5 程序分析

- main()函数

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.    /*串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
```

```
    tk_printf("****** example p2p burst with lbt master ******\r\n");
    tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME__);
    while (1)
    {
    /* RF 系统轮询,用到RF 硬件资源，必须在主循环调用此接口，且主循环不能有阻塞延时 */
     tk86xx _poll();
    /* 需要轮询的函数 */
     usr_proc_poll();
    }
}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，以及用户层级需要轮询的函数，本例程里面是实现 5 秒周期上报数据，上报数据前以 100 毫秒周期侦听信道，直至信道空闲或者侦听超时，上报数据。

main 函数中第一步对系统时钟初始化，目前系统时钟只支持如此配置，用户直接调用即可；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步定时器初始化；第五步打印工程信息。

主循环驱动 RF 系统工作，以及其他需要轮询的功能函数。

- **rf_init()射频初始化函数**

```
#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
#define SYS_RF_FREQ      (497100000)
#define SYS_RF_PWR      15
#define SYS_RF_RATE      P2P_RATE_13
#define SYS_RF_MAXBYTE          30
#define SYS_RF_LBT_THRESHOLD    -90
#define SYS_DEST_DEV_ADDR       (0xffffffff)

void rf_init(void)
{
    /*参数配置*/
    static mid_config_u mid_config = {
        /*工作模式设置*/
     .p2p_config.work_mode                = P2P_BURST_NORMAL, // 突发模式
        /*共用参数设置*/
     .p2p_config.common_config.rate_mode  = SYS_RF_RATE, // 速率模式
```

```
    .p2p_config.common_config.bcn_mode    = P2P_BCN_AUTO,        // BCN 模式
    .p2p_config.common_config.bcn_id    = 1,                     // BCN 的 ID
    .p2p_config.common_config.bcn_freq    = SYS_RF_FREQ,         // BCN 频率
    .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,          // 发送频率
    .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,          // 接收频率
    .p2p_config.common_config.tx_power    = SYS_RF_PWR,          // 发送功率
        /*突发模式参数设置*/
    .p2p_config.burst_config.dev_addr    = 1,                    // 本地地址 启用地址过滤时使用
    .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,       // 不启用本地地址过滤功能    （本地地址过滤,与本地地址配合使用，仅用于工作模式 21）
    };
    /*射频参数初始化*/
    tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);
    tk_printf("lbt threshold:\t%d\n", SYS_RF_LBT_THRESHOLD);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

- **rf_lbt()侦听信道函数**

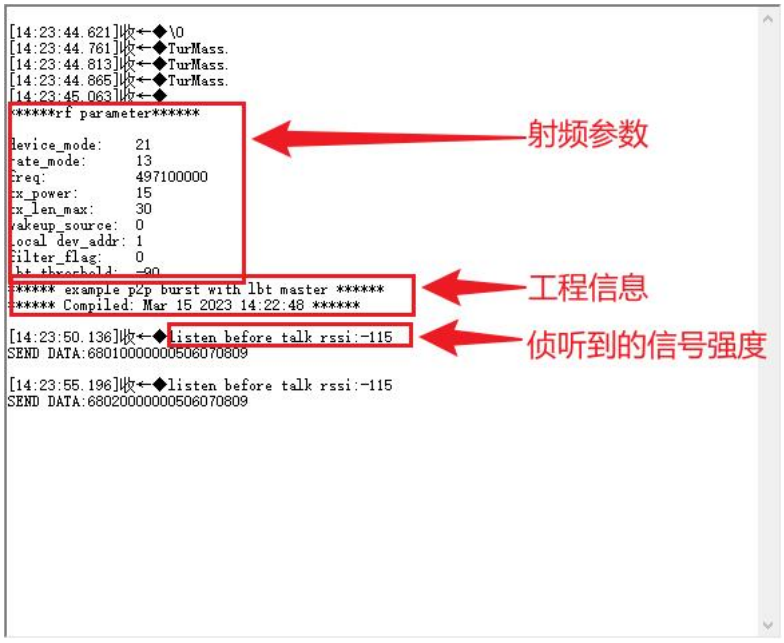
```
int16_t rf_lbt(uint32_t freq)
```

