

RF12B programming guide

1. Brief description

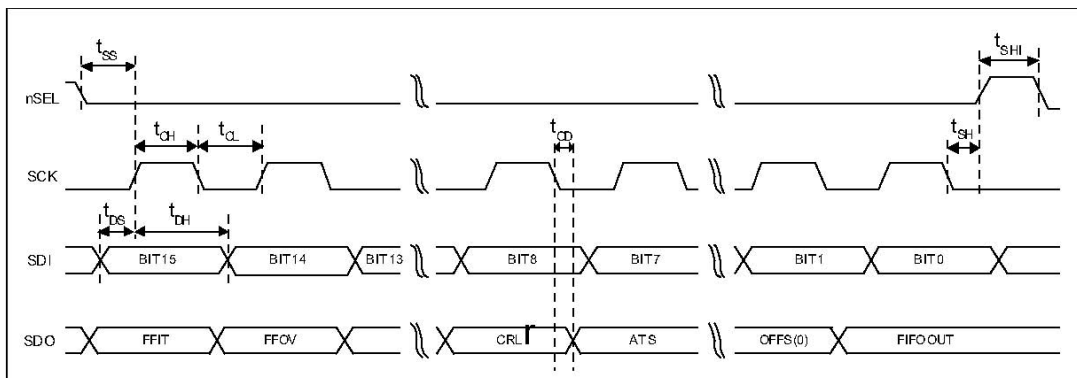
RF12B is a low cost FSK transceiver IC with integrated all RF functions in a single chip. It only need a MCU, a crystal, a decouple capacitor and antenna to build a hi reliable FSK transceiver system. The operation frequency can cover 400 to 1000MHz.

RF12B supports a command interface to setup frequency, deviation, output power and also data rate. No need any hardware adjustment when using in frequency-hopping applications

RF12B can be used in applications such as remote control toys, wireless alarm, wireless sensor, wireless keyboard/mouse, home-automation and wireless data collection.

2. Commands

1. Timing diagram



2. Configuration Setting Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | e1 | ef | b1 | b0 | x3 | x2 | x1 | x0 | 8008h |

e 1: Enable TX register

e f: Enable RX FIFO buffer

b1..b0: select band

| B1 | b0 | band[MHz] |
|----|----|-----------|
| 0 | 0 | Reserved |
| 0 | 1 | 433 |
| 1 | 0 | 868 |
| 1 | 1 | 915 |

x3..x0: select crystal load capacitor

| X3 | x2 | x1 | x0 | load capacitor [pF] |
|-------|----|----|----|---------------------|
| 0 | 0 | 0 | 0 | 8.5 |
| 0 | 0 | 0 | 1 | 9.0 |
| 0 | 0 | 1 | 0 | 9.5 |
| 0 | 0 | 1 | 1 | 10.0 |
| | | | | |
| 1 | 1 | 1 | 0 | 15.5 |
| 1 | 1 | 1 | 1 | 16.0 |

3. Power Management Command

| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
|-----|----|----|----|----|----|----|---|---|----|-----|----|----|----|----|----|----|-------|
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | er | ebb | et | es | ex | eb | ew | dc | 8208h |

- er: Enable receiver
- ebb: Enable base band block
- et: Enable transmitter
- es: Enable synthesizer
- ex: Enable crystal oscillator
- eb: Enable low battery detector
- ew: Enable wake-up timer
- dc: Disable clock output of CLK pin

4. Frequency Setting Command

| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
|-----|----|----|----|----|-----|-----|----|----|----|----|----|----|----|----|----|----|-------|
| | 1 | 0 | 1 | 0 | f11 | f10 | f9 | f8 | f7 | f6 | f5 | f4 | f3 | f2 | f1 | f0 | A680h |

f11..f0: Set operation frequency:

433band: $F_c = 430 + F * 0.0025$ MHz

868band: $F_c = 860 + F * 0.0050$ MHz

915band: $F_c = 900 + F * 0.0075$ MHz

F_c is carrier frequency and F is the frequency parameter. $36 \leq F \leq 3903$

5. Data Rate Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | cs | r6 | r5 | r4 | r3 | r2 | r1 | r0 | C623h |

r6..r0: Set data rate:

$$BR=10000000/29/ (R+1) / (1+cs*7)$$

6. Receiver Control Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 0 | 0 | 1 | 0 | P16 | d1 | d0 | i2 | i1 | i0 | g1 | g0 | r2 | r1 | r0 | 9080h |

