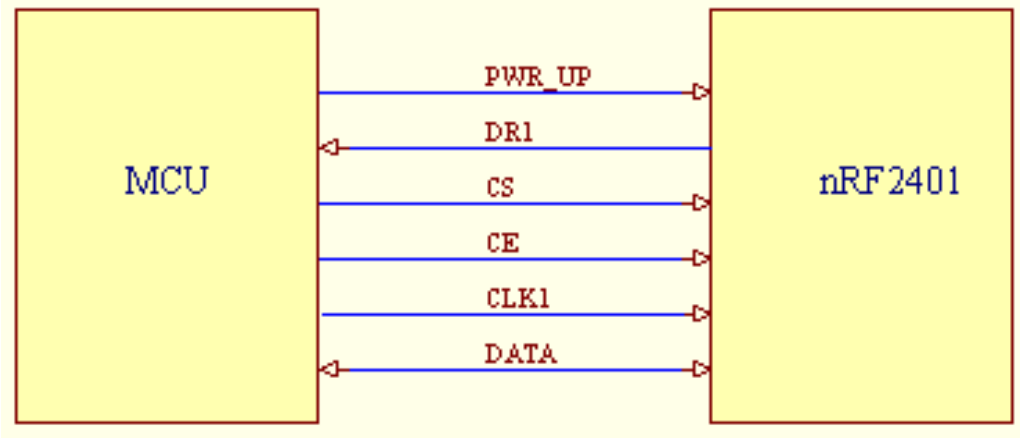
nRF2401编程指南

接口方式

nRF2401 与单片机接口可以采用 I/O 口直接连接或 SPI 接口两种方式。

1、I/O 口直接连接方式

这种方式的特点是可以方便地与各种高、低速单片机接口，方便简单。



I/O 直接接口方式图

软件编程

上电

- 设置 CS 低、CE 低、PWR_UP 高
- 延时 3ms 后，nRF2401 完成上电，进入待机模式

配置 nRF2401

- 设置 CS 高、CE 低，进入配置模式
- 延时 5μs 以上
- 单片机将配置数据通过 I/O 口写入 nRF2401（参考下面的子程序）
- 设置 CS 为低，完成配置

主机通过 nRF2401 发送数据

- 先配置 nRF2401 为 ShockBurst™ TX 模式
- 设置 CE 为高，使 2401 进入 TX 模式
- 延时 5μs 以上
- 单片机将待发送的数据通过 I/O 口写入 nRF2401 的 FIFO 缓冲区中（参考下面的 WRITE 子程序）
- 设置 CE 为低，开始 ShockBurst™模式发送

主机通过 nRF2401 接收数据

- 先配置 nRF2401 为 ShockBurst™ RX 模式
- 设置 CE 为高，202μs 后 2401 进入 RX 模式
- 当 DR1 引脚变为高电平时，表明 nRF2401 的 FIFO 缓冲区已经收到有效数据，主机可以通过查询或者中断方式进入读数据子程序，将 nRF2401 中的数据读出（参考下面的程序）
- 全部数据读完后，DR1 变为低电平

