

RF02 programming guide

1. Brief description

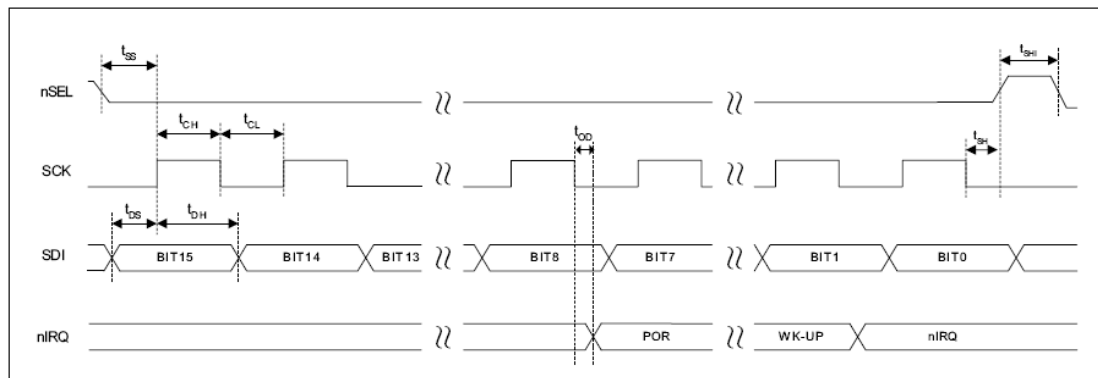
RF02 is a low cost FSK transmit IC witch integrated all RF functions in a single chip. It only need a MCU, a crystal, a decouple capacitor and antenna to build a hi reliable FSK transmitter. The operation frequency can cover 300 to 1000MHz.

RF02 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

RF02 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	b1	b0	d2	d1	d0	x3	x2	x1	x0	ms	m2	m1	m0	8080h

b1..b0: band select:

b1	b0	band[MHz]
0	1	433
1	0	868
1	1	915

d2..d0: select frequency of CLK pin

d2	d1	d0	CLK 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

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

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

ms: select modulation polarity

m2..m0: select frequency deviation

m2	m1	m0	frequency deviation[kHz]
0	0	0	30
0	0	1	60
0	1	0	90
0	1	1	120
1	0	0	150
1	0	1	180
1	1	0	210

3. Power Management 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	a1	a0	ex	es	ea	eb	et	dc	C000h

a1: Crystal oscillator and synthesizer are enabled by Data transmit Command and disable by Sleep command.

a0: Power amplifier is enabled by Data transmit Command and disable by Sleep Command.

- ex: Enable crystal oscillator
- es: Enable synthesizer
- ea: Enable power amplifier
- eb: Enable low battery detection function
- et: Enable wake-up timer
- dc: Disable 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	A7D0h

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

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	1	0	0	0	r7	r6	r5	r4	r3	r2	r1	r0	C800h

r7..r0: set data rate

$BR = 10000000 / 29 / (R + 1)$

BR is data rate

6. Power Setting Command

bit	7	6	5	4	3	2	1	0	POR
	1	0	1	1	0	p2	p1	p0	B0h

p2..p0: set relative output power:

$P_{out} = P_{max} - P * 3$ [dBm]

P_{max} is the max output power; it is related to the antenna impedance.

7. Low Battery Detector and Tx bit Synchronization 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	dwc	0	ebs	t4	t3	t2	t1	t0	C200h

dwc: Disable wake-up timer periodical calibration

ebs: Enable TX bit synchronization function

t4..t0: Set threshold voltage of Low battery detector

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

8. Sleep 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	s7	s6	s5	s4	s3	s2	s1	s0	C400h

If crystal oscillator, synthesizer and power amplifier are auto-controlled, this command will close power amplifier and synthesizer immediately, then stop crystal oscillator after S periods of CLK signal

9. 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	E000h

The wake-up timer period is determined by:

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

For continual operation, bit 'et' must be cleared and set

10. Data Transmit Command

bit	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	1	0

This command indicate that the following data on SDI pin is to be transmitted, the transmission stops if nSel return to hi.

