RF 模块 MD-Si4432-PHY
说明书

V1.0 2009.03

版权声明

本文件中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代，贝能国际有限公司保留对此文件修改之权利且不另行通知，请自行确定所使用之相关技术文件及规格为最新之版本。若因贵公司使用本公司之文件或产品，而涉及第三人之专利或著作权等知识产权之应用及配合时，应由贵公司负责取得同意及授权，本公司仅单纯销售产品，上述同意及授权，非属本公司应为保证之责任。

责任声明

贝能国际有限公司所提供之信息相信为正确且可靠之信息，但并不保证文件中绝无错误。确保应用符合技术规范，是您自身应负的责任。贝能国际有限公司对这些信息不作任何明示或 暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。贝能国际有限公司对因这些信息及使用这些信息而引起的后果不承担任何责任。

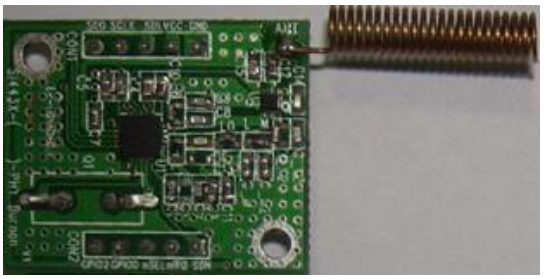
如果将贝能国际有限公司所提供的产品用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障贝能国际有限公司免于承担法律责任，并加以赔偿。

Burnon International Limited 总部位于中国香港。在福州市高新科技园区软件园设有研发中心和生产工厂，在广州设有应用设计中心。此外，贝能在开发系统的设计和生产方面的质量体系已通过了 ISO 9001:2000 认证

简述

本模块采用了 Silicon Labs 的 Si4432 作为无线收发芯片，是一块完整的、体积小巧的、低功耗无线收发模块。模块集成了芯片所需的外围器件，用户通过提供的数字 I/O 口，控制芯片内部寄存器，实现对无线数据发送、接收、RF 参数设置等功能。

基本特征:



- 完整的 FSK，GFSK,OOK 收发器
- MD-Si4432-433M-PHY，MD-Si4432-470M-PHY
  - 工作频率范围 433~470MHz
  - 发射功率典型 17dBm
  - 接收灵敏度-115 dBm@9.6Kbps
  - 空旷通讯距离 800 米以上@9.6Kbps
- 传输速率最大 128Kbps
- FSK 频偏可编程(1~160KHz)
- 接收带宽可编程(10.6~230.4KHz)
- 只需使用 5 个 I/O 口（SDI,SDO,SCK,NSEL,NIRQ)即可工作
- SPI 兼容的控制接口
- 低功耗任务周期模式,自带唤醒定时器，低关断电流（10nA）
- 1.8~3.6V 电源供电
- 低的接收电流(典型 18.5mA)
- 发射电流(典型 60mA)
- 内置温度传感器与 8 位 ADC
- 集成电压调节器
- 针对窄带系统，内置晶体负载电容微调电路，方便生产调试。

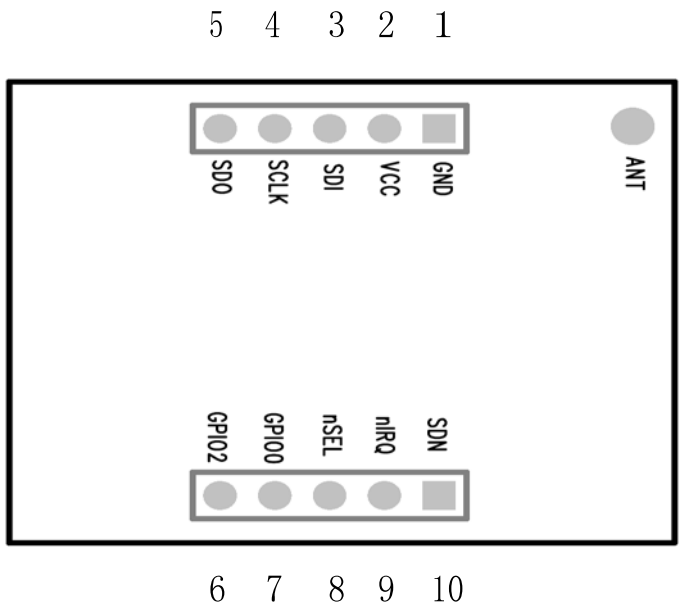
主要用途:

家居自动化和安防门禁系统
无线抄表
车辆防盗
无线传感器
远程无线数据传输
远程工业遥控，遥测

引脚说明:

引脚	定义	类型	说明
1	GND	S	电源地
2	VCC	S	供电电压
3	SDI	DI	SPI 串口数据输入
4	SCLK	DI	SPI 串口时钟输入
5	SDO	DO	SPI 串口数据输出
6	GPIO2	DO	中断请求输出/数字信号控制/接收数据输出（禁止 FIFO 模式）
7	GPIO0	DO	中断请求输出/数字信号控制/接收数据输出（禁止 FIFO 模式）
8	nSEL	DO	SPI 片选输入(低电平有效)
9	nIRQ	DO	中断请求输出(低电平有效)
10	SDN	DI	芯片关断模式输入(高电平有效)