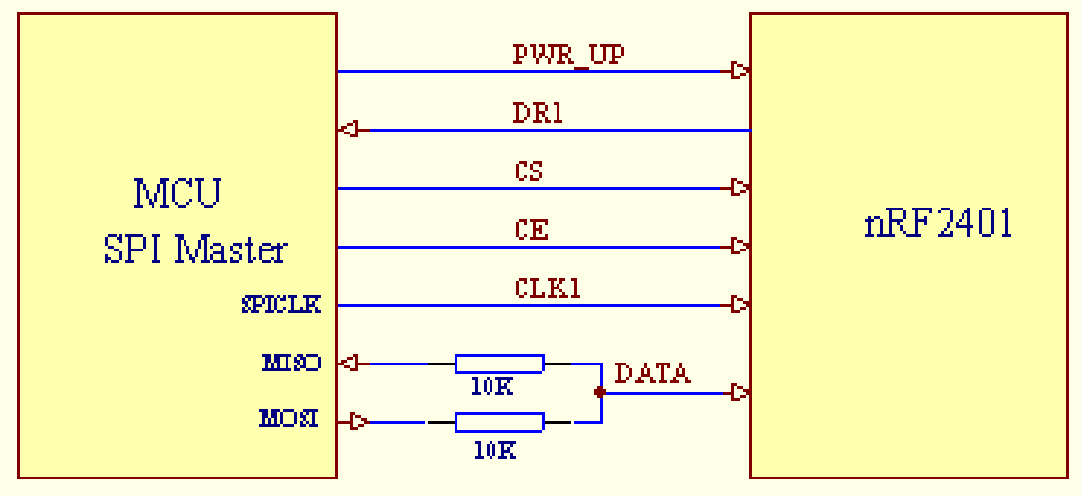
下面给出了 51 系列单片机读写 nRF2401 子程序，供参考：

```
/*************************************************************/
名      称：WRITE
功      能：通过 I/O 接口方式写 1 字节数据 C 语言子程序
入口参数：unsigned char byte     待送出的数据
出口参数：无
影响资源：  ACC
/*************************************************************/
#include <intrins.h>
sbit  BIT7      = ACC^7;
sbit  DATA      = P1.0;         /*DATA 定义为与 nRF2401 DATA 引脚相连的 I/O 脚*/
sbit  CLK1      = P1.1;         /* CLK1 定义为与 nRF2401 CLK1 引脚相连的 I/O 脚*/
void WRITE(unsigned char byte)
{unsigned char i;
      ACC = byte;               /*待发送的数据送到 ACC*/
      i = 8;                    /*共 8 位*/
      while(i)
      {
            DATA = BIT7;        /*送出 ACC 最高位到 DATA 脚*/
            CLK1 = 1;           /*将时钟信号置高*/
            _nop_（）;              /*需要根据 CPU 运行速度调整 NOP 指令的数量*/
            _nop_（）;
            ACC <<= 1;          /*ACC 左移一次*/
            CLK1 = 0;           /*将时钟信号置低*/
            i--;                /*送下一位*/
      }
}
/*************************************************************/
/*************************************************************/
名      称：READ
功      能：通过 I/O 接口方式读 1 字节数据 C 语言子程序
入口参数：  无
出口参数：unsigned char     读回的数据
影响资源：  ACC
/*************************************************************/
#include <intrins.h>
sbit  BIT0      = ACC^0;
sbit  DATA      = P1^0;         /*DATA 定义为与 nRF2401 DATA 引脚相连的 I/O 脚*/
sbit  CLK1      = P1^1;         /* CLK1 定义为与 nRF2401 CLK1 引脚相连的 I/O 脚*/
unsigned char READ(void)
{unsigned char i;
      i = 8;                    /*共 8 位*/
      while(i)
      {
            CLK1 = 1;           /*将时钟信号置高*/
            _nop_（）;          /*需要根据 CPU 运行速度调整 NOP 指令的数量*/
            ACC <<= 1;          /*ACC 左移一位*/
            BIT0 = DATA;/*从 DATA 脚读取一位数据到 ACC 最低位*/
            CLK1 = 0;           /*将时钟信号置低*/
            i--;                /*读取下一位*/
      }
return ACC;                     /*返回数据*/
}
```

```
/************************************************************/
/************************************************************/
名      称：WRITE
功      能：通过 I/O 接口方式写 1 字节数据汇编子程序
入口参数：ACC   待送出的数据
出口参数：无
影响资源：R2，ACC
/************************************************************/
DATA    EQU     P1.0 ;DATA 定义为与 nRF2401 DATA 引脚相连的 I/O 脚
CLK1    EQU     P1.1 ; CLK1 定义为与 nRF2401 CLK1 引脚相连的 I/O 脚
WRITE:
    MOV     R2   , #08
WRITE1:
    RLC A                   ; ACC 左移一位,最高位进入 CY
    MOV     DATA , C        ;将 CY 送到 DATA 引脚
    SETB    CLK1            ;将时钟信号置高
    NOP                     ;需要根据 CPU 运行速度调整 NOP 指令的数量
    NOP
    CLR CLK1                ;将时钟信号置低
    DJNZ    R2   , WRITE1
    RET
/************************************************************/
/************************************************************/
名      称：READ
功      能：通过 I/O 接口方式读 1 字节数据汇编子程序
入口参数：无
出口参数：ACC   读回的数据
影响资源：R2，ACC
/************************************************************/
DATA    EQU     P1.0 ;DATA 定义为与 nRF2401 DATA 引脚相连的 I/O 脚
CLK1    EQU     P1.1 ; CLK1 定义为与 nRF2401 CLK1 引脚相连的 I/O 脚
READ:
    SETB    DATA
    MOV     R2   ,     #08
READ1:
    SETB    CLK1            ;将时钟信号置高
    NOP                     ;需要根据 CPU 运行速度调整 NOP 指令的数量
    RLC A                   ; ACC 左移一位,CY 进入最低位
    MOV     C , DATA        ;从 DATA 脚读取一位数据到 CY
    CLR CLK1                ;将时钟信号置低
    DJNZ    R2 , READ1
    RET
/************************************************************/
```