P16: select function of pin16

| P16 | |
|-----|-----------------|
| 0 | Interrupt input |
| 1 | VDI output |

i2..i0:select baseband bandwidth

| i2 | i1 | i0 | Baseband Bandwidth [kHz] |
|----|----|----|--------------------------|
| 0 | 0 | 0 | reserved |
| 0 | 0 | 1 | 400 |
| 0 | 1 | 0 | 340 |
| 0 | 1 | 1 | 270 |
| 1 | 0 | 0 | 200 |
| 1 | 0 | 1 | 134 |
| 1 | 1 | 0 | 67 |
| 1 | 1 | 1 | reserved |

d1..d0: select VDI response time

| d1 | d0 | Response |
|----|----|-----------|
| 0 | 0 | Fast |
| 0 | 1 | Medium |
| 1 | 0 | Slow |
| 1 | 1 | Always on |

g1..g0: select LNA gain

| g1 | g0 | LNA gain (dBm) |
|----|----|----------------|
| 0 | 0 | 0 |
| 0 | 1 | -6 |
| 1 | 0 | -14 |
| 1 | 1 | -20 |

r2..r0: select DRSSI threshold

| r2 | r1 | r0 | RSSIsetth [dBm] |
|----|----|----|-----------------|
| 0 | 0 | 0 | -103 |
| 0 | 0 | 1 | -97 |
| 0 | 1 | 0 | -91 |
| 0 | 1 | 1 | -85 |
| 1 | 0 | 0 | -79 |
| 1 | 0 | 1 | -73 |
| 1 | 1 | 0 | Reserved |
| 1 | 0 | 1 | Reserved |

The actual DRSSI threshold is related to LNA setup:

$$RSSI_{th} = RSSI_{setth} + G_{LNA}$$

7. Data Filter Command

| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
|-----|----|----|----|----|----|----|---|---|----|----|---|---|---|----|----|----|-------|
| | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | al | ml | 1 | s | 1 | f2 | f1 | f0 | C22Ch |

al: Enable clock recovery auto-lock

ml: Enable clock recovery fast mode

s: select data filter type

| s | Filter type |
|---|------------------|
| 0 | Digital filter |
| 1 | Analog RC filter |

f1..f0: Set DQD threshold

8. FIFO and Reset Mode Command

| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | f3 | f2 | f1 | f0 | sp | al | ff | dr | CA80h |

f3..f0: Set FIFO interrupt level

sp: Select the length of the synchron pattern:

| sp | Byte1 | Byte0 (POR) | Synchron Pattern (Byte1+Byte0) |
|----|----------|-------------|--------------------------------|
| 0 | 2Dh | D4h | 2DD4h |
| 1 | Not used | D4h | D4h |

al: select FIFO fill start condition

| al | condition |
|----|-----------|
| 0 | Sync-word |
| 1 | Always |

ff: Enable FIFO fill

dr: Disable hi sensitivity reset mode

9. Synchron pattern Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | CED4h |

This command is used to reprogram the synchronic pattern;

10. Receiver FIFO Read Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | B000h |

This command is used to read FIFO data when FFIT interrupt generated. FIFO data output starts at 8th SCK period.

11. AFC Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|-----|-----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | a1 | a0 | r11 | r10 | st | fi | oe | en | C4F7h |

a1..a0: select AFC auto-mode:

| a1 | a0 | |
|----|----|------------------------------|
| 0 | 0 | Controlled by MCU |
| 0 | 1 | Run once at power on |
| 1 | 0 | Keep offset when VDI hi |
| 1 | 1 | Keeps independently from VDI |

r11..r10: select range limit

| r1 | r0 | range (fres) |
|----|----|----------------|
| 0 | 0 | No restriction |
| 0 | 1 | +15/-16 |
| 1 | 0 | +7/-8 |
| 1 | 1 | +3-4 |