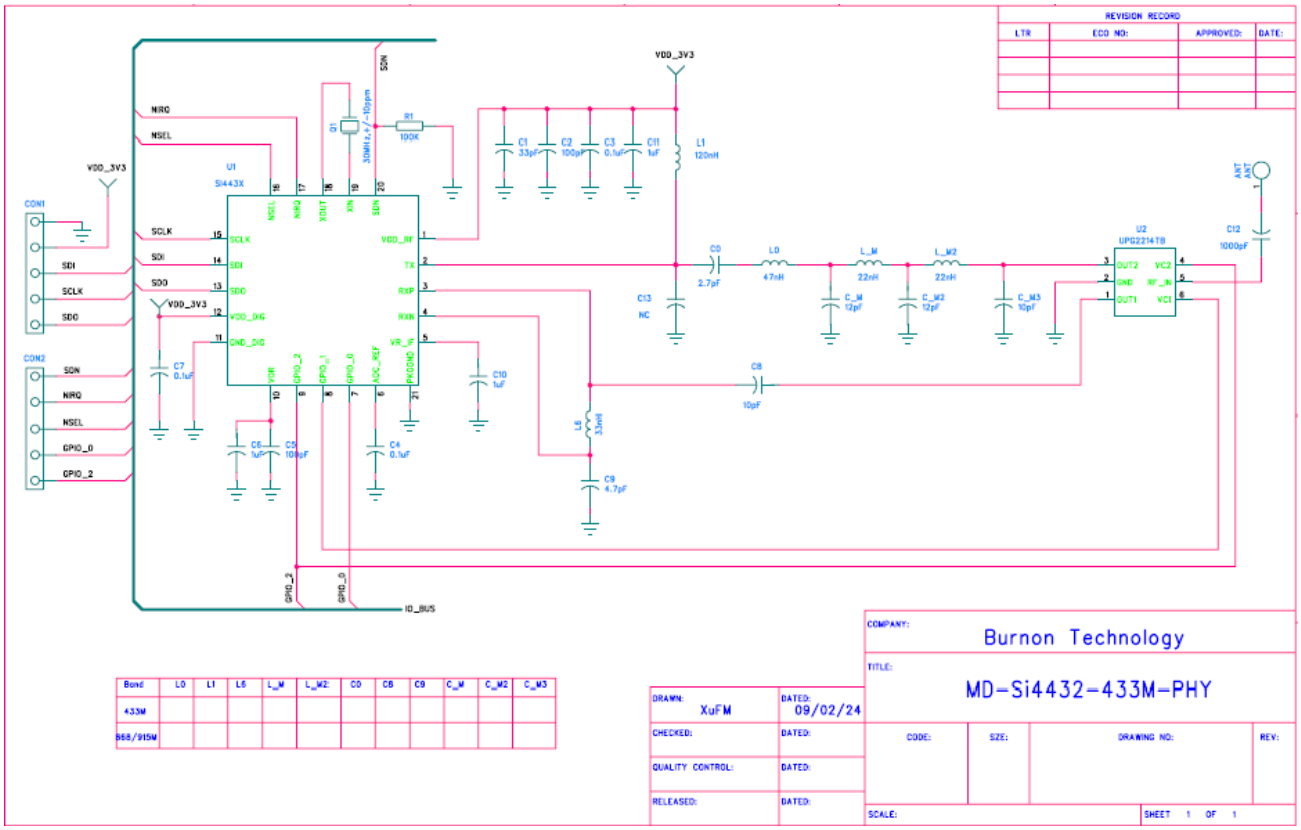
注:D=数字,A=模拟,S=供电,I=输入，O=输出,IO=输入/输出

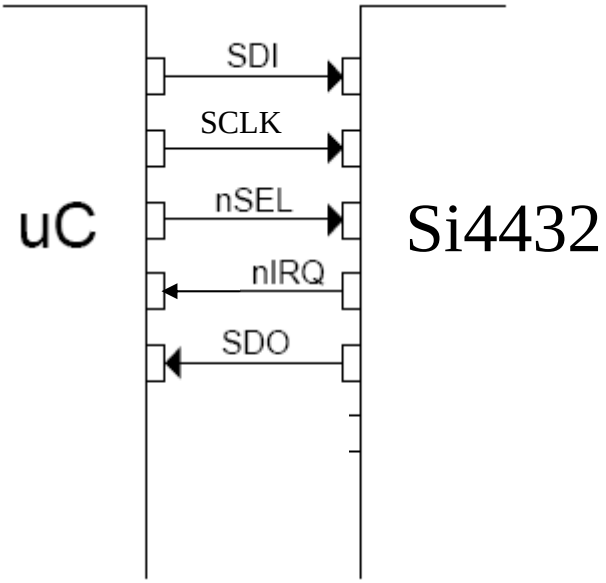


控制原理及方法：

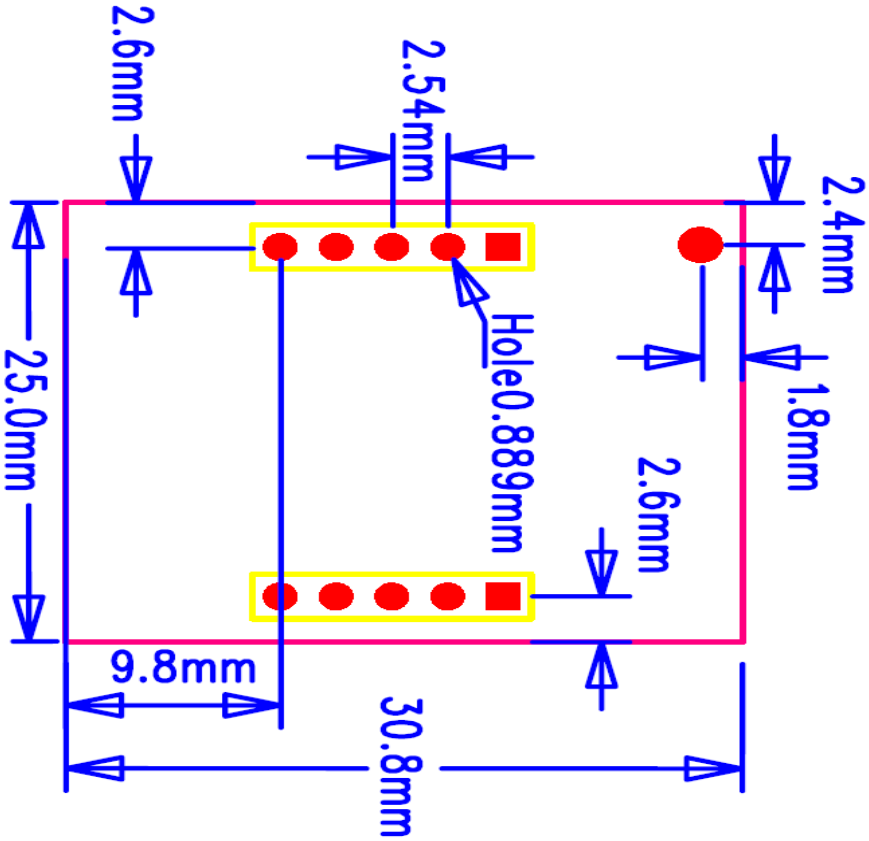
本模块是基于 Silicon Labs 的 Si4432 芯片开发出来的，是将 Si4432 及其正常工作必须的外围元器件集合在一起的一个模块，故模块的控制原理及方法等请参考 Si4432 的数据手册。