2、SPI 接口连接方式

这种方式的特点是可以充分发挥 SPI 接口的高效以及 nRF2401 高速无线传输的优势，具有大数据吞吐量。
SPI 接口方式硬件连接图



说明：MCU 通过 MOSI 输出数据，同时通过 CLK1 送出 SPI 时钟信号，从 MISO 读入数据 ；因为图中有两个 10K 电阻隔离，MOSI 送出的数据不会影响 nRF2401 输出的数据。

软件编程
流程与上面 I/O 直接连接方式相同

读写 nRF2401 子程序

下面给出带 51 内核的 P89LPC913 单片机的 SPI 读写子程序，供参考：

```
/*************************************************************/
名      称：Spi_Init
功      能：P89LPC913 单片机 SPI 接口初始化 C 语言子程序
入口参数：  无
出口参数：  无
影响资源：  无
/*************************************************************/
sfr         SPSTAT      = 0xE1;
sfr         SPCTL       = 0xE2;
sfr         SPDAT       = 0xE3;
sfr         IEN1        = 0xE8;
sbit        ESPI        = IEN1^3 ;
void Spi_Init(void)
{
      SPCTL =0xd1;                  //bit7 :SSIG =1，忽略 SS 信号
                                    // bit6 :SPEN=1 ,允许 SPI
                                    //bit5 :DROD=0  高位在前
                                    //bit4 :MASTER =1 主模式
                                    //bit3 :CPOL=0 , SPI_CLK 空闲时为低电平
                                    //bit2 :CPHA =0 ，前时钟上升沿驱动输出及采样方式
                                    //bit1~0 :01    CCLK/16，SPI 时钟频率 1MHz max
```

```
    ESPI = 0;                   //禁止 SPI 中断
    SPSTAT = 0xc0;              // 清除 SPI 传输完成标志及写冲突标志
}

/*************************************************************/
/*************************************************************/
名      称：Spi_Init
功      能：P89LPC913 单片机 SPI 接口初始化汇编子程序
入口参数：  无
出口参数：  无
影响资源：  无
/*************************************************************/
SPSTAT      EQU     0xE1;
SPCTL       EQU     0xE2;
SPDAT       EQU     0xE3;
IEN1        EQU     0xE8;
ESPI        EQU     IEN1.3 ;

Spi_Init:
    MOV     SPCTL ,#0d1H        ;bit7 :SSIG =1，忽略 SS 信号
                                ;bit6 :SPEN=1 ,允许 SPI
                                ;bit5 :DROD=0  高位在前
                                ;bit4 :MASTER =1 主模式
                                ;bit3 :CPOL=0 , SPI_CLK 空闲时为低电平
                                ;bit2 :CPHA =0 ，前时钟上升沿驱动输出及采样方式
                                ;bit1~0 :01   CCLK/16，SPI 时钟频率 1MHz max
    CLR     ESPI                ;禁止 SPI 中断
    MOV     SPSTAT ,# 0c0H      ;清除 SPI 传输完成标志及写冲突标志
/*************************************************************/
/*************************************************************/
名      称：Spi_Read_Write
功      能：通过 SPI 接口方式读/写 1 字节数据 C 语言子程序
入口参数：  unsigned char byte   待送出的数据
出口参数：  unsigned char        读回的数据
影响资源：
/*************************************************************/
sfr     SPSTAT      = 0xE1;
sfr     SPCTL       = 0xE2;
sfr     SPDAT       = 0xE3;
sfr     IEN1        = 0xE8;
sbit    ESPI        = IEN1^3 ;

unsigned char Spi_Read_Write (unsigned char byte)
{
    SPSTAT = 0xc0;              // 清除 SPI 中断标志及写冲突标志
    SPDAT = byte;                   // 将数据写入 SPI  数据寄存器
    while((SPSTAT & 0x80) ==0);     // 等待 SPI 结束传送
    return SPDAT;                   // 将从 SPI 读到的数据返回调用程序
}
/*************************************************************/
/*************************************************************/
名      称：Spi_Read_Write
功      能：通过 SPI 接口方式读/写 1 字节数据汇编子程序
入口参数：  ACC      待送出的数据
出口参数：  ACC      读回的数据
```