fres

315, 433band: 2.5kHz

868band: 5kHz

915band: 7.5kHz

st: st goes hi will store offset into output register

fi: Enable AFC hi accuracy mode

oe: Enable AFC output register

en: Enable AFC function

12. TX Configuration Control Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|----|----|----|----|----|---|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | mp | m3 | m2 | m1 | m0 | 0 | p2 | p1 | p0 | 9800h |

m: select modulation polarity

m2..m0: select frequency deviation:

| m3 | m2 | m1 | m0 | frequency deviation [kHz] |
|----|----|----|----|---------------------------|
| 0 | 0 | 0 | 0 | 15 |
| 0 | 0 | 0 | 1 | 30 |
| 0 | 0 | 1 | 0 | 45 |
| 0 | 0 | 1 | 1 | 60 |
| 0 | 1 | 0 | 0 | 75 |
| 0 | 1 | 0 | 1 | 90 |
| 0 | 1 | 1 | 0 | 105 |
| 0 | 1 | 1 | 1 | 120 |
| 1 | 0 | 0 | 0 | 135 |
| 1 | 0 | 0 | 1 | 150 |
| 1 | 0 | 1 | 0 | 165 |
| 1 | 0 | 1 | 1 | 180 |
| 1 | 1 | 0 | 0 | 195 |
| 1 | 1 | 0 | 1 | 210 |
| 1 | 1 | 1 | 0 | 225 |
| 1 | 1 | 1 | 1 | 240 |

p2..p0: select output power

| p2 | p1 | p0 | Output power[dBm] |
|----|----|----|-------------------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | -3 |
| 0 | 1 | 0 | -6 |
| 0 | 1 | 1 | -9 |
| 1 | 0 | 0 | -12 |
| 1 | 0 | 1 | -15 |
| 1 | 1 | 0 | -18 |
| 1 | 0 | 1 | -21 |

13. PLL Setting Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|---|-----|-----|---|-----|-----|---|-----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | ob1 | ob0 | 1 | ddy | ddi | 1 | bw0 | CC77h |

Note1: For A0 version, the default value is CC67, it is necessary to use CC77 instead of CC67 in the application program.,

NOTE2: For A1 version, you can use the default value CC77.

ob1-ob0: Microcontroller output clock buffer rise and fall time control.

| ob1 | ob0 | Selected uC CLK frequency |
|-----|-----|---------------------------|
| 0 | 0 | 5 or 10 MHz (recommended) |
| 0 | 1 | 3.3 MHz |
| 1 | X | 2.5 MHz or less |

ddy: phase detector delay enable.

ddi: disables the dithering in the PLL loop.

bw1-bw0: select PLL bandwidth

| bw0 | Max bit rate [kbps] | Phase noise at 1MHz offset [dBc/Hz] |
|-----|---------------------|-------------------------------------|
| 0 | 86.2 | -107 |
| 1 | 256 | -102 |

14. Transmitter Register Write Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | t7 | t6 | t5 | t4 | t3 | t2 | t1 | t0 | B8AAh |

This command is use to write a data byte to RF12 and then RF12 transmit it

15. Wake-Up Timer Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 1 | r4 | r3 | r2 | r1 | r0 | m7 | m6 | m5 | m4 | m3 | m2 | m1 | m0 | E196h |

The wake-up period is determined by:

$$T_{\text{wake-up}} = M * 2^R \text{ [ms]}$$

16. Low Duty-Cycle Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | d6 | d5 | d4 | d3 | d2 | d1 | d0 | en | C80Eh |

d6..d0: Set duty cycle

$$D. C. = (D * 2 + 1) / M * 100\%$$

en: Enable low duty cycle mode

17. Low Battery Detector and Microcontroller Clock Divider Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|----|----|----|---|----|----|----|----|-------|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | d2 | d1 | d0 | 0 | v3 | v2 | v1 | v0 | C000h |

d2..d0: select frequency of CLK pin

| d2 | d1 | d0 | Clock frequency[MHz] |
|----|----|----|----------------------|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1.25 |
| 0 | 1 | 0 | 1.66 |
| 0 | 1 | 1 | 2 |
| 1 | 0 | 0 | 2.5 |
| 1 | 0 | 1 | 3.33 |
| 1 | 1 | 0 | 5 |
| 1 | 1 | 1 | 10 |

CLK signal is derive form crystal oscillator and it can be applied to MCU clock in to save a second crystal.