参考原理图及 MCU 连接示意图：

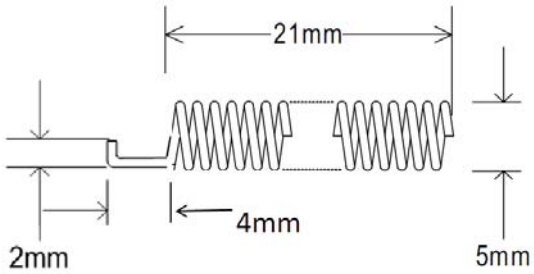




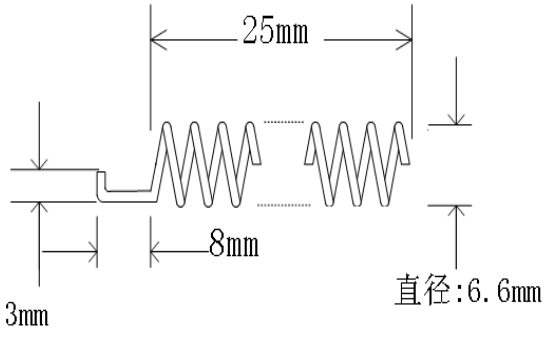
模块尺寸图：



天线信息：

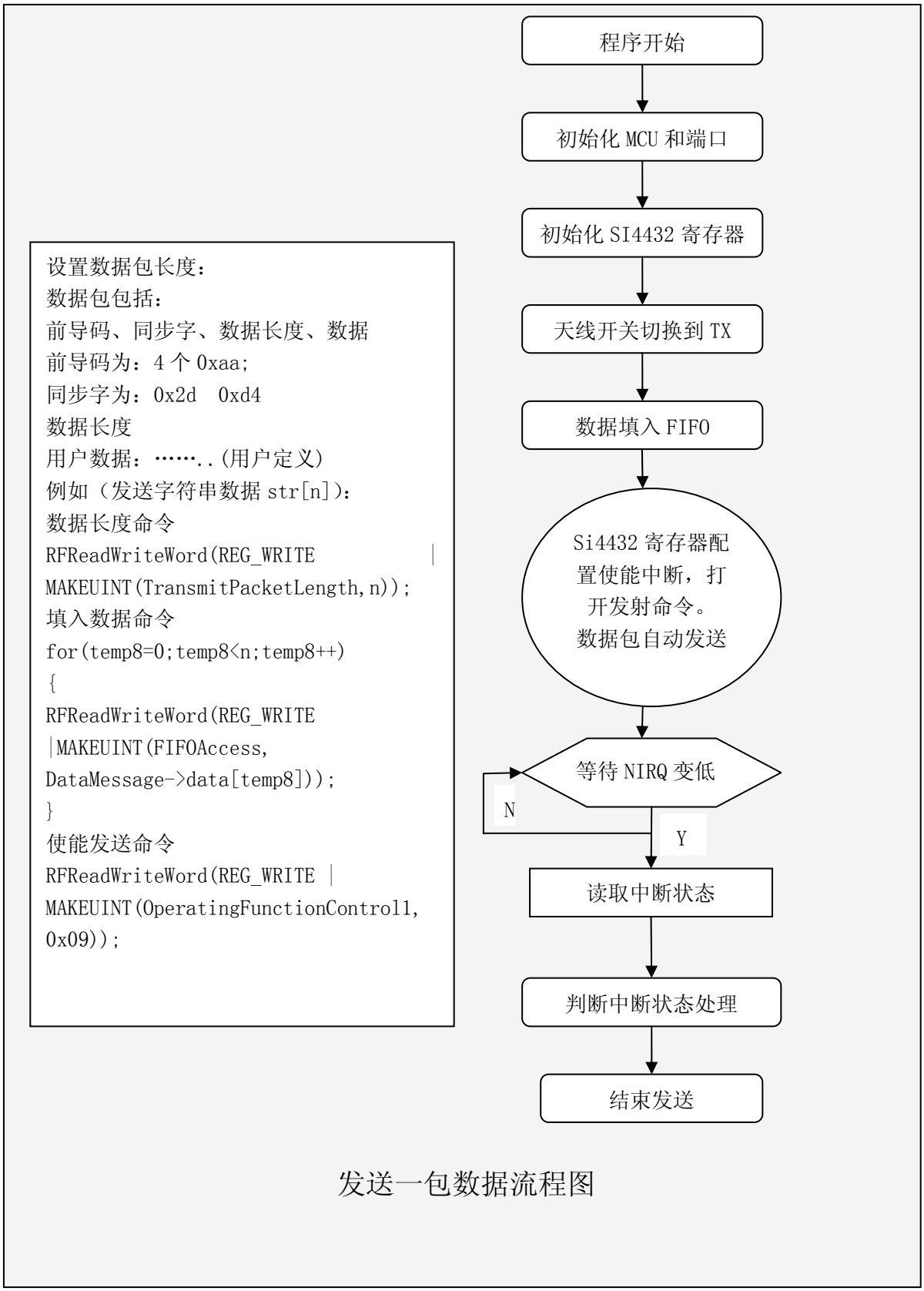


433MHz



470MHz

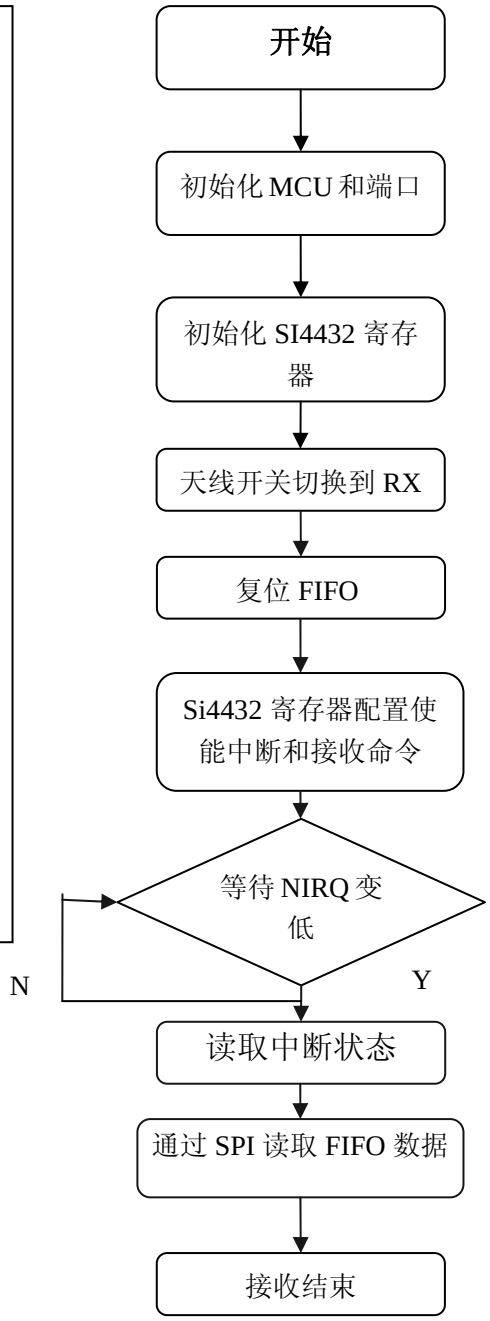
软件简单流程图：



发送一包数据流程图

1. 当接收的时候，芯片自动识别前导码和同步字，不需要干预；
2. 接收数据前，必须复位芯片 FIFO。
   例如（接收一个字符串放 buf[]），以’\0’做为结束符：

```
//clear FIFO
RFReadWriteWord(REG_WRITE
|MAKEUINT(OperatingFunction
Control2, 0x02));
RFReadWriteWord(REG_WRITE
|MAKEUINT(OperatingFunction
Control2, 0x00));
使能接收
RFReadWriteWord(REG_WRITE
|MAKEUINT(OperatingFunction
Control1 , 0x05));
While(NIRQ==0);
读取 FIFO 数据
RFGetBuffer(UCHAR * buff);
```