影响资源： ACC

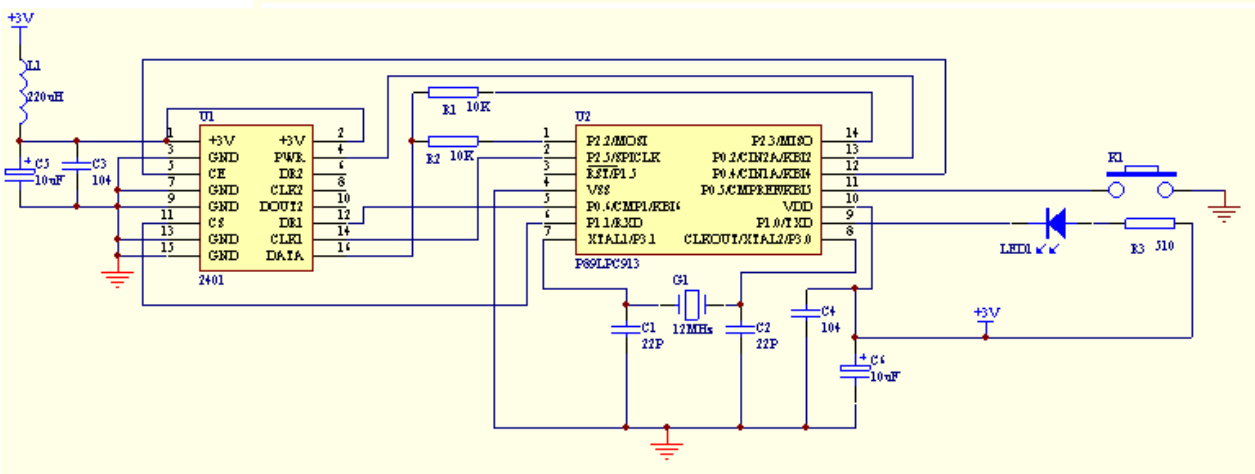
```
 /*************************************************************/
SPSTAT      EQU     0xE1;
SPCTL       EQU     0xE2;
SPDAT       EQU     0xE3;
IEN1        EQU     0xE8;
ESPI        EQU     IEN1.3 ;

Spi_Read_Write:
    MOV     SPSTAT,#11000000B       ;清除 SPI 传输完成标志及写冲突标志
    MOV     SPDAT,A                 ;将数据写入 SPI  数据寄存器
Spi_Read_Write1:
    MOV     A ,  SPSTAT             ;取 SPI 传送标志位
    JNB ACC.7 , Spi_Read_Write1     ;等待 SPI 结束传送
    MOV     A , SPDAT               ;将从 SPI 读到的数据返回调用程序
    RET
/*************************************************************/
```