```
2.{
3.    int16_t rssi;
4.   tk86xx_control(P2P_LISTEN, &freq, &rssi);
5.
6.    return rssi;
7.}
```

此函数实现信道侦听，返回当前信道空中信号强度。

3.3.6 运行结果

编译并下载示例代码到开发板中，打开串口调试助手，串口助手显示如下：



3.4 带 LBT 功能的突发模式 P2P 通信从机

3.4.1 例程目的

本例程只实现数据接收，和 example_p2p _burst _slave 设计一致，这里不再赘述。

3.4.2 程序设计

略。

3.4.3 工程名称

example_p2p_burst_with_lbt_slave

3.4.4 程序流程图

略。

3.4.5 程序分析

略。

3.4.6 运行结果

略。

3.5 时隙模式 P2P 通信主机

3.5.1 例程目的

演示时隙模式下 P2P 通信。

3.5.2 程序设计

本实验设计射频工作模式为时隙模式主机（P2P_SLOT_MASTER），主循环轮询射频状态，主机 3 秒周期上报数据，并打印发送的数据，接收到应答后，打印上报成功率。

3.5.3 工程名称

example_p2p_slot_master

3.5.4 程序流程图

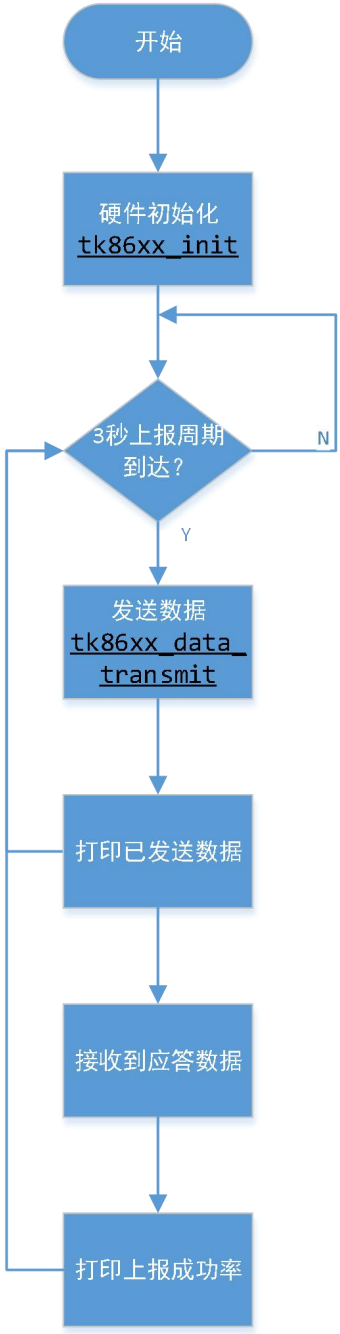


图 3-5 时隙模式 P2P 通信主机程序流程图

3.5.5 程序分析

- main()函数

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.    /*串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
```

```
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p slot master ******\r\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME__);
14. while (1)
15. {
16.  /* RF 系统轮询,用到RF 硬件资源，必须在主循环调用此接口，且主循环不能有阻塞延时 */
17.  tk86xx_poll();
18.  /* 需要轮询的函数 */
19.  usr_proc_poll();
20. }
21.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，以及用户层级的函数轮询，本示例中用户轮询函数实现 3 秒周期上报数据。

main 函数中第一步对系统时钟初始化，用户直接调用即可；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化定时器 1，第五步打印工程信息。

主循环驱动 RF 系统工作，时隙模式时，射频发送完成后，发送时隙状态自动变为空闲模式。用户层级轮询函数实现数据发送和接收。

- **rf_init()射频初始化函数**

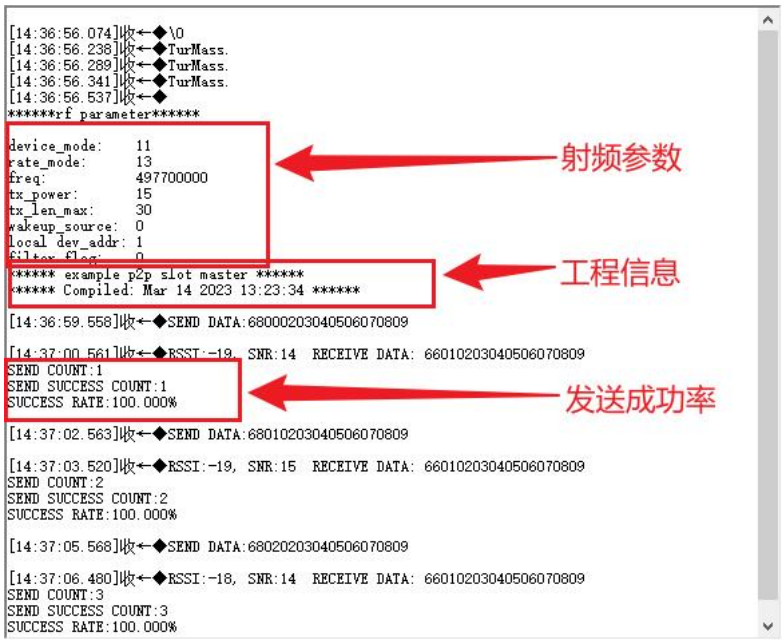
```
1.#define SYS_RF_WORK_MODE    P2P_SLOT_MASTER
2.#define SYS_RF_FREQ      (497700000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.
8.void rf_init(void)
9.{
10.    /*参数配置*/
11.    static mid_config_u mid_config = {
12.        /*工作模式设置*/
13.  .p2p_config.work_mode                = P2P_SLOT_MASTER,    //
14.        /*共用参数设置*/
15.  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,        //速率模式
```

```
    .p2p_config.common_config.bcn_mode    = P2P_BCN_AUTO,        // BCN 模式
    .p2p_config.common_config.bcn_id    = 1,                     // BCN 的 ID
    .p2p_config.common_config.bcn_freq    = SYS_RF_FREQ,         // BCN 频率
    .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,          // 发送频率
    .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,          // 接收频率
    .p2p_config.common_config.tx_power    = SYS_RF_PWR,          // 发送功率
        /*时隙模式参数设置*/
    .p2p_config.slot_config.tx_len_max     = SYS_RF_MAXBYTE,    // 配置最大发送字节数
    };
    /*射频参数初始化*/
    tk86xx_init(P2P_MODE,&mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);
    tk_printf("tx_len_max:\t%d\n", mid_config.slot_config.tx_len_max);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

3.5.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如图：



串口调试助手打印工程信息和发送的数据包。

3.6 时隙模式 P2P 通信从机

3.6.1 例程目的

演示时隙模式下 P2P 通信。

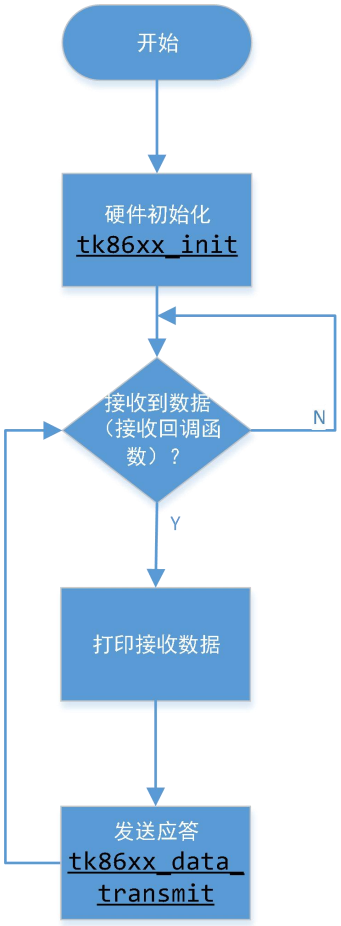
3.6.2 程序设计

本实验设计射频工作模式为时隙模式从机（P2P_SLOT_SLAVE），主循环轮询射频状态，接收到数据后，打印接收到的数据，然后给出应答数据。

3.6.3 工程名称

example_p2p_slot_slave

3.6.4 程序流程图



3.6.5 程序分析

- **main()函数**

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.  /*串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p slot slave ******\r\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME__);
14. while (1)
15. {
```

```
    /* RF 系统轮询,用到RF 硬件资源，必须在主循环调用此接口，且主循环不能有阻塞延时 */
    tk86xx_poll();
    /* 需要轮询的函数 */
    usr_proc_poll();
}
}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，以及用户层级的函数轮询，本示例中用户轮询函数实现对主机进行应答。

main 函数中第一步对系统时钟初始化，用户直接调用即可；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化定时器 1，第五步打印工程信息。