接收一包数据流程图

软件参考例程：

SI4432_RF.H

```
#ifndef _IA4432_RF_H_
#define  _IA4432_RF_H_
#include "mcudef.h"
#include "macrodefine.h"

//ISP  define
#define SdiIn()    (MISO_DIR=1)
#define SdiSet()    (MISO_PIN=1)
#define SdiClr()    (MISO_PIN=0)
#define SdiData()   (MISO_PIN)

#define  SdoOut()   (MOSI_DIR=0)
#define  SdoSet()   (MOSI_PIN=1)
#define  SdoClr()    (MOSI_PIN=0)
#define  SdoData()   (MOSI_PIN)

#define  SckOut()   (SCK_DIR=0)
#define  SckSet()    (SCK_PIN=1)
#define   SckClr()    (SCK_PIN=0)

#define  nIrqIn()        (RF_NIRQ_DIR=1)
#define  nIrqSet()        (RF_NIRQ_PIN=1)
#define  nIrqData()    (RF_NIRQ_PIN)

#define  nSelOut()       (RF_NSEL_DIR=0)
#define  nSelSet()         (RF_NSEL_PIN=1
#define  nSelClr()      (RF_NSEL_PIN=0)

typedef struct _HEADER
{
    UCHAR header3;
    UCHAR header2;
    UCHAR header1;
    UCHAR header0;
    UCHAR  enabled_headers;
} HEADER;
//数据包格式，用户自行定义
typedef   struct    _MESSAGE
{
    HEADER    header;
```



```
    UCHAR    length;
    UCHAR  *data;
} MESSAGE;
        /* ====================================== *
                *          D E F I N I T I O N S         *
                *          芯片寄存器定义                *
        * ====================================== */
/*definitions for the GPIO Configuration 0,1,2 registers*/
//These functions can be multiplex out to the GPIO pins
#define   GPIO_NPOR                  (0x00)    //only on GPIO0
#define   GPIO_POR                   (0x00)    //only on GPIO1
#define   GPIO_UCLK                  (0x00)    //only on GPIO2
#define   GPIO_WUT                   (0x01)
#define   GPIO_LBD                   (0x02)
#define   GPIO_DIN                   (0x03)
#define   GPIO_IT_FALLING                (0x04)
#define   GPIO_IT_RISING                 (0x05)
#define GPIO_IT_STATE                (0x06)
#define   GPIO_ADC                   (0x07)
#define   GPIO_ATB_IN_N              (0x08)
#define   GPIO_ATB_IN_P              (0x09)
#define GPIO_DOUT                    (0x0A)
#define   GPIO_DTB                   (0x0B)
#define   GPIO_ATB_OUT_N                 (0x0C)
#define   GPIO_ATB_OUT_P                 (0x0D)
#define   GPIO_REF                   (0x0E)
#define GPIO_TX_CLK                      (0x0F)
#define GPIO_TX_DATA_IN              (0x10)
#define GPIO_EXT_RETX                (0x11)
#define GPIO_TX_STATE                (0x12)
#define GPIO_TX_FIFO_AFULL           (0x13)
#define GPIO_RX_DATA                 (0x14)
#define GPIO_RX_STATE                (0x15)
#define GPIO_RX_FIFO_AFULL           (0x16)
#define GPIO_ANT1_SWITCH             (0x17)
#define GPIO_ANT2_SWITCH             (0x18)
#define GPIO_VALID_PREA              (0x19)
#define GPIO_INVALID_PREA            (0x1A)
#define GPIO_SYNCH_WORD_DET          (0x1B)
#define GPIO_CCA                     (0x1C)
#define GPIO_VDD                     (0x1D)
#define GPIO_GND                     (0x1F)

//Register setting default values
#define   RF_MCU_LF_CLK              (0)
```

```
#define   RF_MCU_CLOCK_TAIL           (0)
#define   RF_MCU_CLK_RATE                   (0)

//definitions for register usage
#define   REG_READ                    (0x0000)
#define   REG_WRITE                   (0x8000)
/*definitions for the Modulation Mode Control 2 register*/
//definitions for the TX CLK
#define   TX_CLK_OFF                        (0x00)
#define   TX_CLK_ON_GPIO              (0x40)
#define   TX_CLK_ON_SCK               (0x80)
#define TX_CLK_ON_nIRQ                (0xC0)
//definitions for the Modulation source
#define MOD_DIRECT_VIA_GPIO      (0x00)
#define MOD_DIRECT_VIA_SDI            (0x10)
#define MOD_FIFO                      (0x20)
#define MOD_PN9                           (0x30)
#define MOD_INVERT_TX                 (0x08)
//definition for the modulation type
#define MOD_TYPE_CW                   (0x00)
#define MOD_TYPE_OOK                  (0x01)
#define MOD_TYPE_FSK                  (0x02)
#define MOD_TYPE_GFSK                 (0x03)
/*definitions for the Data Access Control register*/
//definitions for the packet handler
#define TX_PACKET_HANDLER_DIS         (0x00)
#define TX_PACKET_HANDLER_EN          (0x08)
#define RX_PACKET_HANDLER_DIS         (0x00)
#define RX_PACKET_HANDLER_EN          (0x80)
#define LSB_FIRST                     (0x40)
#define MSB_FIRST                     (0x00)
#define CRC_DIS                           (0x00)
#define CRC_CCITT                     (0x04)
#define CRC_16                            (0x05)
#define CRC_IEC16                     (0x06)
#define CRC_BIACHEVA                  (0x07)
#define CRC_DATA_ONLY                 (0x20)
#define AUTO_PHASE_DETECT_MANCH       (0x10)

/*definitions for the TX Power register*/
#define RF_FULL_POWER                 (0x07)
typedef enum _RF_REG_MAP                        //4 定义 32 内部寄存器地址
{
  DeviceType                                    = 0x00,
  DeviceVersion                                     = 0x01,
```



```
DeviceStatus                                = 0x02,
InterruptStatus1                            = 0x03,
InterruptStatus2                            = 0x04,
InterruptEnable1                            = 0x05,
InterruptEnable2                            = 0x06,
OperatingFunctionControl1                   = 0x07,
OperatingFunctionControl2                   = 0x08,
CrystalOscillatorLoadCapacitance            = 0x09,
MicrocontrollerOutputClock                  = 0x0A,
GPIO0Configuration                          = 0x0B,
GPIO1Configuration                          = 0x0C,
GPIO2Configuration                          = 0x0D,
IOPortConfiguration                         = 0x0E,
ADCConfiguration                            = 0x0F,
ADCSensorAmplifierOffset                    = 0x10,
ADCValue                                    = 0x11,
TemperatureSensorControl                    = 0x12,
TemperatureValueOffset                      = 0x13,
WakeUpTimerPeriod1                          = 0x14,
WakeUpTimerPeriod2                          = 0x15,
WakeUpTimerPeriod3                          = 0x16,
WakeUpTimerValue1                           = 0x17,
WakeUpTimerValue2                           = 0x18,
LowDutyCycleModeDuration                    = 0x19,
LowBatteryDetectorThreshold                 = 0x1A,
BatteryVoltageLevel                         = 0x1B,
IFFilterBandwidth                           = 0x1C,
AFCLoopGearshiftOverride                    = 0x1D,
AFCTimingControl                            = 0x1E,
ClockRecoveryGearshiftOverride              = 0x1F,
ClockRecoveryOversamplingRatio              = 0x20,
ClockRecoveryOffset2                        = 0x21,
ClockRecoveryOffset1                        = 0x22,
ClockRecoveryOffset0                        = 0x23,
ClockRecoveryTimingLoopGain1                = 0x24,
ClockRecoveryTimingLoopGain0                = 0x25,
ReceivedSignalStrengthIndicator             = 0x26,
RSSIThresholdForClearChannelIndicator   = 0x27,
AntennaDiversityRegister1                   = 0x28,
AntennaDiversityRegister2                   = 0x29,
DataAccessControl                           = 0x30,
EZmacStatus                                 = 0x31,
HeaderControl1                              = 0x32,
HeaderControl2                              = 0x33,
PreambleLength                              = 0x34,
```

```
PreambleDetectionControl                = 0x35,
SyncWord3                               = 0x36,
SyncWord2                               = 0x37,
SyncWord1                               = 0x38,
SyncWord0                               = 0x39,
TransmitHeader3                         = 0x3A,
TransmitHeader2                         = 0x3B,
TransmitHeader1                         = 0x3C,
TransmitHeader0                         = 0x3D,
TransmitPacketLength                    = 0x3E,
CheckHeader3                                = 0x3F,
CheckHeader2                                = 0x40,
CheckHeader1                                = 0x41,
CheckHeader0                                = 0x42,
HeaderEnable3                           = 0x43,
HeaderEnable2                           = 0x44,
HeaderEnable1                           = 0x45,
HeaderEnable0                           = 0x46,
ReceivedHeader3                         = 0x47,
ReceivedHeader2                         = 0x48,
ReceivedHeader1                         = 0x49,
ReceivedHeader0                         = 0x4A,
ReceivedPacketLength                    = 0x4B,
AnalogTestBus                           = 0x50,
DigitalTestBus                          = 0x51,
TXRampControl                           = 0x52,
PLLTuneTime                                 = 0x53,
CalibrationControl                      = 0x55,
ModemTest                               = 0x56,
ChargepumpTest                          = 0x57,
ChargepumpCurrentTrimming_Override      = 0x58,
DividerCurrentTrimming                      = 0x59,
VCOCurrentTrimming                      = 0x5A,
VCOCalibration_Override                 = 0x5B,
SynthesizerTest                         = 0x5C,
BlockEnableOverride1                    = 0x5D,
BlockEnableOverride2                    = 0x5E,
BlockEnableOverride3                    = 0x5F,
ChannelFilterCoefficientAddress         = 0x60,
ChannelFilterCoefficientValue           = 0x61,
CrystalOscillator_ControlTest           = 0x62,
RCOscillatorCoarseCalibration_Override  = 0x63,
RCOscillatorFineCalibration_Override        = 0x64,
LDOControlOverride                      = 0x65,
DeltasigmaADCTuning1                    = 0x67,
```

```
  DeltasigmaADCTuning2                  = 0x68,
  AGCOverride1                          = 0x69,
  AGCOverride2                          = 0x6A,
  GFSKFIRFilterCoefficientAddress       = 0x6B,
  GFSKFIRFilterCoefficientValue         = 0x6C,
  TXPower                               = 0x6D,
  TXDataRate1                           = 0x6E,
  TXDataRate0                           = 0x6F,
  ModulationModeControl1                = 0x70,
  ModulationModeControl2                = 0x71,
  FrequencyDeviation                    = 0x72,
  FrequencyOffset                       = 0x73,
  FrequencyChannelControl               = 0x74,
  FrequencyBandSelect                   = 0x75,
  NominalCarrierFrequency1              = 0x76,
  NominalCarrierFrequency0              = 0x77,
  FrequencyHoppingChannelSelect         = 0x79,
  FrequencyHoppingStepSize              = 0x7A,
  TXFIFOControl1                        = 0x7C,
  TXFIFOControl2                        = 0x7D,
  RXFIFOControl                         = 0x7E,
  FIFOAccess                            = 0x7F,
} RF_REG_MAP;

extern MESSAGE    gcMessage;

void   RfPortInit(void);                    //初始化 spi 端口
BOOL  RfChipInit(void);                    //芯片初始化
UINT  RFReadWriteWord(UINT cmd);           //读写寄存器
void RFIdle(void);                          //进入准备状态
void  RFSleep(void);                       //进入睡眠状态
void  RFTransmitByte(UCHAR data);          //发送一个字节
void RFTransmitMessage(MESSAGE *DataMessage);      //发送一个数据结构
void RFReceiveReady(void);//start receive                 //接收准备
void RFWakeUp(void);                       //睡眠唤醒
void RFGetBuffer(UCHAR * buff);             //读取 fifo 数据
UCHAR RFGetStringMun(UCHAR *str);
#endif
```