11. Status Register Read 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	0	0	0	0	0	0	0	--

This command is used to read internal status register content, output starts at 8th clock of SCK.

12. PLL Setting and Reset Mode Command

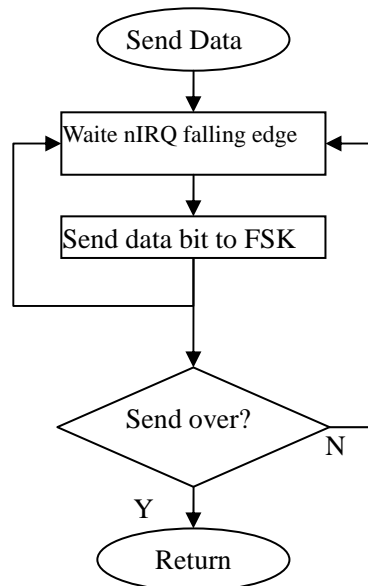
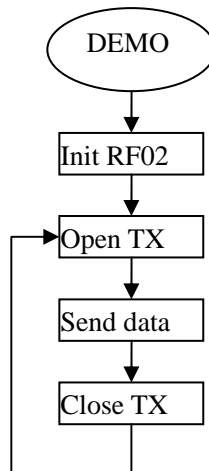
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	POR
	1	1	0	1	0	0	1	0	bw1	bw0	0	0	0	0	dr	0	D200h

Bits 7-6 <bw1 : bw0> select the PLL bandwidth:

Bw1	Bw0	Max datarate [kbps]	Phase Noise at 1MHz offset [dBc/Hz](typical)	Charge pump current
0	1	19.2	-112	25%
1	1	38.4	-110	33%
0	0	68.9	-107	50%
1	0	115.2	-102	100%

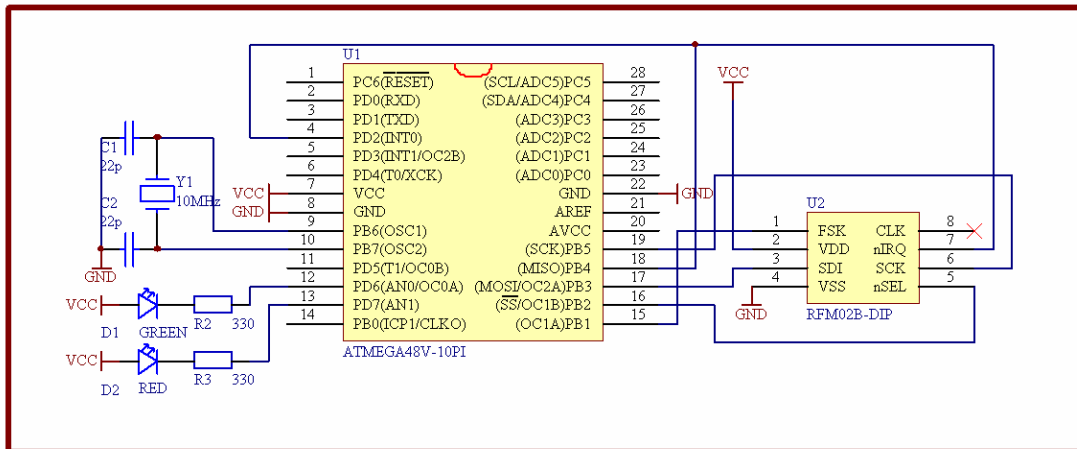
Bit 1 (dr): Disables the highly sensitive RESET mode. If this bit is cleared, a 600 mV glitch in the power supply may cause a system reset. Formore detailed description see the *Reset modes* section

3. Transmission Demo flow diagram



Note: After RF02 initialization, Open transmitter and use nIRQ as data rate clock. MCU write data bit on FSK pin at nIRQ falling edge.