主循环驱动 RF 系统工作，时隙模式时，射频发送完成后，发送时隙状态自动变为空闲状态。

- **rf_init()射频初始化函数**

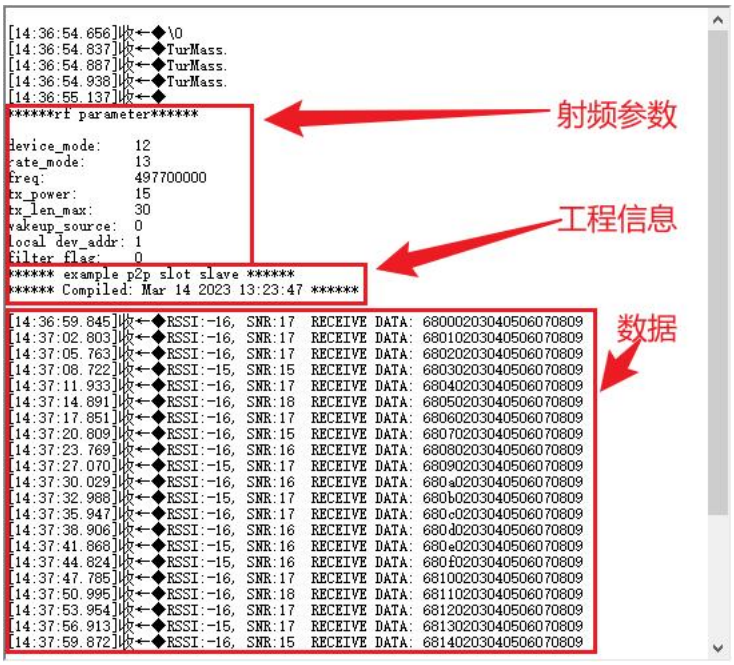
```
#define SYS_RF_WORK_MODE    P2P_SLOT_SLAVE
#define SYS_RF_FREQ      (497700000)
#define SYS_RF_PWR      15
#define SYS_RF_RATE      P2P_RATE_13
#define SYS_RF_MAXBYTE         30
#define SYS_DEST_DEV_ADDR      (0xffffffff)

void rf_init(void)
{
    /*参数配置*/
    static mid_config_u mid_config = {
        /*工作模式设置*/
    .p2p_config.work_mode               = P2P_SLOT_SLAVE,     //时隙模式从机
        /*共用参数设置*/
    .p2p_config.common_config.rate_mode  = SYS_RF_RATE,        //速率模式
    .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,       //BCN 模式
    .p2p_config.common_config.bcn_id    = 1,                   //BCN 的 ID
    .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,        //BCN 频率
    .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,        //发送频率
    .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,        //接收频率
```

```
    .p2p_config.common_config.tx_power    = SYS_RF_MAXBYTE,
// 发送功率
        /*时隙模式参数设置*/
    .p2p_config.slot_config.tx_len_max     = 1,                    //
配置最大发送字节数
     };
    /*射频参数初始化*/
     tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_
mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq)
;
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_pow
er);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.d
ev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filt
er_flag);
    tk_printf("tx_len_max:\t%d\n", mid_config.slot_config.tx_len
_max);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

3.6.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如图：



串口调试助手打印工程信息和发送的数据包。

3.7 P2P 通信无线唤醒主机

3.7.1 例程目的

演示主机通过无线唤醒从机并向从机发送数据。

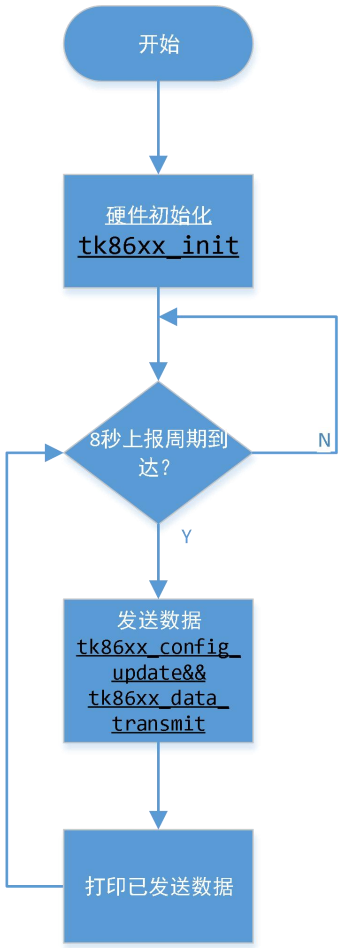
3.7.2 程序设计

本实验设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），主循环轮询射频状态，8 秒周期唤醒从机并发送数据，并通过 UART1 打印发送的数据。

3.7.3 工程名称

example _p2p_wakeup_by_wireless _master

3.7.4 程序流程图



3.7.5 程序分析

- **main()函数**

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.  /*调试串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p wakeup by wireless master ******\
r\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
```

```
14. while (1)
15. {
16.  /* RF 系统轮询，用到RF 硬件资源，必须在主循环调用此接口，且主循环不能有阻塞延时 */
17.  tk86xx_poll();
18.  /* 需要轮询的函数 */
19.  usr_proc_poll();
20. }
21.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，8 秒周期发送唤醒信号及数据。

main 函数中第一步对系统时钟初始化，目前系统时钟支持如此配置，用户直接调用即可；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化定时器 1；第五步打印工程信息。

主循环驱动 RF 系统工作，突发模式时，射频发送完成后，状态变为空闲模式，同时自动切换为接收模式，除非关闭射频。用户层级轮询函数实现数据发送。

- **rf_init()射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.#define SYS_RF_WAKEUP_FREQ    470100000
8.#define SYS_RF_WAKEUP_ID    100
9.#define SYS_RF_WAKEUP_PERIOD   1000
10.#define SYS_RF_TIMER_WAKEUP_PERIOD   10000
11.
12.void rf_init(void)
13.{
14.    /*参数配置*/
15.    static mid_config_u mid_config = {
16.        /*工作模式设置*/
17.  .p2p_config.work_mode                = P2P_BURST_NORMAL,   //突发模式
18.        /*共用参数设置*/
19.  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,        //速率模式
20.  .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,       //BCN 模式
```

