```c
#include <reg51.h>
#include <intrins.h>
#include "api.h"
/*****************************************************/
#define uchar unsigned char
#define TX_ADR_WIDTH    5   // 5 bytes TX(RX) address width
#define TX_PLOAD_WIDTH  20  // 20 bytes TX payload

uchar const TX_ADDRESS[TX_ADR_WIDTH]  = {0x34,0x43,0x10,0x10,0x01}; // Define a static TX address

uchar rx_buf[TX_PLOAD_WIDTH];
uchar tx_buf[TX_PLOAD_WIDTH];
uchar flag;
/*****************************************************/
sbit CE =  P2^5;
sbit CSN=  P2^0;
sbit SCK=  P2^1;
sbit MOSI= P2^2;
sbit MISO= P2^3;
sbit IRQ = P3^2;
/*****************************************************/
uchar   bdata sta;
sbit    RX_DR   =sta^6;
sbit    TX_DS   =sta^5;
sbit    MAX_RT  =sta^4;
/*****************************************************/

/*****************************************************
Function: init_io();
Description:
  flash led one time,chip enable(ready to TX or RX Mode),
  Spi disable,Spi clock line init high
*****************************************************/
void init_io(void)
{
    P0=0x0f;          // led light
    CE=0;             // chip enable
    CSN=1;            // Spi disable
    SCK=0;            // Spi clock line init high
    P0=0xff;          // led close
}
/*****************************************************/

/*****************************************************
Function: Inituart();

Description:
  set uart working mode
*****************************************************/
void Inituart(void)
{
    TMOD = 0x20;                //timer1 working mode 1
    TL1 = 0xfd;                 //f7=9600 for 16mhz Fosc,and ...
    TH1 = 0xfd;                 //...fd=19200 for 11.0592mhz Fosc
    SCON = 0xd8;                //uart mode 3,ren==1
    PCON = 0x80;                //smod=0
    TR1 = 1;                    //start timer1
}
/*****************************************************/

/*****************************************************
Function: init_int0();
```

```
Description:
  enable int0 interrupt;
/*****************************************************/
void init_int0(void)
{
    EA=1;
    EX0=1;                          // Enable int0 interrupt.
}
/*****************************************************/

/*****************************************************
Function: delay100();

Description:
  delay 100ms
*****************************************************
void delay(uchar )
{
    uchar  x;
    uchar  y;
    for(x=0;x<100;x++)
    {
        for(y=0;y<100;y++)
        _nop_();
    }
}

/*****************************************************/
void delay_ms(unsigned int x)
{
    unsigned int i,j;
    i=0;
    for(i=0;i<x;i++)
    {
       j=108;
           ;
       while(j--);
    }
}
/*****************************************************/

/*****************************************************
Function: SPI_RW();

Description:
  Writes one byte to nRF24L01, and return the byte read
  from nRF24L01 during write, according to SPI protocol
*****************************************************/
uchar SPI_RW(uchar byte)
{
    uchar bit_ctr;
    for(bit_ctr=0;bit_ctr<8;bit_ctr++)   // output 8-bit
    {
        MOSI = (byte & 0x80);           // output 'byte', MSB to MOSI
        byte = (byte << 1);             // shift next bit into MSB..
        SCK = 1;                        // Set SCK high..
        byte |= MISO;                   // capture current MISO bit
        SCK = 0;                        // ..then set SCK low again
    }
    return(byte);                       // return read byte
}
/*****************************************************/
```

```
/*************************************************
Function: SPI_RW_Reg();

Description:
  Writes value 'value' to register 'reg'
/*************************************************/
uchar SPI_RW_Reg(BYTE reg, BYTE value)
{
    uchar status;

    CSN = 0;                   // CSN low, init SPI transaction
    status = SPI_RW(reg);      // select register
    SPI_RW(value);             // ..and write value to it..
    CSN = 1;                   // CSN high again

    return(status);            // return nRF24L01 status byte
}
/*************************************************/

/*************************************************
Function: SPI_Read();

Description:
  Read one byte from nRF24L01 register, 'reg'
/*************************************************/
BYTE SPI_Read(BYTE reg)
{
    BYTE reg_val;

    CSN = 0;                // CSN low, initialize SPI communication...
    SPI_RW(reg);            // Select register to read from..
    reg_val = SPI_RW(0);    // ..then read registervalue
    CSN = 1;                // CSN high, terminate SPI communication

    return(reg_val);        // return register value
}
/*************************************************/

/*************************************************
Function: SPI_Read_Buf();

Description:
  Reads 'bytes' #of bytes from register 'reg'
  Typically used to read RX payload, Rx/Tx address
/*************************************************/
uchar SPI_Read_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
{
    uchar status,byte_ctr;

    CSN = 0;                           // Set CSN low, init SPI tranaction
    status = SPI_RW(reg);              // Select register to write to and read status byte

    for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)
        pBuf[byte_ctr] = SPI_RW(0);    // Perform SPI_RW to read byte from nRF24L01

    CSN = 1;                           // Set CSN high again

    return(status);                    // return nRF24L01 status byte
}
/*************************************************/

/*************************************************
Function: SPI_Write_Buf();
```

```
Description:
  Writes contents of buffer '*pBuf' to nRF24L01
  Typically used to write TX payload, Rx/Tx address
/**************************************************/
uchar SPI_Write_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
{
    uchar status,byte_ctr;

    CSN = 0;                    // Set CSN low, init SPI tranaction
    status = SPI_RW(reg);    // Select register to write to and read status byte
    for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) // then write all byte in buffer(*pBuf)
        SPI_RW(*pBuf++);
    CSN = 1;                 // Set CSN high again
    return(status);          // return nRF24L01 status byte
}
/**************************************************/

/**************************************************
Function: RX_Mode();

Description:
  This function initializes one nRF24L01 device to
  RX Mode, set RX address, writes RX payload width,
  select RF channel, datarate & LNA HCURR.