SI4432_RF.C

```
#include "ia4432_rf.h"
//************************************************************
//函数名:VOID RfPortInit(VOID)
//功能:初始化 spi 端口设置
```

```
//************************************************************
void RfPortInit(void)
{
       SckOut();
     SckClr();
     SdiSet();
     SdiIn();
     SdoSet();
     SdoOut();
     nSelOut();
     nSelSet();
     nIrqSet();
      nIrqIn();
}
//************************************************************
//函数名:BOOL  RfChipInit(VOID)
//功能:芯片初始化
//参数:    无
//************************************************************
BOOL RfChipInit(void)
{
     UCHAR  setting=1;
     UCHAR  temp;
     //release all IT flag to set IRQ high
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
     //SW reset    -> wait for POR interrupt   复位成功则 nirq 会拉 low
     RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x80));          //4 复位所有 spi 寄存器
     while(RF_NIRQ_PIN);
     //check the status of the POR
     if( RF_NIRQ_PIN == 0x00 )
     {//POR performedcorrectly
          //disable XTAL
          RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x00));
          //set RF stack stateh
     }
     else
     {//POR error
          return FALSE;
     }
     //release all IT flag
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
     //disable all ITs, except 'ichiprdy'
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x00));
```



```
RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable2, 0x02));
// TXPower
RFReadWriteWord(REG_WRITE|MAKEUINT(TXPower,0x03));
//crystal load cap.
RFReadWriteWord(REG_WRITE | MAKEUINT(CrystalOscillatorLoadCapacitance, 0x7f)); //lzc  72
//set VCO current trimming
RFReadWriteWord(REG_WRITE | MAKEUINT(VCOCurrentTrimming, 0x7f));
RFReadWriteWord(REG_WRITE | MAKEUINT(CrystalOscillator_ControlTest, 0x00));
// Set TX Ramp Control
RFReadWriteWord(REG_WRITE | MAKEUINT(TXRampControl, 0x7f));
//disable AFC
RFReadWriteWord(REG_WRITE | MAKEUINT(AFCLoopGearshiftOverride, 0x40)); //0902190x80
RFReadWriteWord(REG_WRITE | MAKEUINT(AFCTimingControl, 0x08));
//set 10kHz offset at high band -> it helps to tune the XTAL to the correct frequency by the cap bank
RFReadWriteWord(REG_WRITE |MAKEUINT( FrequencyOffset, 0x00));
RFReadWriteWord(REG_WRITE |MAKEUINT( FrequencyChannelControl, 0x00));
//reset digital testbus, disable scan test\
RFReadWriteWord(REG_WRITE | MAKEUINT(DigitalTestBus, 0x00));
//select nothing to the Analog Testbus
RFReadWriteWord(REG_WRITE | MAKEUINT(AnalogTestBus, 0x0b));
//start the crystal oscillator
RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x01));
while(RF_NIRQ_PIN==0);
//set frequency band
RFReadWriteWord(REG_WRITE | MAKEUINT(FrequencyBandSelect,0x57));//470MHZ
//set frequency
RFReadWriteWord(REG_WRITE | MAKEUINT(NominalCarrierFrequency1, 0x4a));//0x4a));
RFReadWriteWord(REG_WRITE | MAKEUINT(NominalCarrierFrequency0, 0x00));//0x00));
//bug in revX: if txhdlen[2:0] is not 0, the length handled wrong way during RX!    设置前导
//set lower byte
RFReadWriteWord(REG_WRITE | MAKEUINT(PreambleLength, 0x08));    //4  4*16nit  = 8byte
RFReadWriteWord(REG_WRITE | MAKEUINT(PreambleDetectionControl, 0x20));
//synch word
RFReadWriteWord(REG_WRITE | MAKEUINT(HeaderControl2, 0x02));
RFReadWriteWord(REG_WRITE | MAKEUINT(SyncWord3,0x2d));
RFReadWriteWord(REG_WRITE | MAKEUINT(SyncWord2,0xd4));
//set RF chip for FIFO operation
//Modulation mode
RFReadWriteWord(REG_WRITE | MAKEUINT(ModulationModeControl1, 0x2c));
RFReadWriteWord(REG_WRITE | MAKEUINT(ModulationModeControl2, ( TX_CLK_ON_GPIO|MOD_FIFO | MOD_TYPE_FSK)));
      //lzc
//enable packet handler & CRC16
 RFReadWriteWord(REG_WRITE | MAKEUINT(DataAccessControl, (RX_PACKET_HANDLER_EN | TX_PACKET_HANDLER_EN |
CRC_16)));
```

```
    //set the registers according the selected RF settings
    RFReadWriteWord(REG_WRITE | MAKEUINT(IFFilterBandwidth, RfSettings[setting][0]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryOversamplingRatio, RfSettings[setting][1]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryOffset2, RfSettings[setting][2]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryOffset1, RfSettings[setting][3]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryOffset0, RfSettings[setting][4]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryTimingLoopGain1, RfSettings[setting][5]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(ClockRecoveryTimingLoopGain0, RfSettings[setting][6]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(TXDataRate1, RfSettings[setting][7]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(TXDataRate0, RfSettings[setting][8]));
    RFReadWriteWord(REG_WRITE | MAKEUINT(FrequencyDeviation, RfSettings[setting][9]));
    //automatic Gain Control (ONLY FOR revX2)
    RFReadWriteWord(REG_WRITE | MAKEUINT(AGCOverride1, 0x20));
    RFReadWriteWord(REG_WRITE | MAKEUINT(AGCOverride2, 0x0b));
    //override ADC refernce voltage to 0.8V
    RFReadWriteWord(REG_WRITE | MAKEUINT(DeltasigmaADCTuning2, 0x03));
    return TRUE;
}
//*************************************************************
//函数名:void  RFSleep(void)
//功能:使能睡眠
//参数:    无
//*************************************************************
void  RFSleep(void)
{
    //stop the crystal oscillator
    UCHAR temp8;
    //stop the crystal oscillator
    temp8 = (UCHAR)(0x00FF & RFReadWriteWord(OperatingFunctionControl1 << 8));
    temp8 &= 0xFE;
    RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, temp8));
}
//*************************************************************
//函数名:void  RFWakeUp(void)
//功能:唤醒
//参数:    无
//*************************************************************
void RFWakeUp(void)
{
    //start the crystal oscillator
    RFReadWriteWord(REG_WRITE |MAKEUINT( OperatingFunctionControl1, 0x01));
}
//*************************************************************
//函数名:void RFIdle(void)
//功能:  准备模式
```



```
//参数:    无
//*************************************************************
void RFIdle(void)                    //4 准备模式
{
     //switch off everything except XTAL
     RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x01));
      //clear  interrupt
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x00));
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable2, 0x00));
    //release all IT flag
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
}
//*************************************************************
//函数名:void  RFTransmitByte(UCHAR data)
//功能:  发送一个字节
//参数:    无
//*************************************************************
void  RFTransmitByte(UCHAR data)
{
     //release all IT flag
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
     if(ItStatus1&0x04==0x00)               //4 发送完毕中断标志位置位
     {
          //clear FIFO
          RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl2, 0x01));
          RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl2, 0x00));
           //set packet content
          RFReadWriteWord(REG_WRITE | MAKEUINT(TransmitPacketLength, 0x01));
          //set data
          RFReadWriteWord(REG_WRITE | MAKEUINT(FIFOAccess, data));
          //enable transmitter
          RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x09));
          //enable the wanted ITs
          RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x04));        //发送完中断允许
          RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable2, 0x00));
     }
      while(RF_NIRQ_PIN==1);
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
}
//*************************************************************
//函数名:void RFTransmitMessage(MESSAGE *DataMessage)
//功能:  发送一个数据结构
```

```
//参数:  MESSAGE 结构体
//*************************************************************
void RFTransmitMessage(MESSAGE *DataMessage)
{
      UCHAR  temp8;
     //gpio  control
      RFReadWriteWord(REG_WRITE | MAKEUINT(GPIO1Configuration, 0xca));  //0x03
      RFReadWriteWord(REG_WRITE | MAKEUINT(GPIO2Configuration, 0xca));  //0x23
      RFReadWriteWord(REG_WRITE | MAKEUINT(IOPortConfiguration, 0x02));  //0x02
      Delay(100);
     //release all IT flag
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
     //clear FIFO
     temp8=(UCHAR)(0x00FF & RFReadWriteWord(OperatingFunctionControl2<<8));//
     RFReadWriteWord( OperatingFunctionControl2 );
     temp8 |= 0x03;
     RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl2, temp8));
     temp8 &= 0xFc;
     RFReadWriteWord(REG_WRITE |MAKEUINT( OperatingFunctionControl2, temp8));
     //set headers and filters
     //bug in revX: if txhdlen[2:0] is not 0, the length handled wrong way during RX!
     temp8 =(UCHAR)(0x00FF & RFReadWriteWord(HeaderControl2 << 8));
     temp8 &= 0x07;
     RFReadWriteWord(REG_WRITE | MAKEUINT(HeaderControl2 , temp8));
    //set packet content
     RFReadWriteWord(REG_WRITE | MAKEUINT(TransmitPacketLength, DataMessage->length));
     //set data
       for(temp8=0;temp8<DataMessage->length;temp8++)
         {
            RFReadWriteWord(REG_WRITE | MAKEUINT(FIFOAccess, DataMessage->data[temp8]));
          }
     //enable transmitter
     RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1, 0x09));
     //enable the wanted ITs
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x04));      //发送完中断允许
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable2, 0x00));
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
}
//*************************************************************
//函数名:void RFReceiveReady(void)
//功能: 进入接收数据状态
//参数:    无
//*************************************************************
```



```
void RFReceiveReady(void)
{
       UCHAR temp8;
//-------------------------------------------------------------------------
//GPIO0 - RX_CLK
//gpio  control
RFReadWriteWord(REG_WRITE | MAKEUINT(GPIO1Configuration, 0xca));  //0x03
RFReadWriteWord(REG_WRITE | MAKEUINT(GPIO2Configuration, 0xca));  //0x23
RFReadWriteWord(REG_WRITE | MAKEUINT(IOPortConfiguration, 0x04));  //0x04
//-------------------------------------------------------------------------
//clear FIFO
RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl2, 0x02));
RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl2, 0x00));
//restart ADC   mcp6004
RFReadWriteWord(REG_WRITE |MAKEUINT( HeaderControl1 , 0x8c));
temp8 =(UCHAR)(0x00FF & RFReadWriteWord(HeaderControl2 << 8));
 temp8 &= 0x07;
RFReadWriteWord(REG_WRITE | MAKEUINT(HeaderControl2 , temp8));
RFReadWriteWord(REG_WRITE | MAKEUINT(DeltasigmaADCTuning1, 0x80));
RFReadWriteWord(REG_WRITE | MAKEUINT(DeltasigmaADCTuning1, 0x00));
//enable receiver chain
temp8 = (UCHAR)(0x00FF & RFReadWriteWord(OperatingFunctionControl1 << 8));
     temp8 |= 0x05;
     RFReadWriteWord(REG_WRITE | MAKEUINT(OperatingFunctionControl1 , temp8));
     //enable the wanted ITs
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x02));
     RFReadWriteWord(REG_WRITE | MAKEUINT(InterruptEnable1, 0x03));
     RFReadWriteWord(REG_WRITE |MAKEUINT( InterruptEnable2, 0x00));
     ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
     ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
}
//*************************************************************
//函数名:void RFGetBuffer(UCHAR * buff)
//功能: 对取 buff 数据
//参数:    字符串指针
//*************************************************************
void RFGetBuffer(UCHAR * buff)
{
     UCHAR k,J;
     k=  (UCHAR)(0x00FF & RFReadWriteWord(ReceivedPacketLength << 8)) ;          //读取数据长度
     for(J=0;J<k;J++)
     {
          buff[J] = (UCHAR)(0x00FF & RFReadWriteWord(FIFOAccess << 8));           //读取 fifo 数据
     }
     buff[J] =0x00;
```

```
}
//*************************************************************
//函数名:UCHAR RFGetStringMun(UCHAR *str)
//功能: 提取字符串长度
//参数: 字符串指针
//*************************************************************
UCHAR RFGetStringMun(UCHAR *str)
{
        UCHAR  temp;
        for(temp=0;str[temp]!='\0';temp++);
       return temp;
}
//*************************************************************
//函数名:UINT RFReadWriteWord(UINT cmd)
//功能: //SPI  数据读写
//参数:    无
//*************************************************************
UINT RFReadWriteWord(UINT cmd)
{
      UCHAR ri;
      UINT temp;
      temp =0;
      SckClr();                          //拉低时钟线
      nSelClr();                        //片选
      for(ri=0;ri<16;ri++)              //16 位数据
          {
            SckClr();                    //拉低时钟线
         if(0x8000&cmd)               //取出一位数据
              {
                 SdoSet();
          }
          else
              {
                 SdoClr();
          }
//      Delay(8);
         SckSet();                      //上升沿打入数据
         cmd<<=1;                   //左移一位数据
         temp <<=1;
              if(SdiData())
              {
                temp |= 0x0001;
              }
       }
     SckClr();                        //****************注意必须恢复原来的引脚状态************
```