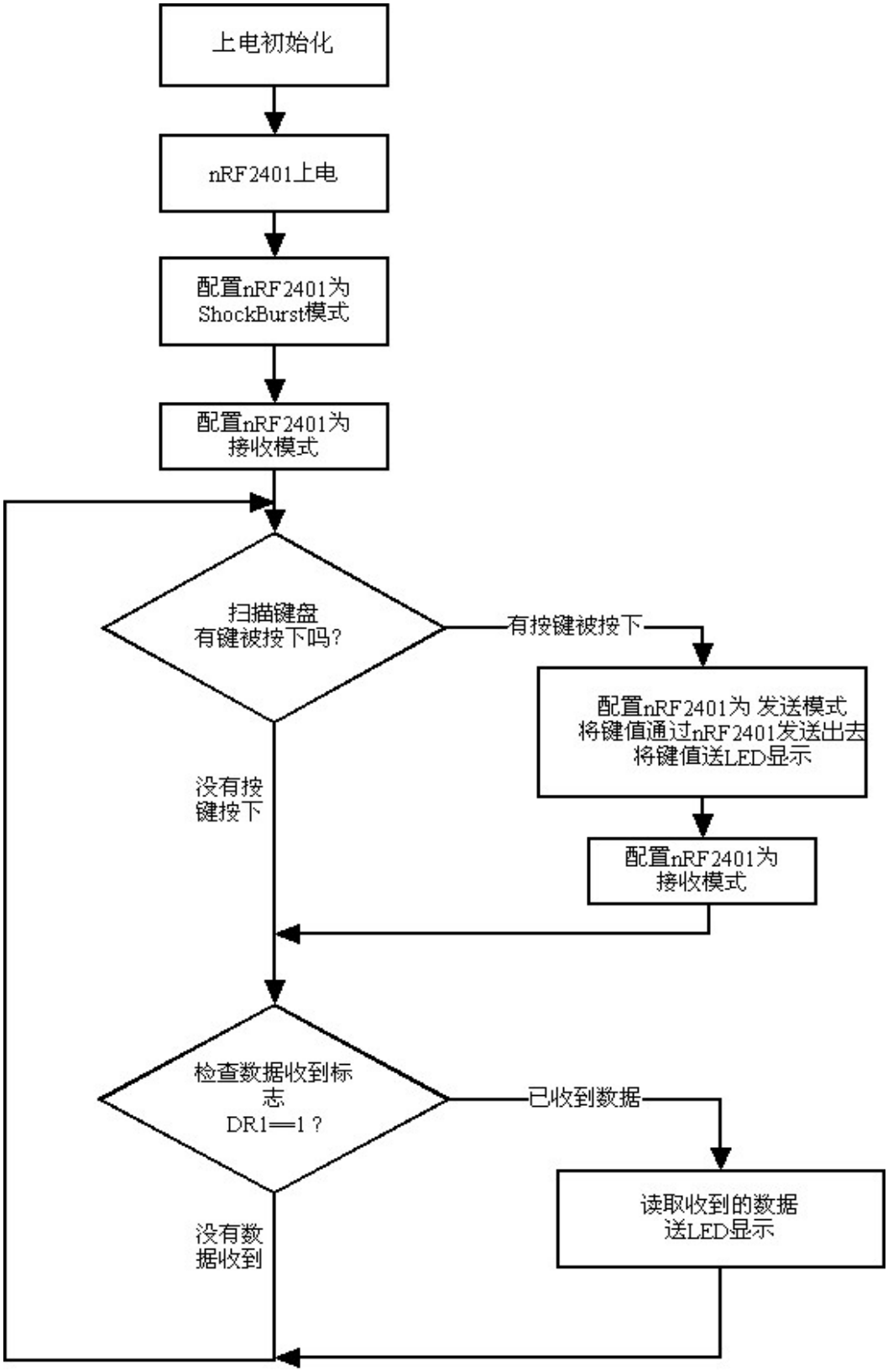
应用实例

下面给出一个以 P89LPC913 单片机和 nRF2401 构成的应用系统，单片机和 nRF2401 之间采用 SPI 接口方式。该系统由两套相同的电路板构成，功能是上电后将 nRF2401 设置为接收模式并扫描按键，如果任何一块板上的按键被按下，则 LED 点亮，单片机将 nRF2401 设置为发送模式后将一个数据包通过 nRF2401 发送出去；另外一块板上的 nRF2401 成功收到数据包后，由单片机读出并检查正确后将 LED 点亮。

