

RF01 programming guide

1. Brief description

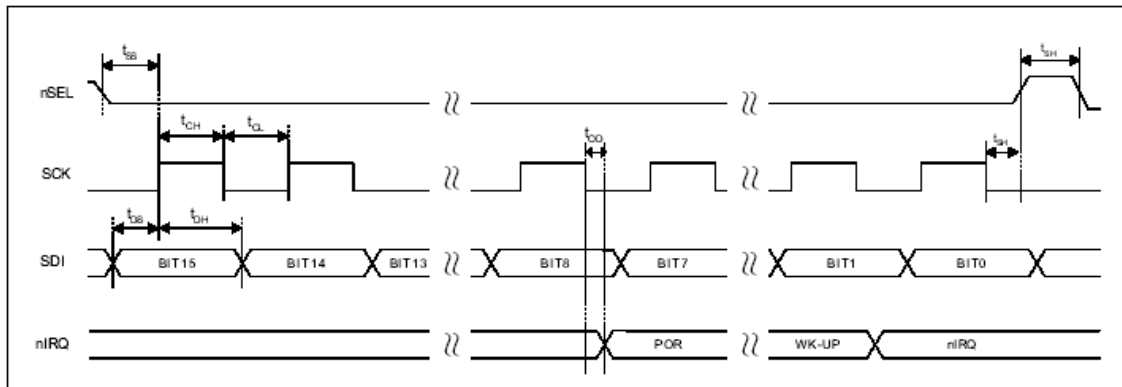
RF01 is a low cost FSK receive IC with integrated all RF functions in a single chip. It only need a MCU, a crystal, a decouple capacitor and antenna to build a hi reliable FSK receiver. The operation frequency can cover 300 to 1000MHz.

RF01 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

RF01 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	eb	et	ex	x3	x2	x1	x0	i2	i1	i0	dc	893Ah

b1..b0: select band

b1	b0	band[MHz]
0	0	315
0	1	433
1	0	868
1	1	915

eb: Enable low battery detection function

et: Enable wake-up timer

ex: Enable crystal oscillator

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

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

dc: Disable signal output of CLK pin

3. 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

315band: $F_c = 310 + F * 0.0025$ MHz

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, F is frequency parameter and $36 \leq F \leq 3903$

4. Receiver Setting 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	d1	d0	g1	g0	r2	r1	r0	en	C0C1h

d1..d0: select VDI source

d1	d0	VDI output
0	0	Digital RSSI output(DRSSI)
0	1	Data quality detection output (DQD)
1	0	Clock recovery lock output
1	1	Always on

g1..g0: select LNA gain

g1	g0	LNA gain (dBm)
0	0	0
0	1	-14
1	0	-6
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	-67
1	0	1	-61

The actual DRSSI threshold is related to LNA setup:

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

en: Enable the receiver

5. 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:

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

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

6. 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	1	0	0	d6	d5	d4	d3	d2	d1	d0	en	CC0Eh

d6..d0: Set duty cycle

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

en: Enable low duty cycle mode

7. 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	1	0	d2	d1	d0	t4	t3	t2	t1	t0	C200h

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

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

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

8. 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	1	0	a1	a0	r1	r0	st	fi	oe	en	C6F7h

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

r1..r0: 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

9. 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	1	0	0	al	ml	1	s1	s0	f2	f1	f0	C42Ch

al: Enable clock recovery auto-lock

ml: Enable clock recovery fast mode

s1..s0: select data filter type

s1	s0	Filter type
0	0	OOK
0	1	Digital filter
1	0	reserved

f1..f0: Set DQD threshold

10. 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	cs	r6	r5	r4	r3	r2	r1	r0	C823h

r7..r0: Set data rate

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

11. Output and FIFO 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	1	1	0	f3	f2	f1	f0	s1	s0	ff	fe	CE85h

f3..f0: Set FIFO interrupt level

s1..s0: select FIFO fill start condition

s1	s0	
0	0	VDI
0	1	Sync-word
1	0	Reserved
1	1	Always

ff: Enable FIFO fill

fe: Enable FIFO function

12.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	1	0	1	0	0	0	0	0	0	0	0	dr	DAOOh