If not used, please set bit “dc” to disable CLK output

To integrate the load capacitor internal can not only save cost, but also adjust reference frequency by software

v3..v0: Set threshold voltage of Low battery detector:

$$V_{lb} = 2.2 + V * 0.1 \text{ [V]}$$

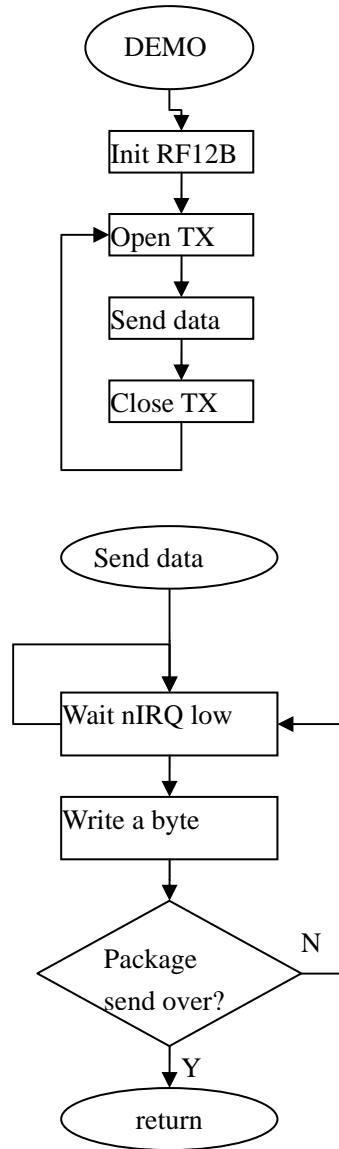
18. Status Read Command

| | | | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | POR |
| | 0 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | - |

This command starts with a 0 and be used to read internal status register

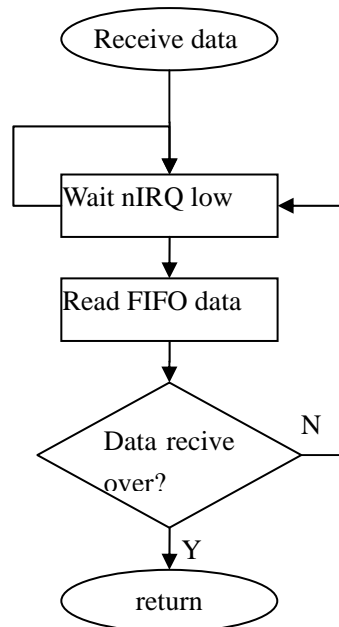
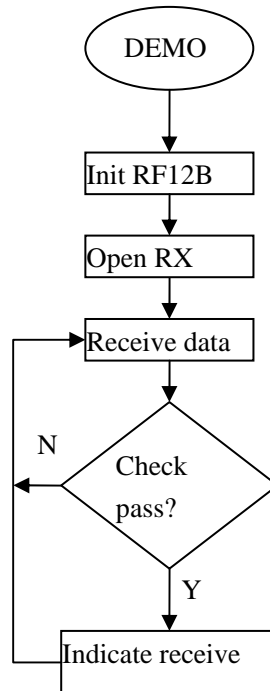
3. Demo flow diagram

Transmitter:



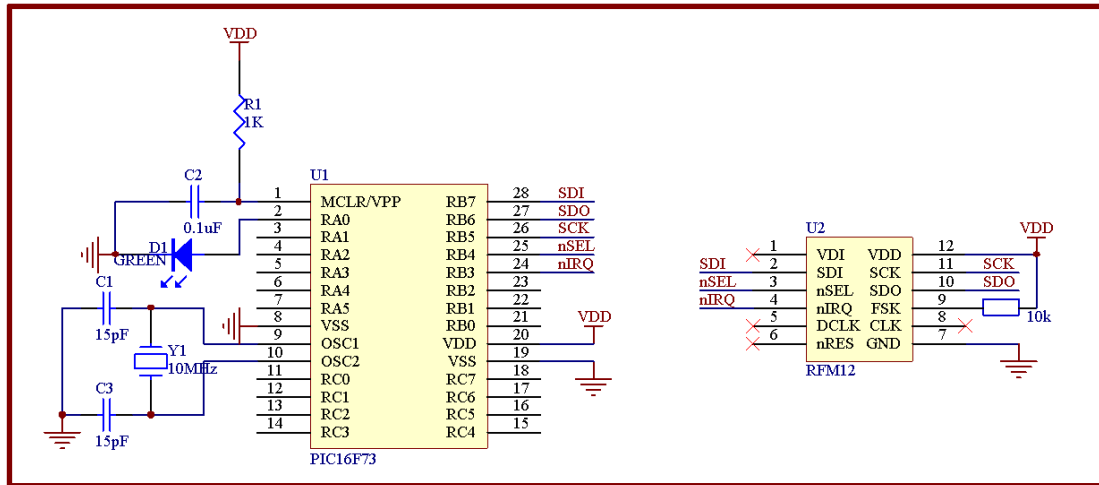
Note: Initialize RF12B and open transmitter, RF12B will transmit a byte and pull nIRQ low when transmit over, then MCU can write next byte to transmit

Receiver:



Note: After RF12B initialization, Open FIFO receive mode and wait nIRQ low, only then MCU can read received and stored in FIFO data. For next package receive, please reset FIFO.

4. (for PIC microcontroller)



RF12 transmitter demo:

/*****

copyright (c) 2010

Title: RFM12B transmitter simple example based on PIC C

Current version: v1.1

Function: Package send Demo

Processor: PIC16F73 DIP-28

Clock: 10MHz Crystal

Operate frequency: 434MHz

Data rate: 4.8kbps

Package size: 23byte

Author: Simon.Yang

Company: Hope microelectronic Co.,Ltd.

Contact: +86-0755-82973805

E-MAIL: faerf@hoperf.com

Date: 2010-06-28

*****/

#include "pic.h"

typedef unsigned char uchar;

typedef unsigned int uint;