4. Example 1(for AVR microcontroller)



/*****

copyright (c) 2006

Title: RF02B simple example based on AVR C
 Current version: v1.0
 Function: Package send Demo
 Processor: ATMEGA48
 Clock: 10MHz Crystal
 Operate frequency: 434MHz
 Data rate: 4.8kbps
 Package size: 23byte
 Author: Tank
 Company: Hope microelectronic Co.,Ltd.
 Contact: +86-0755-86106557
 E-MAIL: hopefsk@hoperf.com
 Date: 2006-10-24

Connections

ATMEGA48 SIDE	RF02B SIDE
SCK	→ SCK
MISO	: NC
MOSI	→ SDI
SS	→ nSEL
PB1	→ FSK
INT0	← nIRQ
PC0~PC3: LED0~LED3	

*****/

```
#include <mega48.h>

#define DDR_IN          0
#define DDR_OUT        1

#define PORT_SEL       PORTB
#define PIN_SEL        PINB
#define DDR_SEL        DDRB

#define PORT_SDI       PORTB
#define PIN_SDI        PINB
#define DDR_SDI        DDRB

#define PORT_SCK       PORTB
#define PIN_SCK        PINB
#define DDR_SCK        DDRB

#define PORT_SDO       PORTB
#define PIN_SDO        PINB
#define DDR_SDO        DDRB

#define PB7            7//--\
#define PB6            6// |
#define RFXX_SCK       5// |
#define RFXX_SDO       4// |RF_PORT
#define RFXX_SDI       3// |
#define RFXX_SEL       2// |
#define RFXX_DATA      1// |
#define PBO            0//--/

#define SEL_OUTPUT()   DDR_SEL |= (1<<RFXX_SEL)
#define HI_SEL()       PORT_SEL|= (1<<RFXX_SEL)
#define LOW_SEL()      PORT_SEL&=~(1<<RFXX_SEL)

#define SDI_OUTPUT()   DDR_SDI |= (1<<RFXX_SDI)
#define HI_SDI()       PORT_SDI|= (1<<RFXX_SDI)
#define LOW_SDI()      PORT_SDI&=~(1<<RFXX_SDI)

#define SDO_INPUT()    DDR_SDO&= ~(1<<RFXX_SDO)
#define SDO_HI()       PIN_SDO&(1<<RFXX_SDO)

#define SCK_OUTPUT()   DDR_SCK |= (1<<RFXX_SCK)
#define HI_SCK()       PORT_SCK|= (1<<RFXX_SCK)
#define LOW_SCK()      PORT_SCK&=~(1<<RFXX_SCK)
```

```
void RFX_PORT_INIT(void) {
    HI_SEL();
    HI_SDI();
    LOW_SCK();
    SEL_OUTPUT();
    SDI_OUTPUT();
    SDO_INPUT();
    SCK_OUTPUT();
}

unsigned int RFX_WRT_CMD(unsigned int aCmd) {
    unsigned char i;
    unsigned int temp;
    LOW_SCK();
    LOW_SEL();
    for(i=0;i<16;i++) {
        temp<<=1;
        if(SDO_HI()) {
            temp|=0x0001;
        }
        LOW_SCK();
        if(aCmd&0x8000) {
            HI_SDI();
        }else{
            LOW_SDI();
        }
        HI_SCK();
        aCmd<<=1;
    };
    LOW_SCK();
    HI_SEL();
    return(temp);
}
```

```
void RF02B_SEND(unsigned char aByte) {
    unsigned char i;

    for(i=0;i<8;i++) {
        while(PINB&(1<<RFX_SDO)); //Polling nIRQ
        while(!(PINB&(1<<RFX_SDO)));
        if(aByte&0x80) {
            PORTB|=(1<<RFX_DATA);
        }else{
            PORTB&=~(1<<RFX_DATA);
        }
    }
}
```

```
    }
    aByte<<=1;
  }

}

void main(void)
{
  unsigned int i, j, ChkSum;

  RFXX_PORT_INIT();

  RFXX_WRT_CMD(0xCC00);
  RFXX_WRT_CMD(0x8B61); //433BAND, +/-90kHz
  RFXX_WRT_CMD(0xA640); //434MHz
  RFXX_WRT_CMD(0xD040); //RATE/2
  RFXX_WRT_CMD(0xC823); //4.8kbps
  RFXX_WRT_CMD(0xC220); //ENABLE BIT SYNC
  RFXX_WRT_CMD(0xC001); //CLOSE ALL

  PORTB |= (1<<RFXX_DATA);
  DDRB |= (1<<RFXX_DATA); //SET DATA OUTPUT

  while(1) {
    RFXX_WRT_CMD(0xC039); //START TX
    ChkSum=0;
    RF02B_SEND(0xAA); //PREAMBLE
    RF02B_SEND(0xAA); //PREAMBLE
    RF02B_SEND(0xAA); //PREAMBLE
    RF02B_SEND(0x2D); //HEAD HI BYTE
    RF02B_SEND(0xD4); //HEAD LOW BYTE
    RF02B_SEND(0x30); //DATA0
    ChkSum+=0x30;
    RF02B_SEND(0x31); //DATA1
    ChkSum+=0x31;
    RF02B_SEND(0x32);
    ChkSum+=0x32;
    RF02B_SEND(0x33);
    ChkSum+=0x33;
    RF02B_SEND(0x34);
    ChkSum+=0x34;
    RF02B_SEND(0x35);
    ChkSum+=0x35;
    RF02B_SEND(0x36);
```