原理图



程序流程图



源代码

```
/*****************************************************************************
 nRF2401 2.4GHz 参考源代码
功能：
 * 1.nRF2401 的初始化控制
 * 2.nRF2401 的数据发射与数据接收
 * 3.SPI 方式通讯
 * 4.CPU=P89LPC913
 * 5.由 Keil C51  V7.20 编译通过
 *
 * 迅通科技保留版权
*****************************************************************************/
/*****************************************************************************/
#include <reg913.h>
#include <intrins.h>
/*****************************************************************************/
struct RFConfig                         //nRF2401 配置字结构体定义
{
    unsigned char n;                    //配置字长度
    unsigned char buf[15];              //配置字数组
};

typedef struct RFConfig RFConfig;

#define ADDR_INDEX   8                  // RFConfig.buf 中的地址开始字节索引
#define ADDR_COUNT   4                  //  地址字节数

code RFConfig tconf =                   //nRF2401 发送模式配置字
{
    15,                                 //配置字长度
    0x50,                               //接收通道 2 有效数据长度值，80bit(10byte)
    0x50,                               //接收通道 1 有效数据长度值，80bit(10byte)
    0x00, 0x00, 0x00, 0x00, 0x00,       //接收通道 2 地址值，最多 40bit(5 bytes),少于 40bit 的部分设置为 0
    0x00,0xaa, 0xbb, 0x12, 0x34,        //接收通道 1 地址值，最多 40bit(5 bytes),少于 40bit 的部分设置为 0
    0x83,                               //32 bit 地址长度(bit7-bit2)；16bit CRC(bit1 0:8bit CRC/1:16bit CRC) ;
//CRC 允许(bit0 0:禁止 CRC/1:允许 CRC)
    0x6f,                               //单通道接收(bit7 0:单通道接收/1:双通道接收)；
//ShockBurst 模式(bit6 0:Direct 模式/1:ShockBurst 模式)；
//1Mbps 速率(bit5 0:250Kbps/1:1Mbps)；16MHz 晶振频率(bit4~2)；
//RF 输出功率(bit1~0)
    0x04                                //02 频道(bit7~1)；发送(bit0 0 发送:/1:接收)
};

code RFConfig rconf =                   //nRF2401 接收模式配置字
{
    15,
    0x50,
    0x50,
    0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0xaa, 0xbb, 0x12, 0x34,
    0x83,
    0x6f,
    0x05                                //02 频道(bit7~1)；接收(bit0 0 发送:/1:接收)
};
/*****************************************************************************/
unsigned char   TXData[14];             //4 字节地址+10 字节待发送的数据
unsigned char   RXData[10];             //10 字节接收数据
unsigned char   Data1=    0xff;
/*****************************************************************************/
sbit  CS   =P1^1;                       //与 nRF2401 的接口定义
sbit  CE   =P0^4;
sbit  DR1  =P0^6;
sbit  PWR_UP  =P0^2;
```

```
/*****************************************************************************/
sbit   KEY =P0^5;
sbit   LED =P1^0;
/*****************************************************************************/
void init(void);
void nRF2401_ON(void);
void Init_Receiver(void);
void Init_Transmitter(void);
void Set_TXmode(void);
void Set_RXmode(void);
void Delay10us(void);
void Delay100us(unsigned char i);
void TX_Packet（）;
void RX_Packet（）;
unsigned char Spi_Read_Write (unsigned char byte);
void CheckButtons(void);
/*************************************init     START ******************************/
void init(void)
{
//IO INIT
        P0M1=0x00;                      //P0 配置为准双向模式
        P0M2=0x00;
        P1M1=0x00;                      //P1 配置为准双向模式
        P1M2=0x00;
        P2M1=0x00;                      //P2 配置为准双向模式
        P2M2=0x00;

        P0=0Xff;
        P1=0xff;
        P2=0xff;

//SPI INIT
        SPCTL =0xd1;                    //bit7 :SSIG =1，忽略 SS 信号
                                        //bit6 :SPEN=1 ,允许 SPI
                                        //bit5 :DROD=0  高位在前
                                        //bit4 :MASTER =1 主模式
                                        //bit3 :CPOL=0 , SPI_CLK 空闲时为低电平
                                        //bit2 :CPHA =0 , 前时钟上升沿驱动输出及采样方式