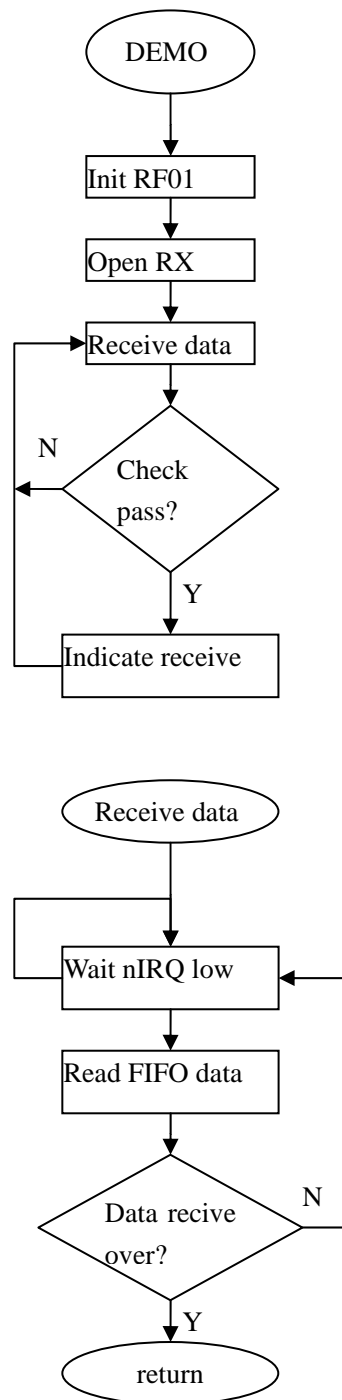
Bit 0 (*dr*): Disables the highly sensitive RESET mode. If this bit is cleared, a 600 mV glitch in the power supply may cause a system reset. Formore detailed description see the *Reset modes* section.

13.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	-

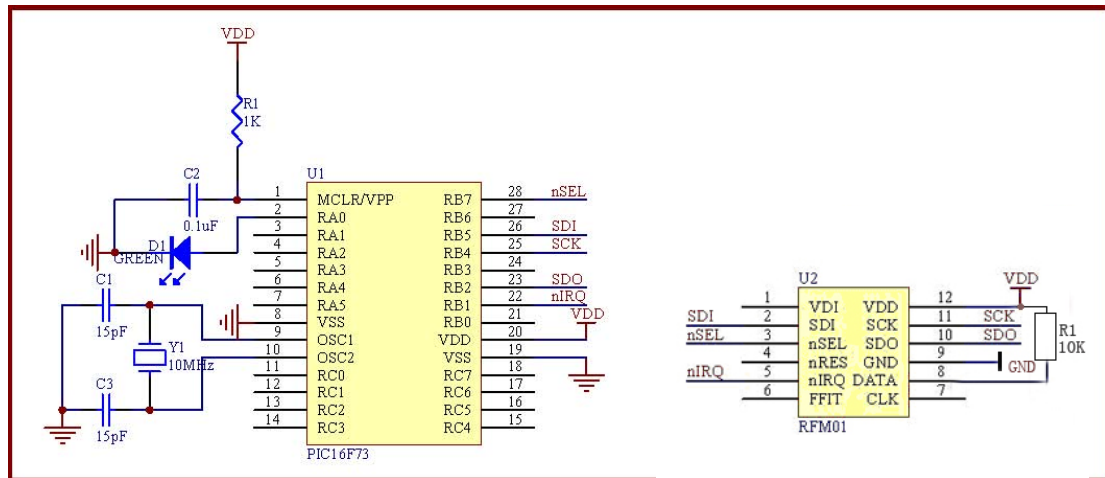
This command starts with a 0 and be used to read internal status register

3. Demo flow diagram



Note: After RF01 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. Example 1 (for PIC microcontroller)



/*****

copyright (c) 2010

Title: RF01 simple example based on PIC C
 Current version: V1.2
 Function: Package receive Demo
 Processor: PIC16F73
 Clock: 10MHz Crystal
 Operate frequency: 434MHz
 Data rate: 4.8kbps
 Package size: 23byte
 Author: BaiGuang
 Company: Hope microelectronic Co.,Ltd.
 Contact: +86-0755-82973806
 E-MAIL: rfeng@hoperf.com
 Date: 2010-07-14