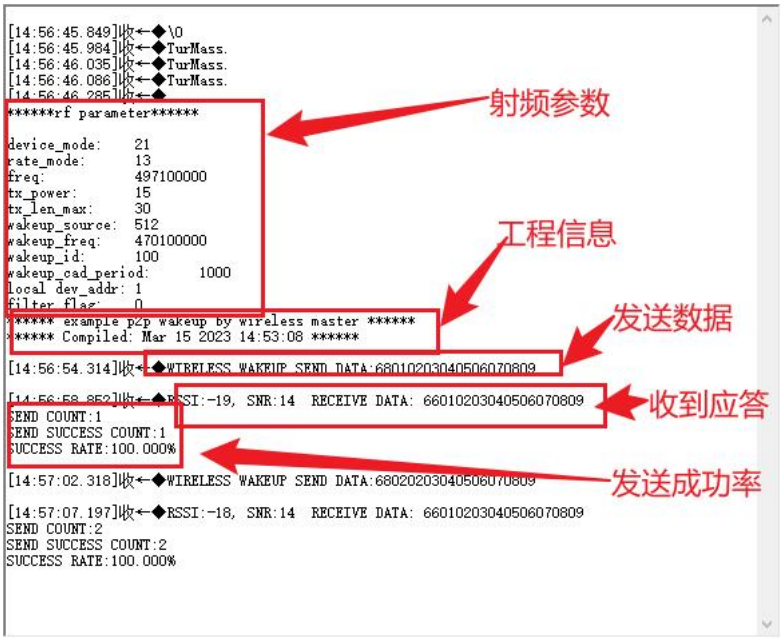
```
    .p2p_config.common_config.bcn_id    = 1,                    // BCN 的 ID
    .p2p_config.common_config.bcn_freq    = SYS_RF_FREQ,        // BCN 频率
    .p2p_config.common_config.tx_freq     = SYS_RF_FREQ,        // 发送频率
    .p2p_config.common_config.rx_freq     = SYS_RF_FREQ,        // 接收频率
    .p2p_config.common_config.tx_power    = SYS_RF_PWR,         // 发送功率
        /*突发模式参数设置*/
    .p2p_config.burst_config.dev_addr     = 1,                  // 本地地址 启用地址过滤时使用
    .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      // 不启用本地地址过滤功能  （本地地址过滤，与本地地址配合使用，仅用于工作模式21）
        /*唤醒参数参数设置*/
    .p2p_config.wakeup_config.wakeup_source      = P2P_WAKEUP_TIMER,
      // 配置唤醒源为定时器
    .p2p_config.wakeup_config.wakeup_id          = SYS_RF_WAKEUP_ID,
      // 唤醒 ID 无线唤醒时使用
    .p2p_config.wakeup_config.wakeup_freq        = SYS_RF_WAKEUP_FREQ,
      // 唤醒频率 无线唤醒时使用
    .p2p_config.wakeup_config.wakeup_timing      = SYS_RF_TIMER_WAKEUP_PERIOD,// 定时唤醒周期 定时唤醒时使用
    .p2p_config.wakeup_config.wakeup_level       = P2P_EXTI_LOW,
      // IO 唤醒电平   IO 唤醒时使用
    .p2p_config.wakeup_config.wakeup_cad_period = 1000,
      // 无线唤醒周期  无线唤醒时使用
    };
   /*射频参数初始化*/
    tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
```

```
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);

    tk_printf("wakeup_source:\t%d\n", mid_config.wakeup_config.wakeup_source);
    tk_printf("wakeup_timing:\t%d\n", mid_config.wakeup_config.wakeup_timing);
    tk_printf("wakeup_freq:\t%d\n", mid_config.wakeup_config.wakeup_freq);
    tk_printf("wakeup_id:\t%d\n", mid_config.wakeup_config.wakeup_id);
    tk_printf("wakeup_cad_period:\t%d\n", mid_config.wakeup_config.wakeup_cad_period);
#endif
}
```

此函数实现射频参数初始化，以及接收函数注册。

3.7.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如图。



串口调试助手打印工程信息和发送的数据包。

3.8 P2P 通信无线唤醒从机

3.8.1 例程目的

演示无线唤醒模式从机接收唤醒信号唤醒后回复数据，然后 3 秒后休眠。

3.8.2 程序设计

本实验设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），主循环轮询是否接收到数据，一旦接收到数据，通过 UART1 打印接收到的数据，回复数据以后继续休眠。

3.8.3 工程名称

example_p2p _wakeup_ by_wireless _slave

3.8.4 程序流程图

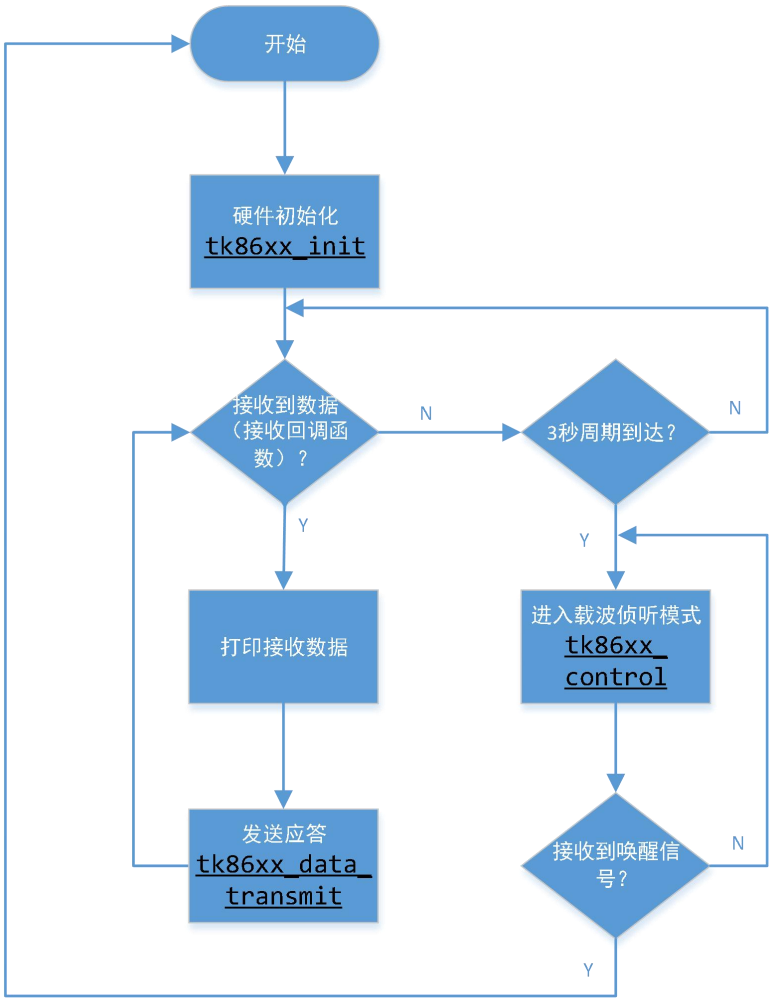


图 3-6 P2P 通信无线唤醒从机程序流程图

3.8.5 程序分析

- main()函数

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.  /*调试串口初始化*/
6. uart1_drv_init(BAUD_RATE_115200);
```