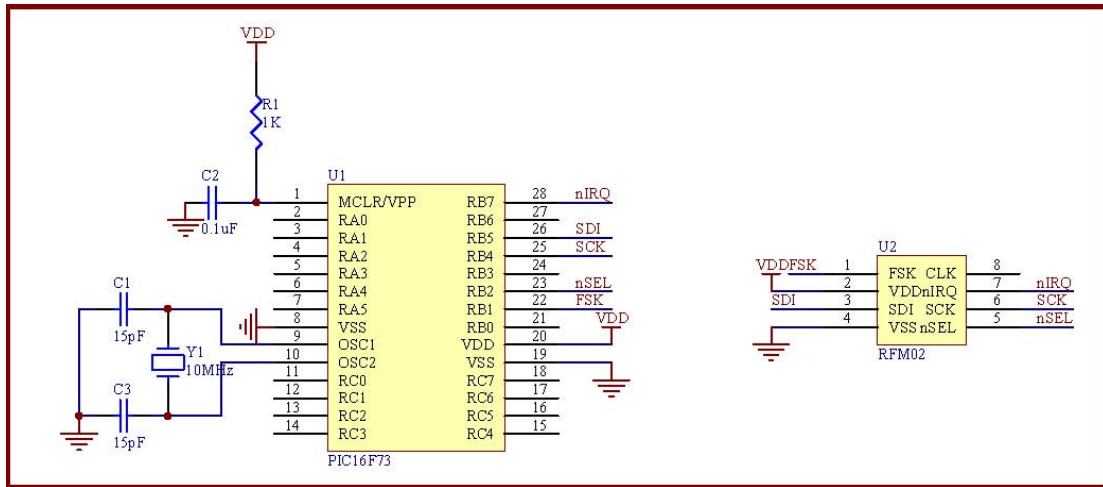
```
    ChkSum+=0x36;
    RF02B_SEND(0x37);
    ChkSum+=0x37;
    RF02B_SEND(0x38);
    ChkSum+=0x38;
    RF02B_SEND(0x39);
    ChkSum+=0x39;
    RF02B_SEND(0x3A);
    ChkSum+=0x3A;
    RF02B_SEND(0x3B);
    ChkSum+=0x3B;
    RF02B_SEND(0x3C);
    ChkSum+=0x3C;
    RF02B_SEND(0x3D);
    ChkSum+=0x3D;
    RF02B_SEND(0x3E);
    ChkSum+=0x3E;
    RF02B_SEND(0x3F); //DATA15
    ChkSum+=0x3F;
    RF02B_SEND(ChkSum); //DATA16
    RF02B_SEND(0xAA); //DUMMY BYTE

    RFXX_WRT_CMD(0xC001); //CLOSE TX

    for (i=0; i<5000; i++) for (j=0; j<123; j++);

};
}
```