#define SDI RB7

#define SDO RB6

#define SCK RB5

#define nSEL RB4

#define LED RA0

```
#define SDI_OUT()    TRISB7=0
#define SDO_IN()    TRISB6=1
#define SCK_OUT()   TRISB5=0
#define nSEL_OUT()  TRISB4=0
#define LED_OUT()   TRISA0=0

void Init_RF12(void);
void Write0( void );
void Write1( void );
void WriteCMD( uint CMD );
void DelayUs( uint us );
void DelayMs( uint ms );
void WriteFSKbyte( uchar DATA );

__CONFIG(0x3FF2);

void Init_RF12(void)
{
    LED_OUT();
    LED=0;
    nSEL_OUT();
    SDI_OUT();
    SDO_IN();
    SCK_OUT();
    nSEL=1;
    SDI=1;
    SCK=0;
    WriteCMD(0x80D8); //enable register, 433MHz, 12.5pF
    WriteCMD(0x8208); //Turn on crystal, !PA
    WriteCMD(0xA640); //
    WriteCMD(0xC647); //
    WriteCMD(0XCC77); //
    WriteCMD(0x94A0); //VDI, FAST, 134kHz, 0dBm, -103dBm
    WriteCMD(0xC2AC);
    WriteCMD(0xCA80);
    WriteCMD(0xCA83); //FIFO8, SYNC,
    WriteCMD(0xC49B);
    WriteCMD(0x9850); //!mp, 9810=30kHz, MAX OUT
    WriteCMD(0xE000); //NOT USE
    WriteCMD(0xC80E); //NOT USE
    WriteCMD(0xC000); //1.0MHz, 2.2V
}

void main()
```

```
{
uint ChkSum;
Init_RF12();
while(1)
{
    ChkSum=0;
    WriteCMD(0x8228);    //OPEN PA
    DelayUs( 4 );
    WriteCMD(0x8238);
    NOP();
    NOP();
    WriteFSKbyte( 0xAA );
    WriteFSKbyte( 0xAA );
    WriteFSKbyte( 0xAA );
    WriteFSKbyte( 0x2D );
    WriteFSKbyte( 0xD4 );

    WriteFSKbyte( 0x30 );//DATA0
    ChkSum+=0x30;
    WriteFSKbyte( 0x31 );//DATA1
    ChkSum+=0x31;
    WriteFSKbyte( 0x32 );
    ChkSum+=0x32;
    WriteFSKbyte( 0x33 );
    ChkSum+=0x33;
    WriteFSKbyte( 0x34 );
    ChkSum+=0x34;
    WriteFSKbyte( 0x35 );
    ChkSum+=0x35;
    WriteFSKbyte( 0x36 );
    ChkSum+=0x36;
    WriteFSKbyte( 0x37 );
    ChkSum+=0x37;
    WriteFSKbyte( 0x38 );
    ChkSum+=0x38;
    WriteFSKbyte( 0x39 );
    ChkSum+=0x39;
    WriteFSKbyte( 0x3A );
    ChkSum+=0x3A;
    WriteFSKbyte( 0x3B );
    ChkSum+=0x3B;
    WriteFSKbyte( 0x3C );
    ChkSum+=0x3C;
    WriteFSKbyte( 0x3D );
}
```

```
    ChkSum+=0x3D;
    WriteFSKbyte( 0x3E );
    ChkSum+=0x3E;
    WriteFSKbyte( 0x3F );//DATA15
    ChkSum+=0x3F;
    ChkSum&=0x0FF;
    WriteFSKbyte( ChkSum );
    WriteFSKbyte( 0xAA );
    WriteCMD( 0x8208 );    //CLOSE PA
    WriteCMD( 0x8200 );    //Receive end, enter sleep
    LED=1;
    DelayMs(100);
    LED=0;
    DelayMs(1000);
}
}
```

```
void Write0( void )
{
    SCK=0;
    NOP();
    SDI=0;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    SCK=1;
    NOP();
}
```

```
void Writel( void )
{
    SCK=0;
```

```
NOP();
SDI=1;
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
NOP();
SCK=1;
NOP();
}
```

```
void WriteCMD( uint CMD )
{
    uchar n=16;
    SCK=0;
    nSEL=0;
    while(n--)
    {
        if(CMD&0x8000)
            Write1();
        else
            Write0();
        CMD=CMD<<1;
    }
    SCK=0;
    nSEL=1;
}
```

```
void WriteFSKbyte( uchar DATA )
{
    uchar RGIT=0;
    uint temp=0xB800;
    temp|=DATA;
```

```
Loop: SCK=0;
    nSEL=0;
    SDI=0;
    SCK=1;
    if(SDO)                //Polling SDO
    {
        RGIT=1;
    }
    else
    {
        RGIT=0;
    }
    SCK=0;
    SDI=1;
    nSEL=1;
    if(RGIT==0)
    {
        goto Loop;
    }
    else
    {
        RGIT=0;
        WriteCMD(temp);
    }
}
```

```
void DelayUs( uint us )
{
    uint i;
    while( us-- )
    {
        i=2;
        while( i-- )
        {
            NOP();
        }
    }
}
```

```
void DelayMs(uint ms)
{
    uchar i;
    while(ms--)
```



```
{
  i=35;
  while(i-->0)
  {
    DelayUs(1);
  }
}
```

RF12 receiver demo:

```
/******
```

copyright (c) 2010

Title: RFM12B receive simple example based on PIC C

Current version: v1.1

Function: Package send Demo

Processor: PIC16F73 DIP-28

Clock: 10MHz Crystal

Operate frequency: 434MHz

Data rate: 4.8kbps

Package size: 23byte

Author: Simon.Yang

Company: Hope microelectronic Co.,Ltd.