```
7. /*射频初始化*/
8. rf_init();
9. /*定时器初始化*/
10. timer1_drv_init();
11. /*例程信息打印*/
12. tk_printf("****** example p2p wakeup by wireless slave ******\r
\n");
13. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
14. while (1)
15. {
16.  /* RF 系统轮询，用到RF 硬件资源，必须在主循环调用此接口，且主循环不能
有阻塞延时 */
17.  tk86xx_poll();
18.  /* 需要轮询的函数 */
19.  usr_proc_poll();
20. }
21.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，如果接收到数据，则打印数据，3 秒后继续休眠。

main 函数中第一步对系统时钟初始化，用户直接调用即可；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化定时器 1；第五步打印工程信息。

主循环驱动 RF 系统工作，突发模式时，如果没有数据发送一直处于接收状态，除非关闭射频。第 19 行，用户层级轮询函数实现应答并在 3 秒后进入载波侦听状态。

- **rf_init()射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.#define SYS_RF_WAKEUP_FREQ    470100000
8.#define SYS_RF_WAKEUP_ID    100
9.#define SYS_RF_WAKEUP_PERIOD   1000
10.#define SYS_RF_TIMER_WAKEUP_PERIOD   10000
11.
12.void rf_init(void)
13.{
14.    /*参数配置*/
15.    static mid_config_u mid_config = {
16.        /*工作模式设置*/
```

```
    .p2p_config.work_mode                = P2P_BURST_NORMAL,    //突发模式
        /*共用参数设置*/
    .p2p_config.common_config.rate_mode  = SYS_RF_RATE,         //速率模式
    .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,        //BCN 模式
    .p2p_config.common_config.bcn_id    = 1,                    //BCN 的 ID
    .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,         //BCN 频率
    .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,         //发送频率
    .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,         //接收频率
    .p2p_config.common_config.tx_power   = SYS_RF_PWR,          //发送功率
        /*突发模式参数设置*/
    .p2p_config.burst_config.dev_addr    = 1,                   //本地地址 启用地址过滤时使用
    .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      //不启用本地地址过滤功能    （本地地址过滤,与本地地址配合使用，仅用于工作模式21）
        /*唤醒参数参数设置*/
    .p2p_config.wakeup_config.wakeup_source     = P2P_WAKEUP_TIMER,
      // 配置唤醒源为定时器
    .p2p_config.wakeup_config.wakeup_id         = SYS_RF_WAKEUP_ID,
      // 唤醒 ID 无线唤醒时使用
    .p2p_config.wakeup_config.wakeup_freq       = SYS_RF_WAKEUP_FREQ,
      // 唤醒频率 无线唤醒时使用
    .p2p_config.wakeup_config.wakeup_timing     = SYS_RF_TIMER_WAKEUP_PERIOD,// 定时唤醒周期 定时唤醒时使用
    .p2p_config.wakeup_config.wakeup_level      = P2P_EXTI_LOW,
      // IO 唤醒电平   IO 唤醒时使用
    .p2p_config.wakeup_config.wakeup_cad_period = 1000,
      // 无线唤醒周期  无线唤醒时使用
    };
   /*射频参数初始化*/
    tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
```

```
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);

    tk_printf("wakeup_source:\t%d\n", mid_config.wakeup_config.wakeup_source);
    tk_printf("wakeup_timing:\t%d\n", mid_config.wakeup_config.wakeup_timing);
    tk_printf("wakeup_freq:\t%d\n", mid_config.wakeup_config.wakeup_freq);
    tk_printf("wakeup_id:\t%d\n", mid_config.wakeup_config.wakeup_id);
    tk_printf("wakeup_cad_period:\t%d\n", mid_config.wakeup_config.wakeup_cad_period);
#endif
}
```

此函数实现射频参数初始化，接收回调函数注册，以及无线唤醒参数设置。

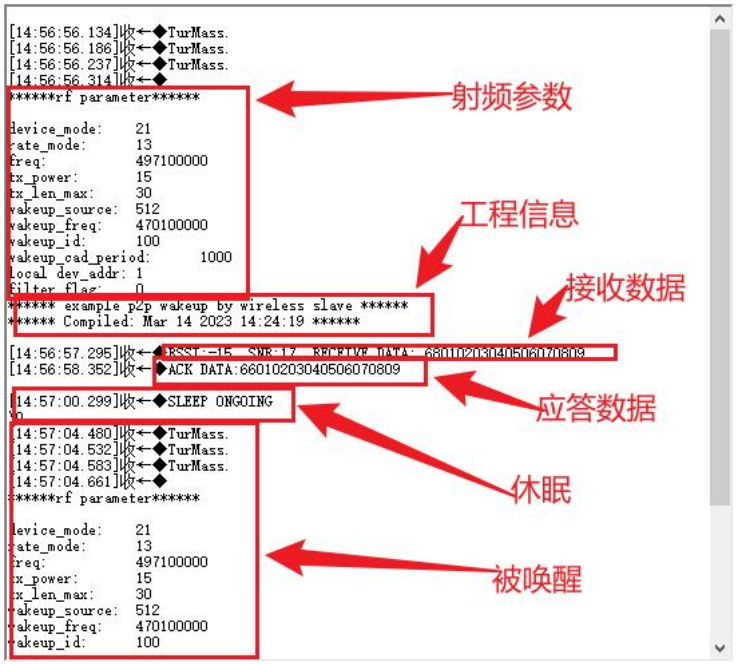
- **sleep_config()休眠配置函数**

```
void sleep_config(void)
{
    mid_config_u mid_config;