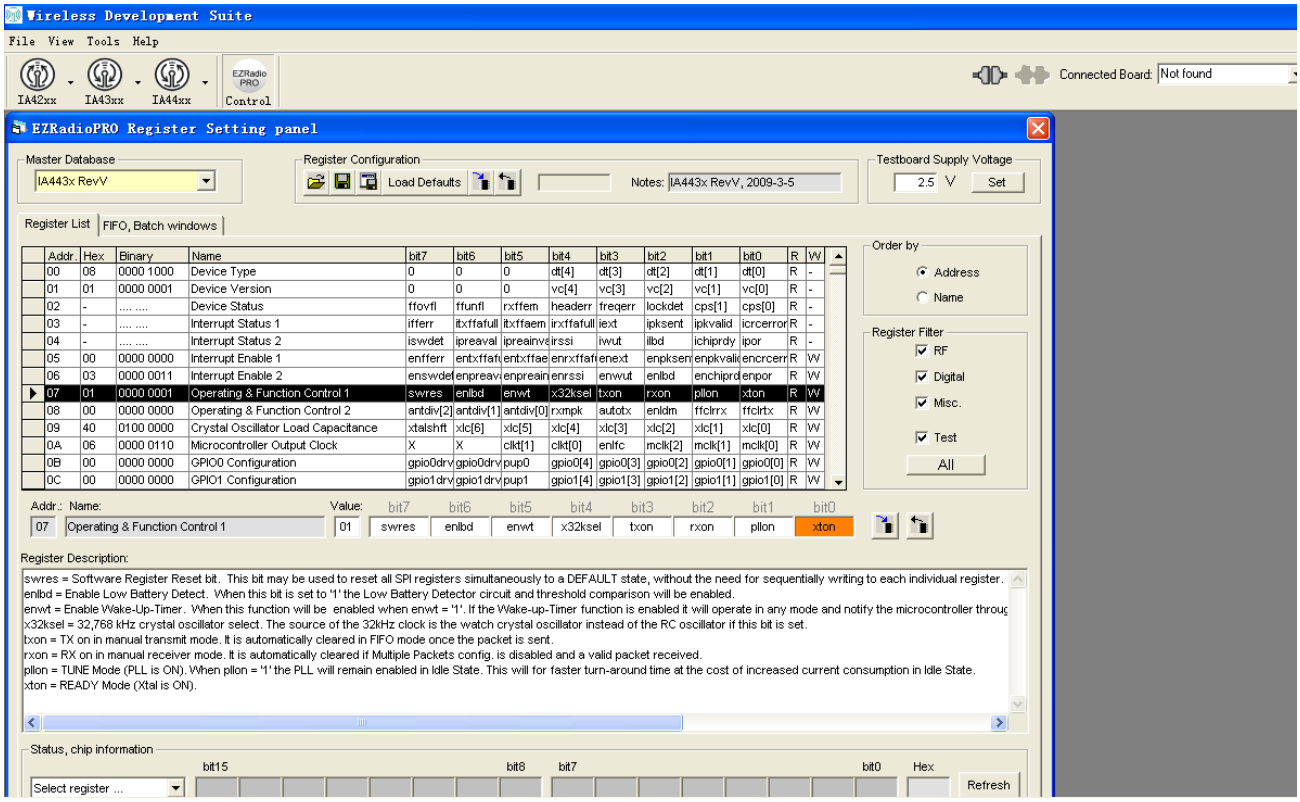


```
    nSelSet();
    return temp;
}
```