                                        //bit1~0 :01    CCLK/16，SPI 时钟频率
        ESPI = 0;                       //禁止 SPI 中断
        SPSTAT = 0xc0;                  // 清除 SPI 传输完成标志及写冲突标志

        LED=0;                          //LED 点亮
        Delay100us(200);                //延时让 LED 保持点亮状态
        LED=1;                          //LED 熄灭
}
/*************************************init      END ******************************/
/*************************************nRF2401_ON START*****************************/
void nRF2401_ON(void)
{
        CE = 0;                             //CE 初始化
        CS = 0;                             //CS 初始化
        PWR_UP=1;                           //nRF2401 上电
        Delay100us(30);                 //延时 3ms 后，nRF2401 进入待机模式
}
/*************************************nRF2401_ON END ******************************/
/*************************************Init_Receiver START **************************/
void Init_Receiver(void)
{
        unsigned char b;

        CE = 0;
        CS = 1;                             //进入配置模式
        Delay10us（）;                  //延时 5us 以上
        for(b=0;b<rconf.n;b++)              //将接收模式配置字的全部 15 字节写入 nRF2401 中
```

```
        {
            Spi_Read_Write(rconf.buf[b]);
    }
        CS = 0;                                     //配置完成，进入待机状态
    CE = 1;                         //激活接收模式，经过 202us 内部延时后 nRF2401 开始接收空中的高频信号
}
/*************************************Init_Receiver END ***************************/
/*************************************Init_Transmitter START ***********************/
void Init_Transmitter(void)
{
    unsigned char b;

        CE = 0;
    CS = 1;                                     //进入配置模式
    Delay10us（）;                          //延时 5us 以上
    for(b=0;b<rconf.n;b++)                      //将发送模式配置字的全部 15 字节写入 nRF2401 中
    {
      Spi_Read_Write(tconf.buf[b]);
     }
    CS = 0;                                     //配置完成，进入待机状态
}
/*************************************Init_Transmitter END ***************************/
/*************************************Set_RXmode START ***********************/
void Set_RXmode(void)                   //用于 nRF2401 上电后，已经配置好了地址、数据长度等，快速切换到接收模式
{
        CE = 0;
        CS = 1;                                 //进入配置模式
        Delay10us（）;                          //延时 5us 以上
        Spi_Read_Write(rconf.buf[14]);  //仅将接收模式配置字的第 15 个字节写入 nRF2401 中
        CS = 0;                                 //配置完成，进入待机状态
    CE = 1;                         //激活接收模式，经过 202us 内部延时后 nRF2401 开始接收空中的高频信号
}
/*************************************Set_RXmode END ***************************/
/*************************************Set_TXmode START ***********************/
void Set_TXmode(void)      //用于 nRF2401 上电后，已经配置好了地址、数据长度等，快速切换到发送模式
{
        CE = 0;
        CS = 1;                                 //进入配置模式
        Delay10us（）;                          //延时 5us 以上
        Spi_Read_Write(tconf.buf[14]);  //仅将发送模式配置字的第 15 个字节写入 nRF2401 中
        CS = 0;                                 //配置完成，进入待机状态
}
/*************************************Set_RXmode END ***************************/
/*************************************Delay10us     START ***************************/
void Delay10us(void)
{
        unsigned char    y;

        for(y=0;y<10;y++)
            _nop_（）;
}
/*************************************Delay10us       END ***************************/
/*************************************Delay100us     START ***************************/
void Delay100us(unsigned char i)
{
        unsigned char    x;
        unsigned char    y;
        for(x=0;x<i;x++)
        {
            for(y=0;y<100;y++)