    /* 通过无线唤醒 */
    mid_config.p2p_config.wakeup_config.wakeup_source = P2P_WAKEUP_WIRELESS;
    mid_config.p2p_config.wakeup_config.wakeup_id = SYS_RF_WAKEUP_ID;
    mid_config.p2p_config.wakeup_config.wakeup_freq = SYS_RF_WAKEUP_FREQ;
    mid_config.p2p_config.wakeup_config.wakeup_cad_period = SYS_RF_WAKEUP_PERIOD;
    /* 配置休眠参数 */
     tk86xx_config_update(&mid_config, P2P_WAKEUP_SRC | P2P_WAKEUP_ID | P2P_WAKEUP_CAD_PERIOD | P2P_WAKEUP_FREQ);
    /* 进入休眠 */
    tk86xx_control(P2P_SLEEP, NULL, NULL);
}
```

第 6 行配置唤醒源为无线唤醒，第 7 行配置本机唤醒 ID，第 8 行配置无线侦听频率，第 9 行配置无线侦听周期为 1 秒，第 11 行调用 tk86xx_config_update 函数更新唤醒参数，第 13 行执行休眠函数 tk86xx_control()。

3.8.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，同时主机串上电，串口调试助手显示如图：



串口调试助手打印射频参数，工程信息、接收到的数据以及发送的应答数据。

3.9 突发模式 P2P 通信带传感器主机

3.9.1 例程目的

演示突发模式主机端读取传感器，并发送数据。

3.9.2 程序设计

本示例设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），10 秒周期定时唤醒，然后上报数据，并通过 UART1 打印发送的数据，上报完成 1 秒后休眠。

3.9.3 工程名称

example_ p2p_lightsensor_ master

3.9.4 程序流程图

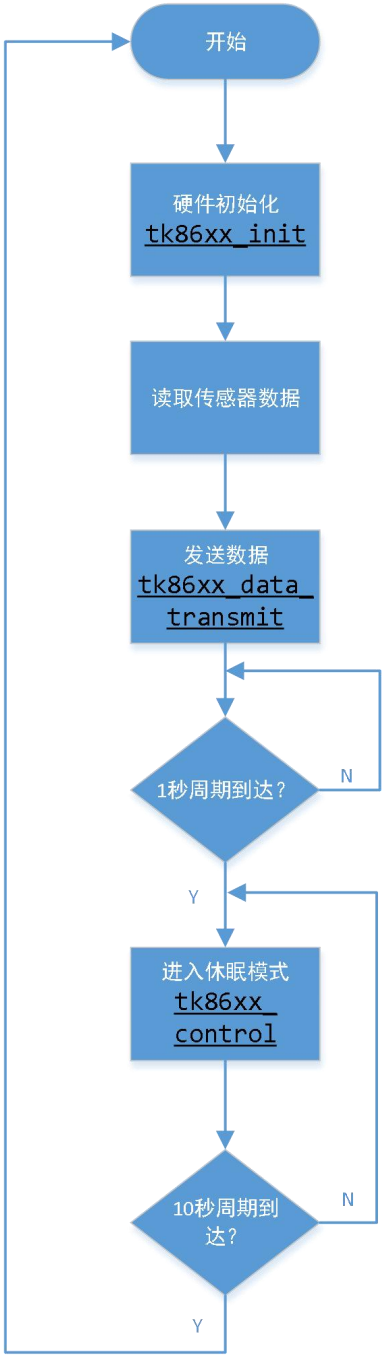


图 3-7 突发模式 P2P 通信带传感器主机程序流程图

3.9.5 程序分析

- **main()函数**

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5.    /*调试串口初始化*/
```

```
    6. uart1_drv_init(BAUD_RATE_115200);
    7. /*射频初始化*/
    8. rf_init();
    9. /*光照传感器 BH1750 初始化*/
    10. bh1750_init();
    11. /*定时器初始化*/
    12. timer1_drv_init();
    13. /*例程信息打印*/
    14. tk_printf("****** example p2p lightseneor master ******\r\n");
    15. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
    16. while (1)
    17. {
    18.  /* RF 系统轮询，用到 RF 硬件资源，必须在主循环调用此接口，且主循环不能
有阻塞延时 */
    19.  tk86xx_poll();
    20.  /* 需要轮询的函数 */
    21.  usr_proc_poll();
    22. }
    23.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，和传感器数据上报，然后 1 秒周期到达进入休眠。

main 函数中第一步对系统时钟初始化；第二步初始化串口 1，用来打印信息；第三步初始化射频参数；第四步初始化传感器；第五步初始化定时器 1；第六步打印工程信息。

主循环驱动 RF 系统工作在突发模式，在此模式下，RF 始终处于空闲状态（即等待接收数据）。当用户有数据要发送的时候，会自动切换到发送状态，且在发送完成后，又会自动切换回空闲状态（等待接收数据）。第 21 行用户层级轮询函数实现传感器数据上报，以及 1 秒后自动休眠。

- **rf_init()射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
6.#define SYS_DEST_DEV_ADDR      (0xffffffff)
7.#define SYS_RF_WAKEUP_FREQ    470100000
8.#define SYS_RF_WAKEUP_ID    100
9.#define SYS_RF_WAKEUP_PERIOD   1000
10.#define SYS_RF_TIMER_WAKEUP_PERIOD   10000
11.
12.void rf_init(void)
```

```
13.{
14.    /*参数配置*/
15.    static mid_config_u mid_config = {
16.        /*工作模式设置*/
17.  .p2p_config.work_mode                = P2P_BURST_NORMAL,    //
突发模式
18.        /*共用参数设置*/
19.  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,         //
速率模式
20.  .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,        //
BCN 模式
21.  .p2p_config.common_config.bcn_id    = 1,                   // BC
N 的 ID
22.  .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,         //
BCN 频率
23.  .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,         //
发送频率
24.  .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,         //
接收频率
25.  .p2p_config.common_config.tx_power   = SYS_RF_PWR,          //
发送功率
26.        /*突发模式参数设置*/
27.  .p2p_config.burst_config.dev_addr    = 1,                   //
本地地址 启用地址过滤时使用
28.  .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      //
不启用本地地址过滤功能   （本地地址过滤，与本地地址配合使用，仅用于工作模式21）
29.        /*唤醒参数参数设置*/
30.  .p2p_config.wakeup_config.wakeup_source     = P2P_WAKEUP_EXTI_
B0,         // 配置唤醒源为外部IO B0
31.  .p2p_config.wakeup_config.wakeup_id         = SYS_RF_WAKEUP_ID,
          // 唤醒 ID 无线唤醒时使用
32.  .p2p_config.wakeup_config.wakeup_freq       = SYS_RF_WAKEUP_FR
EQ,         // 唤醒频率 无线唤醒时使用
33.  .p2p_config.wakeup_config.wakeup_timing     = SYS_RF_TIMER_WAK
EUP_PERIOD,// 定时唤醒周期 定时唤醒时使用
34.  .p2p_config.wakeup_config.wakeup_level      = P2P_EXTI_LOW,
          // IO 唤醒电平   IO 唤醒时使用
35.  .p2p_config.wakeup_config.wakeup_cad_period = 1000,
          // 无线唤醒周期   无线唤醒时使用
36.    };
37.    /*射频参数初始化*/
38.    tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());
39.
40.#if SYS_INFO
```

```
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);

    tk_printf("wakeup_source:\t%d\n", mid_config.wakeup_config.wakeup_source);
    tk_printf("wakeup_timing:\t%d\n", mid_config.wakeup_config.wakeup_timing);
    tk_printf("wakeup_freq:\t%d\n", mid_config.wakeup_config.wakeup_freq);
    tk_printf("wakeup_id:\t%d\n", mid_config.wakeup_config.wakeup_id);
    tk_printf("wakeup_cad_period:\t%d\n", mid_config.wakeup_config.wakeup_cad_period);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

- **sleep_config()休眠配置函数**

