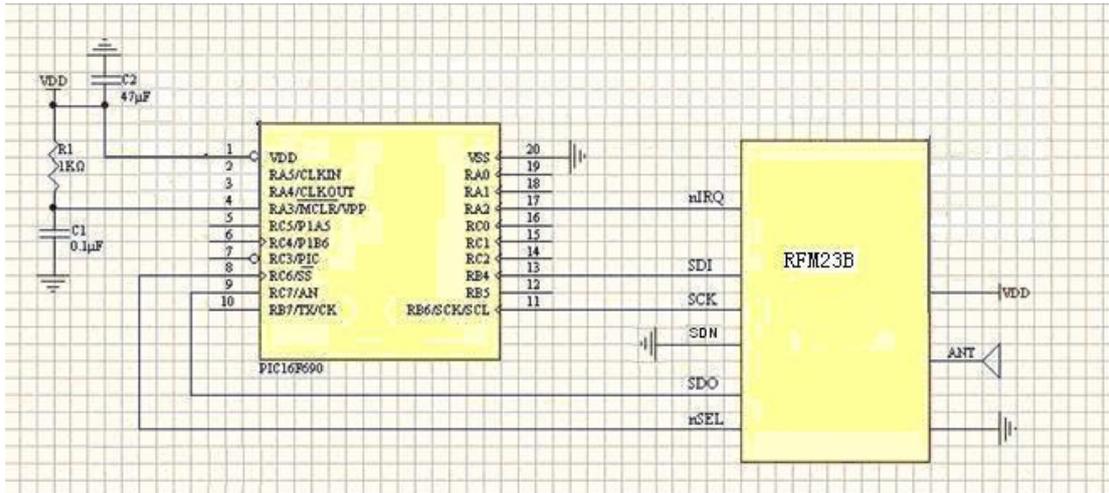


Title: RFM23B transmit demo program



Note: The SDN pin connect to GND or control by IO port of MCU. When the module is in working, the SDN pin must set to low. When the SDN pin is high, the Module current consumption is 0.01uA.

Title: RFM23B transmit demo program

Current version: v1.0

Function: Package send Demo

Processor PIC16F690 DIP-20

Clock: internal 8M.

Operate frequency: 434MHz

Data rate: 4.8kbps

modulation: FSK

deviation: 45K

frame mode: PH + FIFO

payload 0x30, 0x31...0x3f, 0x78(chksum)

Company: HopeRF microelectroni

Contact: +86-0755-82973806

Date: 2000-05-31

*/

```
#include<pic.h>
```

```
#include<math.h>
```

```
const          unsigned          char          tx_buf[17]          =
{0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f,0x78};
// ch8 is 140
```

```

unsigned char  temp0;

unsigned char send_command(void);

void to_tx_mode(void);
void to_ready_mode(void);

void send_8bit_command(unsigned char i);
void send_read_address(unsigned char i);

void spi_write(unsigned char j, unsigned char i);

void RF23B_init_parameter(void);
void delay_50ms(void);
void delay_200ms(void);
void delay_5ms(void);
void delay_1ms(void);

void port_init(void);
void power_on_delay(void);

unsigned char spi_read(unsigned char i);

void Write0( void );
void Write1( void );

void Write8bitcommand(unsigned char command);

void main()
{
    unsigned char  i, j, chksum;

    OSCCON = 0X70;    // pic osc initial

    WDTCON = 0X00;// pic watch dog initial

    power_on_delay(); // pic power on delay for system stable
    port_init();      // pic I/O port initial

    INTCON = 0x00; // interrupt disable

    RF23B_init_parameter(); // RF IC register initial

```

```

    to_tx_mode();    // start Tx
    while(1);
}

void Write0( void ) // send data 0
{
    RB6=0;
    NOP();

    RB4=0;
    NOP();

    RB6=1;
    NOP();
}

void Write1( void ) // send data 1
{
    RB6=0;
    NOP();

    RB4=1;
    NOP();

    RB6=1;
    NOP();
}

void Write8bitcommand(unsigned char command) // send 8bit command
{
    unsigned char n=8;
    RC6 = 1;    // RC6 = 1;
    RB6=0;      // RB6 = 0;
    RC6=0;      // RC6 = 0;
    while(n--)  // cycle to 8 times
    {
        if(command&0x80)// send one bit command
            Write1();
        else
            Write0();
        command = command << 1;
    }
}

```

```
}
```

```
void delay_200ms(void)
```

```
{  
    unsigned char j;  
    for(j = 0; j<40; j++)// delay 200ms = 40*5ms  
    {  
        delay_5ms();  
    }  
}
```

```
void delay_50ms(void)
```

```
{  
    unsigned char j;  
    for(j = 0; j<10; j++)  
    {  
        delay_5ms(); // delay 50ms = 10*5ms  
    }  
}
```

```
void delay_5ms(void)
```

```
{  
    int i;  
    for(i = 0; i<650; i++)  
    {  
        ; // delay 5ms  
    }  
}
```

```
void delay_1ms(void)
```

```
{  
    unsigned char i;  
    for(i = 0; i<130; i++)  
    {  
        ; // delay 1ms  
    }  
}
```

```
void port_init(void)
```

```
{  
    ANSEL = 0;  
    ANSELH = 0;  
    WPUA = 0;  
    IOCA = 0;
```

```

    TRISA = 0x0f;
    TRISB = 0x80;
    WPUB = 0x00;
    TRISC = 0b10101111;

}
void power_on_delay(void)
{
    unsigned int i;
    for(i = 0; i<1000; i++)
    {
        delay_1ms();
    }
}