MAIN.C

```
#include "mian.h"
//*************************************************************
//函数名:VOID MAIN(VOID)
//参数:    无
//返回值:无
//*************************************************************
UCHAR  DataBuff[20];
//寄存器配置
const UCHAR RfSettings[9][11] =      //revX2
{
//    IFBW, COSR, CRO2, CRO1, CRO0, CTG1, CTG0, TDR1, TDR0, FDEV, B_TIME
    {0x17, 0x41, 0x60, 0x27, 0x52, 0x00, 0x32, 0x00, 0x9D, 0x39, 209},    //DR2400BPS_DEV36KHZ
    {0x08, 0x41, 0x60, 0x27, 0x52, 0x00, 0x0f, 0x01, 0x3B, 0x48, 105},    //DR4800BPS_DEV45KHZ
    {0x16, 0xd0, 0x00, 0x9a, 0x49, 0x00, 0x7b, 0x0c, 0xcd, 0x28, 53 },    //DR9.6KBPS_DEV40KHZ_MOD8
    {0x04, 0x90, 0x20, 0x51, 0xec, 0x02, 0x11, 0x02, 0x8F, 0x40, 50 },    //DR10KBPS_DEV40KHZ_MOD8//
    {0x04, 0xC8, 0x00, 0xA3, 0xD7, 0x00, 0x42, 0x05, 0x1F, 0x40, 25 },    //DR20K0PS_DEV40KHZ_MOD4
    {0x05, 0x64, 0x01, 0x47, 0xae, 0x05, 0x0A, 0x0A, 0x3D, 0x40, 13  },    //DR40KBPS_DEV40KHZ_MOD2
    {0x05, 0x50, 0x01, 0x99, 0x9a, 0x06, 0x68, 0x0C, 0xCD, 0x28, 10  },    //DR50KBPS_DEV25KHZ_MOD1
    {0x81, 0x78, 0x01, 0x11, 0x11, 0x01, 0x0a, 0x19, 0x9A, 0x50, 5   },    //DR 100KBPS_ DEV100KHZ_MOD1
    {0x82, 0x60, 0x01, 0x55, 0x55, 0x02, 0x2a, 0x20, 0x00, 0x64, 4   },    //DR125KBPS_DEV62KHZ5_MOD1
};
```