            _nop_（）;
        }
}
/*************************************Delay100us       END ***************************/
/*************************************TX_Packet START ***************************/
void TX_Packet（）
```

```
{
        unsigned char i;

        TXData[0]=0xaa;         //地址高字节   MSB
        TXData[1]=0xbb;         //地址
        TXData[2]=0x12;         //地址
        TXData[3]=0x34;         //地址低字节
        TXData[4]=Data1;        //数据 0 :发送的键值
        TXData[5]=0x02;         //数据 1
        TXData[6]=0x03;         //数据 2
        TXData[7]=0x04;         //数据 3
        TXData[8]=0x05;         //数据 4
        TXData[9]=0x06;         //数据 5
        TXData[10]=0x07;        //数据 6
        TXData[11]=0x08;        //数据 7
        TXData[12]=0x09;        //数据 8
        TXData[13]=0x10;        //数据 9    LSB

        CS=0;
        CE=1;                   //将 CE 置高
        Delay10us（）;          //延时 5us 以上，将地址和数据送到 nRF2401 的 FIFO 中
        for (i=0;i<14;i++)
        {
            Spi_Read_Write(TXData[i]);
        }
        CE=0; //将 CE 置低，经过 195us 内部延时后 nRF2401 开始高频发送，发送完成后进入待机模式
}
/************************************ TX_Packet END ********************************/
/************************************ RX_Packet START ******************************/
void RX_Packet（）
{
        unsigned char i;

        for (i=0;i<10;i++)
        {
            RXData[i] = Spi_Read_Write(0xff);
        }
}
/************************************ RX_Packet END ********************************/
/************************************ Spi_Read_Write START *************************/
unsigned char Spi_Read_Write (unsigned char byte)
{
        SPSTAT = 0xc0;                  // 清除 SPI 传输完成标志及写冲突标志
        SPDAT = byte;                   // 将数据写入 SPI  数据寄存器
        while((SPSTAT & 0x80) ==0);     // 等待 SPI 传输完成
        return SPDAT;                   // 返回从 SPI 读到的数据
}
/************************************ Spi_Read_Write END ***************************/
/************************************ CheckButtons START ***************************/
void CheckButtons(void)
{
        KEY=1;
        if (KEY == 0)                   //检查按键是否按下
        {
            Delay100us(200);            //键消抖延时
            if (KEY == 0)               //检查按键是否依然按下
            {
                Data1=0x00;             //按键已经按下，发送键值
                Init_Transmitter（）;   //配置为发送模式
                TX_Packet（）;          //发送数据包
                LED=0;                  //LED 点亮
                Delay100us(200);        //延时让 LED 保持点亮状态,同时等待 nRF2401 发送完并返回待机模式
                LED=1;                  //LED 熄灭
                Data1=0xff;
```

```
                    Set_RXmode（）;          //配置为接收模式
                }
            }
}
/************************************ CheckButtons END ********************************/
/*****************************************MAIN   START ****************************/
void main(void)
{
    init（）;                                   //上电初始化
    nRF2401_ON（）;                             //nRF2401 上电
    Init_Receiver（）;                          //配置为接收模式

    while(1)
        {
            CheckButtons（）;              //检查按键是否按下
            DR1=1;
            if(DR1==1)                     //检查 nRF2401 是否收到数据
            {
                RX_Packet（）;             //读取收到的数据
                if (RXData[0]==0x00)       //检查键值是否正确
                {
                    LED=0;                 //LED 点亮
                    Delay100us(200);       //延时让 LED 保持点亮状态
                    LED=1;                 //LED 熄灭
                }

            }
        }

}
/************************************MAIN        END ****************************/
```