```

```

unsigned char spi_read(unsigned char i)
{
    unsigned char result;
    send_read_address(i);
    result = send_command();
    RC6 = 1;
    return(result);
}

```

```

void RF23B_init_parameter(void)
{
    spi_write(0x06, 0x00); // interrupt all disable

    spi_write(0x07, 01); // to ready mode

    spi_write(0x09, 0x7f); // cap = 12.5pf

    spi_write(0x0a, 0x05); //clk output is 2MHz
    spi_write(0x0b, 0xf4); // GPIO0 is for Rx data output
    spi_write(0x0c, 0xef); // GPIO1 Tx/Rx data clk output
    spi_write(0x0d, 0x00); // GPIO2 for MCLK output
    spi_write(0x0e, 0x00); //GPIO port use default value
}

```

```

spi_write(0x0f, 0x70); // NO ADC used

spi_write(0x10, 0x00); //no adc used

spi_write(0x12, 0x00); // no temperature sensor used
spi_write(0x13, 0x00); // no temperature sensor used

spi_write(0x70, 0x20); // no mancheset code, no data whiting, data rate < 30Kbps

spi_write(0x1c, 0x1d); // IF filter bandwidth
spi_write(0x1d, 0x40); // AFC LOOP
spi_write(0x1e, 0x08); //AFC timing

spi_write(0x20, 0xa1); //clock recovery
spi_write(0x21, 0x20); //clock recovery
spi_write(0x22, 0x4e; //clock recovery
spi_write(0x23, 0xa5); //clock recovery
spi_write(0x24, 0x00); //clock recovery timing
spi_write(0x25, 0x0a); //clock recovery timing

spi_write(0x2a, 0x1e);
spi_write(0x2c, 0x29);
spi_write(0x2d, 0x04);
spi_write(0x2e, 0x29);

spi_write(0x6e, 0x27); // Tx data rate 1
spi_write(0x6f, 0x52); // Tx data rate 0

spi_write(0x30, 0x41); // data access control

spi_write(0x32, 0xff); // header control

spi_write(0x33, 0x42); // // header 3, 2, 1,0 used for head length, fixed packet length,
synchronize word length 3, 2,

spi_write(0x34, 64); // 64 nibble = 32byte preamble
spi_write(0x35, 0x20); //0x35 need to detect 20bit preamble
spi_write(0x36, 0x2d); // synchronize word
spi_write(0x37, 0xd4);
spi_write(0x38, 0x00);
spi_write(0x39, 0x00);
spi_write(0x3a, 's'); // set tx header

```

```

spi_write(0x3b, 'o');
spi_write(0x3c, 'n');
spi_write(0x3d, 'g');
spi_write(0x3e, 17); // total tx 17 byte
spi_write(0x3f, 's'); // set rx header
spi_write(0x40, 'o');
spi_write(0x41, 'n');
spi_write(0x42, 'g');
spi_write(0x43, 0xff); // all the bit to be checked
spi_write(0x44, 0xff); // all the bit to be checked
spi_write(0x45, 0xff); // all the bit to be checked
spi_write(0x46, 0xff); // all the bit to be checked

spi_write(0x56, 0x01);

spi_write(0x6d, 0x07); // tx power to Max

spi_write(0x79, 0x0); // no frequency hopping
spi_write(0x7a, 0x0); // no frequency hopping

spi_write(0x71, 0x22); // Gfsk, fd[8] =0, no invert for Tx/Rx data, fifo mode, txclk -->gpio

spi_write(0x72, 0x38); // frequency deviation setting to 35k = 56*625

spi_write(0x73, 0x0); // no frequency offset
spi_write(0x74, 0x0); // no frequency offset

spi_write(0x75, 0x53); // frequency set to 434MHz
spi_write(0x76, 0x64); // frequency set to 434MHz
spi_write(0x77, 0x00); // frequency set to 434MHz
}

void spi_write(unsigned char j, unsigned char i)
{
    j |= 0x80;
    Write8bitcommand(j); // spi write funciton
    send_8bit_command(i);
    RC6 = 1;
}

```

```

void send_read_address(unsigned char i)
{
    i &= 0x7f;           // spi read function
    Write8bitcommand(i);
}

```

```

void send_8bit_command(unsigned char i)
{
    unsigned char n = 8;
    RB6=0;
    while(n--)
    {
        if(i&0x80)
            Write1();
        else
            Write0();
        i = i << 1;
    }
    RB6=0;
}

```

```

unsigned char send_command(void)
{
    unsigned char Result, i;

    RB6=0;
    Result=0;
    for(i=0;i<8;i++)
    {
        Result=Result<<1;
        RB6=1;
        NOP();
        if(RC7)
        {
            Result|=1;
        }

        RB6=0;
        NOP();
    }
    return(Result);
}

```

```

void to_tx_mode(void)
{
    unsigned char i;

    to_ready_mode();
    delay_50ms();

    spi_write(0x08, 0x03); // fifo reset
    spi_write(0x08, 0x00);

    spi_write(0x34, 64); // preamble = 64nibble
    spi_write(0x3e, 17); // packet length = 17 bytes
    for (i = 0; i<17; i++)
    {
        spi_write(0x7f, tx_buf[i]); // send payload of the FiFo
    }
    spi_write(0x05, 04); // enable packet sent interrupt

    i = spi_read(0x03); //read the Interrupt Status1 register
    i = spi_read(0x04);

    spi_write(0x07, 09); // start tx

    while(RA2); // check if interrupt

    to_ready_mode(); // to ready mode
}

void to_ready_mode(void)
{

    spi_write(0x07, 01);

}

```