注：关于 Si4432 的寄存器配置帮助请登陆 www.silabs.com 下载文件 Si4432_Register_Settings_RevV-v13.xls
另外如果需要快速了解 Si4432 各个寄存器含义，请登陆 www.silabs.com 下载 WDS 软件（wireless development suit）
举例界面如下：




```
volatile UCHAR ItStatus1, ItStatus2, Rssi, ByteTime;
UCHAR  RFmodule;
UCHAR  String[]="TESTING ";
UCHAR  StringOk[]="OK";
MESSAGE    gcMessage;
volatile DEMO_STATES gcDEMO_STATES;
void main(void)
{
     UINT   count=0,count1=0,count2=0;
     UINT   ContTime;
     UCHAR con;
     UCHAR countbuf[3];
     gcDataMessage.length=8;
     gcDataMessage.data=String;
     RfPortInit();                          //初始化 spi 端口
     RfChipInit();                         //初始化 si4432 芯片
     Delay(1000);
while(1)
{
//*************************发送一个字符串*****************************************
    RFTransmitMessage(&gcDataMessage);          //发送数据

     while(nIrqData()==1);           //等待发送完毕
     if(nIrqData()==0)
          {
```



```
            //release all IT flag to set IRQ high        读取中断状态
            ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
            ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
            if( (ItStatus1 & 0x04) == 0x04 )      //    判断状态，发送完成中断?
              {
                 LED_ON();
              }
           }
     Delay_1ns(50000);
}
//*************************发送一个字符串*****************************************

//*************************接收一个字符串*****************************************
     RFReceiveReady();
     while(RF_NIRQ_PIN);   //等待 nirq 中断
     if(RF_NIRQ_PIN==0)
          {
            //release all IT flag to set IRQ high            读取中断状态
            ItStatus1 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus1 << 8));
            ItStatus2 = (UCHAR)(0x00FF & RFReadWriteWord(InterruptStatus2 << 8));
           if( (ItStatus1 & 0x02) == 0x02 )                              //判断是否是接收中断
              {
                //gcDEMO_STATES=sDemoStartReceive;
              RFGetBuffer(DataBuff);                                  //读取 FIFO 数据
              if(DataBuff[1]=='E')
              {
                   LED_ON();
              }
             }
          }
//*************************接收一个字符串*****************************************
}
}
```

销售及服务网点

Sales And Service

中国-香港特别行政区
China-Hongkong S.A.R
Tel：852-2758-0858
Fax：852-2758-0258
技术支持：Tech@burnon.com
联系销售：Sales@burnon.com

中国一福州
China-Fuzhou
Tel：86-591-8738-2588
Fax：86-591-8738-2589

中国一广州
China-Guangzhou
Tel：86-20-8554-7526
Fax：86-20-8555-2108

中国一深圳
China-Shenzhen
Tel：86-755-8384-6767
Fax：86-755-8384-6799

中国一上海
China-Shanghai
Tel：86-21-5168-8830
Fax：86-21-5420-2319

中国一珠海
China-Zhuhai
Tel：86-756-3883-919
Fax：86-756-3883-909

中国一温州
China-Wenzhou
Tel：86-577-8869-0300
Fax：86-577-8869-0191

中国一南京
China-Nanjing
Tel：86-25-8342-5993
Fax：86-25-8342-5996

中国一北京
China-Beijing
Tel：86-10-8266-5580
Fax：86-10-8266-5895

中国一沈阳
China-Shenyang
Tel：86-24-8624-4056
Fax：86-24-8620-0860

中国一青岛
China-Qingdao
Tel：86-532-8575-4275
Fax：86-532-8572-6489

中国一成都
China-Chengdu
Tel：86-28-8555-2060
Fax：86- 28-8555-2958

中国一西安
China-Xian
Tel：86-29-8832-0363
Fax：86-29-8833-8405

中国一重庆
China-Chongqin
Tel：86-23-6871-6505
Fax：86-23-6871-6505

中国一武汉
China-Wuhan
Tel：86-27-8731-3417
Fax：86-27-8731-9917

中国一厦门
China-Xiamen
Tel：86-592-5181-410
Fax：86-592-5181-430