*****/

```
#include "pic.h"
typedef unsigned char uchar;
typedef unsigned int uint;

#define SDI RB5
#define SCK RB4
#define SDO RB2
#define nIRQ RB1
#define nSEL RB7
#define LED RA0
#define LED_OUT() TRISA0=0
#define nIRQ_IN() TRISB1=1
#define SDI_OUT() TRISB5=0
#define SCK_OUT() TRISB4=0
```


HOPE RF

RF01 V1.2

```
#define SDO_IN()      TRISB2=1
#define DATA_IN()   TRISB1=1
#define nSEL_OUT()   TRISB7=0
```

```
void RF1_Init( void );
void Write0( void );
void Write1( void );
void Delayus( uint us );
void WriteCMD( uint CMD );
uchar RF01_RDFIFO(void);
void Delays(void);
__CONFIG(0x3FF2);
bank1 uchar RF_RXBUF[19];
```

```
void RF1_Init( void )
{
    nSEL=1;
    SDI=1;
    SCK=0;
    nSEL_OUT();
    SDI_OUT();
    SDO_IN();
    nIRQ_IN();
    SCK_OUT();
    LED_OUT();
    LED=0;
}
```

```
void main()
{
    uchar i=0, j=0;
    uint CheckSum;
    Delays();
    RF1_Init();
    WriteCMD(0x0000);
    WriteCMD(0x898A); //433BAND, 134kHz
    WriteCMD(0xA640); //434MHz
    WriteCMD(0xC847); //4.8kbps
    WriteCMD(0xC69B); //AFC setting
    WriteCMD(0xC42A); //Clock recovery manual control, Digital filter, DQD=4
    WriteCMD(0xC240); //output 1.66MHz
    WriteCMD(0xC080);
    WriteCMD(0xCE84); //use FIFO
    WriteCMD(0xCE87);
```

```
WriteCMD(0xC081); //OPEN RX
while(1)
{
    while(!nIRQ)
    {
        RF_RXBUF[i++]=RF01_RDFIFO();
        if(i==17)
        {
            i=0;
            WriteCMD(0xCE48);
            WriteCMD(0xCE87);
            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 )
{
    SDI=0;
    SCK=0;
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
}
```

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

void Writel( void )
{
    SDI=1;
    SCK=0;
    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)
            Writel();
        else
            Write0();
        CMD=CMD<<1;
    }
    SCK=0;
}
```

```
nSEL=1;
}

uchar RF01_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();  
        }  
    }  
}
```

```
void Delays(void)  
{  
    uchar i=10;  
    while(i--)  
    {  
        Delayus(1);  
    }  
}
```

Note:

- (1) Using MCU taking the module to sleeping module:

The module can be get into the sleeping module by following statement.

```
WriteCMD(0xc000); //close the receiver module  
WriteCMD(0x888b); //433 BAND, 134Khz, close Clk pin output, in the sleep state close  
the crystal.
```

- (2) Module automatically sleep and wake up, using MCU to set the wake_up time and working time.

Add the following statement in the initialization:

```
WriteCMD(0xc000); //close receiver  
WriteCMD(0xe4fa); //sleep time 4000ms, M=250, R=4 set the sleep time  
WriteCMD(0x8a8b); //433 BAND, 134Khz, clk pin output disable, open wake_up time. into  
sleep mode, open regular wake_up interrupt.
```

MCU awakened by the nIRQ pin switch that trigger by the wake_up interrupt. Open the module to receives the data. The wake_up time is custom by the customer. If want the module re_enter the sleeping state, Do following operation.

```
WriteCMD(0x0000); //clear interrupt  
WriteCMD(0x898b); //433 BAND, 134Khz, clk pin output disable, close wake_up time.  
WriteCMD(0x8a8b); //433 BAND, 134Khz, clk pin output disable, open wake_up time.
```

Sleep current is less 1UA.

<p>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</p>	<p>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 MECHANABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.</p> <p>©2006, HOPE MICROELECTRONICS CO.,LTD. All rights reserved.</p>
--	--