```
void sleep_config(void)
{
    mid_config_u mid_config;

    /* 通过定时器唤醒 */
    mid_config.p2p_config.wakeup_config.wakeup_source = P2P_WAKEUP_TIMER;
    mid_config.p2p_config.wakeup_config.wakeup_timing = 10000;
    /* 更新休眠参数 */
    tk86xx_config_update(&mid_config, P2P_WAKEUP_SRC | P2P_WAKEUP_TIMING);
    /* 休眠 */
    tk86xx_control(P2P_SLEEP, NULL, NULL);
}
```

第 6 行配置唤醒源为定时器，第 7 行配置唤醒周期为 10 秒，第 9 行调用 tk86xx_config_update 函数更新唤醒参数，第 11 行执行休眠函数 tk86xx_control。

3.9.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如下：

串口调试助手打印射频参数，工程信息和发送的数据包。

3.10 突发模式 P2P 通信带传感器从机

3.10.1 例程目的

本例程和 example_p2p _burst _slave 设计一致，这里不再赘述。

3.10.2 程序设计

略。

3.10.3 工程名称

example_p2p _lightsensor _slave

3.10.4 程序流程图

略。

3.10.5 程序分析

略。

3.10.6 运行结果

略。

3.11 P2P 通信 IO 唤醒

3.11.1 例程目的

演示终端休眠后由 IO 唤醒，并实现透传数据发送。

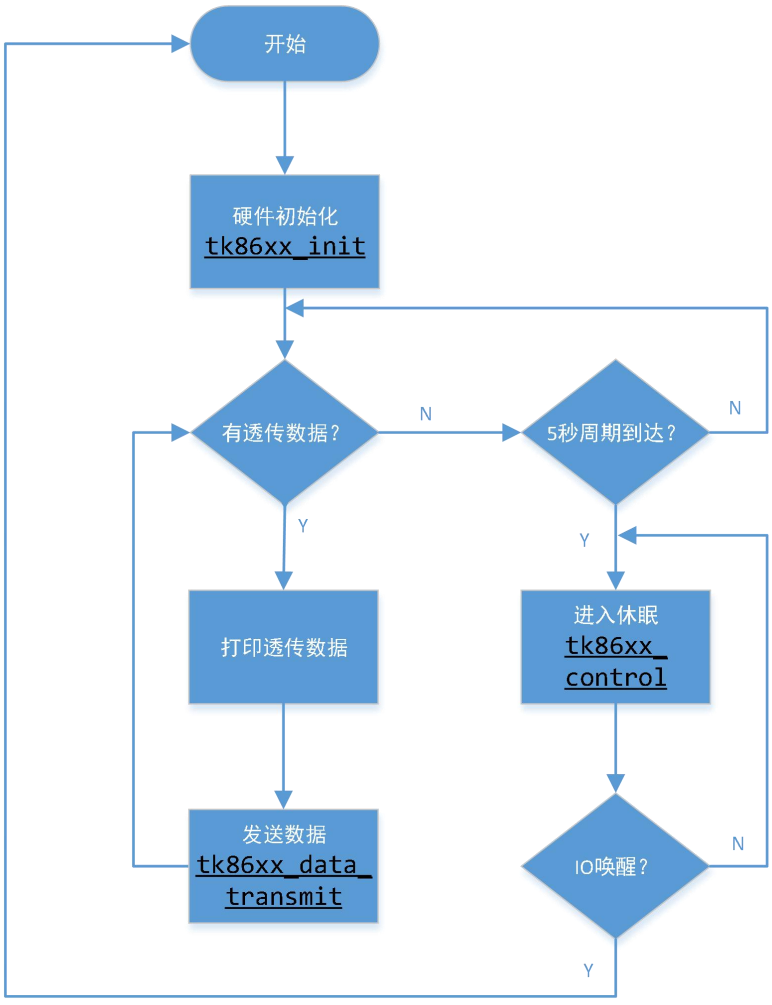
3.11.2 程序设计

本示例设计射频工作模式为变长突发模式（P2P_BURST_NORMAL），休眠后由 IO 唤醒，唤醒后可以透传数据，并通过 UART1 打印发送的数据，连续 5 秒无透传数据，则终端休眠。

3.11.3 工程名称

example_ p2p_wakeup_by_ io

3.11.4 程序流程图



3.11.5 程序分析

- **main()函数**

```
1.int main(void)
2.{
3. /*系统时钟初始化*/
4. sysctrl_sysclk_select(SYSCLK_OSC32M);
5. /*唤醒 IO 口初始化*/
6. sleep_io_init();
7. /*调试串口初始化*/
8. uart1_drv_init(BAUD_RATE_115200);
9. /*射频初始化*/
10. rf_init();
11. /*定时器初始化*/
12. timer1_drv_init();
13. /*例程信息打印*/
14. tk_printf("****** example p2p wakeup by io ******\r\n");
15. tk_printf("****** Compiled: %s %s ******\n\r", __DATE__, __TIME
__);
16. while (1)
17. {
18. /* RF 系统轮询，用到 RF 硬件资源，必须在主循环调用此接口，且主循环不能
有阻塞延时 */
19. tk86xx_poll();
20. /* 需要轮询的函数 */
21. usr_proc_poll();
22. }
23.}
```

在 main 函数中完成硬件初始化，主循环中实现 RF 系统轮询，有数据需要透传，则透传数据，连续 5 秒无数据透传，则进入休眠。

main 函数中第一步对系统时钟初始化；第二步初始化用来唤醒的 IO 端口；第三步初始化串口 1，用来打印信息；第四步初始化射频参数；第五步初始化定时器 1；第六步打印工程信息。

主循环驱动 RF 系统工作在突发模式，在此模式下，RF 始终处于空闲状态（即等待接收数据）。当用户有数据要发送的时候，会自动切换到发送状态，且在发送完成后，又会自动切换回空闲状态（等待接收数据）。用户层级的轮询函数实现数据透传发送和 5 秒周期休眠。

- **rf_init()射频初始化函数**

```
1.#define SYS_RF_WORK_MODE    P2P_BURST_NORMAL
2.#define SYS_RF_FREQ      (497100000)
3.#define SYS_RF_PWR      15
4.#define SYS_RF_RATE      P2P_RATE_13
5.#define SYS_RF_MAXBYTE         30
```