5. Example 2(for PIC microcontroller)



/******

copyright (c) 2006

Title: RF02B simple example based on PIC C
 Current version: v1.0
 Function: Package send Demo
 Processor: PIC16F73
 Clock: 10MHz Crystal
 Operate frequency: 434MHz
 Data rate: 4.8kbps
 Package size: 23byte
 Author: Robben
 Company: Hope microelectronic Co.,Ltd.
 Contact: +86-0755-86106557
 E-MAIL: hopefsk@hoperf.com
 Date: 2006-11-10

*****/

#include "pic.h"

```
typedef unsigned char uchar;
typedef unsigned int uint;
```

```
#define SDI          RB5
#define SCK          RB4
#define nSEL         RB2
#define FSK          RB1
#define nIRQ         RB7
#define SDO          RB6
#define SDI_OUT()   TRISB5=0
#define SCK_OUT()   TRISB4=0
#define nSEL_OUT()  TRISB2=0
```

```
#define FSK_OUT()      TRISB1=0
#define nIRQ_IN()     TRISB7=1
#define SDO_IN()      TRISB6=1

void Write0( void );
void Writel( void );
void WriteCMD( uint CMD );
void RF2_Init( void );
void DelayUs( uint us );
void WriteFSKbyte( uchar DATA );
void DelayMs( uint ms );

__CONFIG(0x3FF2);
/*****
初始化端口
*****/
void RF2_Init( void )
{
    nSEL=1;
    SDI=1;
    SCK=0;
    FSK=0;
    nSEL_OUT();
    SDI_OUT();
    SDO_IN();
    SCK_OUT();
    FSK_OUT();
}
void main()
{
    uint ChkSum=0;

    RF2_Init();

    WriteCMD( 0xCC00 );
    WriteCMD( 0x8B61 );
    WriteCMD( 0xA640 );
    WriteCMD( 0xD040 );
    WriteCMD( 0xC823 );
    WriteCMD( 0xC220 );
    WriteCMD( 0xC001 );

    while(1)
    {
```

```
WriteCMD( 0xC039 );

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&=0xFF;
WriteFSKbyte( ChkSum );
WriteFSKbyte( 0xAA );
WriteCMD( 0xC001 );
```

```
    DelayMs( 1000 );
}
}
/*****
命令字写 0, 提供时序
*****/
void Write0( void )
{
    SCK=0;
    NOP();
    SDI=0;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    SCK=1;
    NOP();
}
/*****
命令字写 1, 提供时序
*****/
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();
SCK=1;
NOP();
}
/*****
写一个字节发送数据
*****/
void WriteFSKbyte( uchar DATA )
{
    uchar n=8;
    nSEL=1;
    while(n-->0)
    {
        while(!nIRQ);
        while(nIRQ);
        if(DATA&0x80)
            FSK=1;
        else
            FSK=0;
        DATA=DATA<<1;
    }
}
/*****
写一条命令字
*****/
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;
}
/*****
延时
*****/
void DelayUs( uint us )
{
    uint i;
    while( us-- )
    {
        i=2;
        while( i-- )
        {
            NOP();
        }
    }
}
/*****
延时
*****/
void DelayMs(uint ms)
{
    uchar i;
    while(ms--)
    {
        i=35;
        while(i--)
        {
            DelayUs(1);
        }
    }
}
```

Address: Rm B.8/F LiJingGe Emperor
Regency 6012 ShenNan Rd, Shenzhen, China
Tel: 86-755-82973805
Fax: 86-755-82973550
Email: sales@hoperf.com
trade@hoperf.com
Website: <http://www.hoperf.com>
<http://www.hoperf.cn>
<http://hoperf.en.alibaba.com>

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 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.

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