Contact: +86-0755-82973805

E-MAIL: faerf@hoperf.com

Date: 2010-06-28

```
*****/
```

```
#include "pic.h"
```

```
typedef unsigned char uchar;
```

```
typedef unsigned int uint;
```

```
#define SDI RB7
#define SDO RB6
#define SCK RB5
#define nSEL RB4
#define nIRQ RB3
#define LED RA0
#define LED_OUT() TRISA0=0
#define nIRQ_IN() TRISB3=1
#define SDI_OUT() TRISB7=0
#define SDO_IN() TRISB6=1
#define SCK_OUT() TRISB5=0
```

```
#define nSEL_OUT() TRISB4=0

void Init_RF12(void);
void Write0( void );
void Write1( void );
void WriteCMD( uint CMD );
uchar RF12_RDFIFO(void);
void Delayus( uint us );

__CONFIG(0x3FF2);
bank1 uchar RF_RXBUF[19];
void Init_RF12(void)
{

    LED_OUT();
    nSEL_OUT();
    SDI_OUT();
    SDO_IN();
    SCK_OUT();
    nIRQ_IN();
    nSEL=1;
    SDI=1;
    SCK=0;
    SDO=0;
    LED=0;
    WriteCMD(0x80D8); //enable register, 433MHz, 12.5pF
    WriteCMD(0x82D8); //enable receive, !PA
    WriteCMD(0xA640); //
    WriteCMD(0xC647); //
    WriteCMD(0x94A0); //VDI, FAST, 134kHz, 0dBm, -103dBm
    WriteCMD(0xC2AC);
    WriteCMD(0XCC77); //
    WriteCMD(0xCA80);
    WriteCMD(0xCA83); //FIFO8, SYNC,
    WriteCMD(0xC49B);
    WriteCMD(0x9850); //!mp, 9810=30kHz, MAX OUT
    WriteCMD(0xE000); //NOT USE
    WriteCMD(0xC800); //NOT USE
    WriteCMD(0xC000); //1.0MHz, 2.2V
}

void main()
{
    uchar i=0, j=0;
```

```
uint CheckSum;

Init_RF12();

while(1)
{
    while(!nIRQ)
    {
        RF_RXBUF[i++]=RF12_RDFIFO();
        if(i==17)
        {
            i=0;
            WriteCMD(0xCA80);
            WriteCMD(0xCA83);           //reset FIFO and read to receive next Byte
            CheckSum=0;
            for(j=0;j<16;j++)
                CheckSum+=RF_RXBUF[j]; //add 0x30-----0x3F
            CheckSum&=0x0FF;
            if(CheckSum==RF_RXBUF[16])
            {
                LED=1;
            }
            Delayus(1);
            LED=0;
        }
    }
}

void Write0( void )
{
    SCK=0;
    NOP();
    SDI=0;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}
```

```
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    SCK=1;
    NOP();
}

void Writel( void )
{
    SCK=0;
    NOP();
    SDI=1;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    SCK=1;
    NOP();
}

void WriteCMD( uint CMD )
{
    uchar n=16;
    SCK=0;
    nSEL=0;
    while(n-->0)
    {
        if(CMD&0x8000)
            Writel();
        else
            Write0();
    }
}
```

```
    CMD=CMD<<1;
  }
  SCK=0;
  nSEL=1;
}

uchar RF12_RDFIFO(void)
{
  uchar i, Result;
  SCK=0;
  SDI=0;
  nSEL=0;
  for(i=0;i<16;i++)
  {
    //skip status bits
    SCK=1;
    NOP();
    NOP();
    SCK=0;
    NOP();
    NOP();
  }
  Result=0;
  for(i=0;i<8;i++)
  {
    //read fifo data byte
    Result=Result<<1;
    if(SDO)
    {
      Result|=1;
    }
    SCK=1;
    NOP();
    NOP();
    SCK=0;
    NOP();
    NOP();
  }
  nSEL=1;
  return(Result);
}

void Delayus( uint us )
{
  uint i;
```

```
while( us-- )
{
    i=1000;
    while( i-- )
    {
        NOP();
    }
}
```

HOPE MICROELECTRONICS CO.,LTD

Add:4/F, Block B3, East Industrial Area,
Huaqiaocheng, Shenzhen, Guangdong,
China

Tel: 86-755-86096602

Fax: 86-755-86096587

Email: sales@hoperf.com

Website: <http://www.hoperf.com>

<http://www.hoperf.cn>

<http://hoperf.en.alibaba.com>

This document may contain preliminary information and is subject to change by Hope Microelectronics without notice. Hope Microelectronics assumes no responsibility or liability for any use of the information contained herein. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Hope Microelectronics or third parties. The products described in this document are not intended for use in implantation or other direct life support applications where malfunction may result in the direct physical harm or injury to persons. NO WARRANTIES OF ANY KIND, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MECHANICALITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.

©2006, HOPE MICROELECTRONICS CO.,LTD. All rights reserved.