```
#define SYS_DEST_DEV_ADDR       (0xffffffff)
#define SYS_RF_WAKEUP_FREQ    470100000
#define SYS_RF_WAKEUP_ID    100
#define SYS_RF_WAKEUP_PERIOD   1000
#define SYS_RF_TIMER_WAKEUP_PERIOD   10000

void rf_init(void)
{
    /*参数配置*/
    static mid_config_u mid_config = {
        /*工作模式设置*/
  .p2p_config.work_mode                = P2P_BURST_NORMAL,    //突发模式
        /*共用参数设置*/
  .p2p_config.common_config.rate_mode  = SYS_RF_RATE,         //速率模式
  .p2p_config.common_config.bcn_mode   = P2P_BCN_AUTO,        //BCN 模式
      .p2p_config.common_config.bcn_id    = 1,                    // BCN 的 ID
  .p2p_config.common_config.bcn_freq   = SYS_RF_FREQ,         //BCN 频率
  .p2p_config.common_config.tx_freq    = SYS_RF_FREQ,         //发送频率
  .p2p_config.common_config.rx_freq    = SYS_RF_FREQ,         //接收频率
  .p2p_config.common_config.tx_power   = SYS_RF_PWR,          //发送功率
        /*突发模式参数设置*/
  .p2p_config.burst_config.dev_addr    = 1,                   //本地地址 启用地址过滤时使用
  .p2p_config.burst_config.filter_flag = P2P_FILTER_OFF,      //不启用本地地址过滤功能    本地地址过滤,与本地地址配合使用，仅用于工作模式21
        /*唤醒参数参数设置*/
  .p2p_config.wakeup_config.wakeup_source     = P2P_WAKEUP_TIMER,
        // 配置唤醒源为定时器
  .p2p_config.wakeup_config.wakeup_id         = SYS_RF_WAKEUP_ID,
        // 唤醒 ID 无线唤醒时使用
  .p2p_config.wakeup_config.wakeup_freq       = SYS_RF_WAKEUP_FREQ,
        // 唤醒频率 无线唤醒时使用
  .p2p_config.wakeup_config.wakeup_timing     = SYS_RF_TIMER_WAKEUP_PERIOD,
// 定时唤醒周期 定时唤醒时使用
  .p2p_config.wakeup_config.wakeup_level      = P2P_EXTI_LOW,
        // IO 唤醒电平    IO 唤醒时使用
```

```
    .p2p_config.wakeup_config.wakeup_cad_period = 1000,
        // 无线唤醒周期  无线唤醒时使用
    };
    /*射频参数初始化*/
    tk86xx_init(P2P_MODE, &mid_config, usr_mid_callback_get());

#if SYS_INFO
    tk_printf("\n******rf parameter******\n");
    tk_printf("\ndevice_mode:\t%d\n", mid_config.work_mode);
    tk_printf("rate_mode:\t%d\n", mid_config.common_config.rate_mode);
    tk_printf("freq:\t\t%d\n", mid_config.common_config.tx_freq);
    tk_printf("tx_power:\t%d\n", mid_config.common_config.tx_power);
    tk_printf("local dev_addr:\t%x\n", mid_config.burst_config.dev_addr);
    tk_printf("filter_flag:\t%d\n", mid_config.burst_config.filter_flag);

    tk_printf("wakeup_source:\t%d\n", mid_config.wakeup_config.wakeup_source);
    tk_printf("wakeup_timing:\t%d\n", mid_config.wakeup_config.wakeup_timing);
    tk_printf("wakeup_freq:\t%d\n", mid_config.wakeup_config.wakeup_freq);
    tk_printf("wakeup_id:\t%d\n", mid_config.wakeup_config.wakeup_id);
    tk_printf("wakeup_cad_period:\t%d\n", mid_config.wakeup_config.wakeup_cad_period);
#endif
}
```

此函数实现射频参数初始化，以及接收回调函数注册。

- **sleep_config()休眠配置函数**

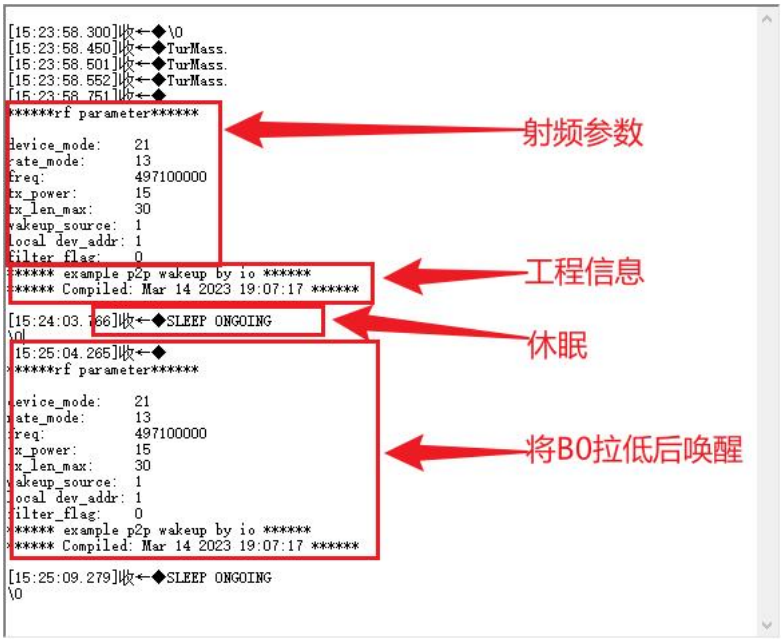
```
void sleep_config(void)
{
    mid_config_u mid_config;

    /* 通过B0低电平唤醒 */
    mid_config.p2p_config.wakeup_config.wakeup_source = P2P_WAKEUP_EXTI_B0;
```

```
7.    mid_config.p2p_config.wakeup_config.wakeup_level = P2P_EXTI_L
OW;
8.    /* 通过B0低电平唤醒，故B0初始化时为输入带上拉 */
9.    gpio_pulldirection_set(GPIO_B0, GPIO_PIN_PULLUP);
10.   tk86xx_config_update(&mid_config, P2P_WAKEUP_SRC | P2P_WAKEUP
_LEVEL);
11.    tk86xx_control(P2P_SLEEP, NULL, NULL);
12.}
```

第 5 行配置唤醒源为外部 IO B0，第 7 行配置唤醒源电平，第 9 行配置 B0 端口为输入带上拉，第 10 行调用 tk86xx_config_update 函数更新唤醒参数，第 11 行执行休眠函数。

3.11.6 运行结果

编译并下载实验代码到开发板中，打开串口调试助手，开发板上电，串口调试助手显示如下：



串口调试助手打印射频参数，工程信息和发送的数据包。