  After init, CE is toggled high, which means that
  this device is now ready to receive a datapacket.
/**************************************************/
void RX_Mode(void)
{
    CE=0;
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); // Use the same address on the RX de
vice as the TX device

    SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);      // Enable Auto.Ack:Pipe0
    SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);  // Enable Pipe0
    SPI_RW_Reg(WRITE_REG + RF_CH, 40);        // Select RF channel 40
    SPI_RW_Reg(WRITE_REG + RX_PW_P0, TX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);   // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);     // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR ena
bled..

    CE = 1; // Set CE pin high to enable RX device

  //  This device is now ready to receive one packet of 16 bytes payload from a TX device sending to addr
ess
  //  '3443101001', with auto acknowledgment, retransmit count of 10, RF channel 40 and datarate = 2Mbps.

}
/**************************************************/

/**************************************************
Function: TX_Mode();

Description:
  This function initializes one nRF24L01 device to
  TX mode, set TX address, set RX address for auto.ack,
  fill TX payload, select RF channel, datarate & TX pwr.
  PWR_UP is set, CRC(2 bytes) is enabled, & PRIM:TX.

  ToDo: One high pulse(>10us) on CE will now send this
  packet and expext an acknowledgment from the RX device.
/**************************************************/
```

```
void TX_Mode(void)
{
    CE=0;

    SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);    // Writes TX_Address to nRF24L01
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); // RX_Addr0 same as TX_Adr for Auto.
Ack
    SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH); // Writes data to TX payload

    SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);      // Enable Auto.Ack:Pipe0
    SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);  // Enable Pipe0
    SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a); // 500us + 86us, 10 retrans...
    SPI_RW_Reg(WRITE_REG + RF_CH, 40);        // Select RF channel 40
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);   // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);     // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT &
TX_DS enabled..
    CE=1;

}
/**************************************************/

/**************************************************
Function: check_ACK();

Description:
  check if have "Data sent TX FIFO interrupt",if TX_DS=1,
  all led light and after delay 100ms all led close
/**************************************************
void check_ACK()
{
    uchar test;
    test=SPI_Read(READ_REG+STATUS); // read register STATUS's
    test=test&0x20;                 // check if have Data sent TX FIFO interrupt (TX_DS=1)
    if(test==0x20)                  // TX_DS =1
    {
        P0=0x00;                    // turn on all led
        delay100();                 // delay 100ms
        P0=0xff;
    }
}
/**************************************************/

/**************************************************
Function: TxData();

Description:
  write data x to SBUF
/**************************************************/
void TxData (uchar x)
{
    SBUF=x;          // write data x to SBUF
    while(TI==0);
        TI=0;
}
/**************************************************/

/**************************************************
Function: CheckButtons();

Description:
  check buttons ,if have press,read the key values,
  turn on led and transmit it;  after transmition,
  if received ACK, clear TX_DS interrupt and enter RX Mode;
```

```
  turn off the led
/*************************************************/
void CheckButtons()
{
    uchar Temp,xx,Tempi;

    P0=0xff;
    Temp=P0&0x0f;                    //read key value from port P0
    if (Temp!=0x0f)
    {
        delay_ms(10);
        Temp=P0&0x0f;               // read key value from port P0
        if (Temp!=0x0f)
        {
                xx=Temp;
                Tempi=Temp<<4;      // Left shift 4 bits
                P0=Tempi;           // Turn On the led
                tx_buf[0]=Tempi;    // Save to tx_buf[0]
                TX_Mode();          // set TX Mode and transmitting
                TxData(xx);         // send data to uart
                //check_ACK();      // if have acknowledgment from RX device,turn on all led
                SPI_RW_Reg(WRITE_REG+STATUS,SPI_Read(READ_REG+STATUS)); // clear interrupt flag(TX_DS)
                delay_ms(200);
                P0=0xff;            // Turn off the led
                RX_Mode();          // set receive mode

                while((P0&0x0f)!=0x0f);
        }
    }
}
/*************************************************/

/*************************************************
Function: main();

Description:
  control all subprogrammes;
/*************************************************/
void main(void)
{
    uchar xx;
    init_io();      // Initialize IO port
    Inituart();     // initialize 232 uart
    init_int0();    // enable int0 interrupt
    RX_Mode();      // set RX mode
    while(1)
    {
        CheckButtons(); // scan key value and transmit
        if(flag)        // finish received
        {
            flag=0;     //  set flag=0
            P0=rx_buf[0];   // turn on led
            delay_ms(200);
            P0=0xff;        // turn off led
            xx=rx_buf[0]>>4;// right shift 4 bits
            TxData(xx);     // send data to uart
        }
    }
}
/*************************************************/

/*************************************************
Function: ISR_int0() interrupt 0;
```

```
Description:
  if RX_DR=1 or TX_DS or MAX_RT=1,enter this subprogram;
  if RX_DR=1,read the payload from RX_FIFO and set flag;
/*****************************************************/
void ISR_int0(void) interrupt 0
{
    sta=SPI_Read(STATUS);    // read register STATUS's value
    if(RX_DR)                // if receive data ready (RX_DR) interrupt
    {
        SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);// read receive payload from RX_FIFO buffer
        flag=1;
    }
    if(MAX_RT)
    {
        SPI_RW_Reg(FLUSH_TX,0);
    }
        SPI_RW_Reg(WRITE_REG+STATUS,sta);// clear RX_DR or TX_DS or MAX_RT interrupt flag
}
/*****************************************************/
```