EN: This Datasheet/Document is presented by the manufacturer.

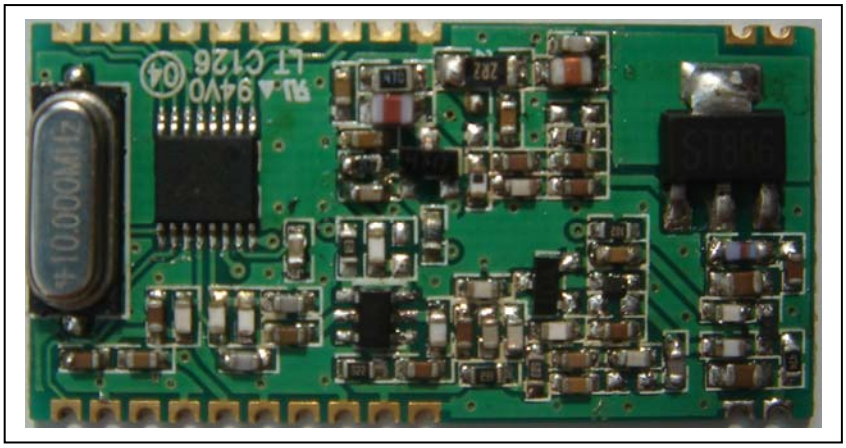
Please visit our website for pricing and availability at www.hestore.hu.

UNIVERSAL ISM BAND FSK TRANSCEIVER MODULE

WITH 500mW OUTPUT POWER

RFM12BP

（the purpose of this spec covers mainly for the physical characteristic of the module, for register configure and its related command info please refer to RF12B datasheet）



General Introduction

RFM12BP is a low costing ISM band transceiver module implemented with 500mW output power. It works signal ranges from 433/868/915MHZ bands, The SPI interface is used to communicate with microcontroller for parameter setting.

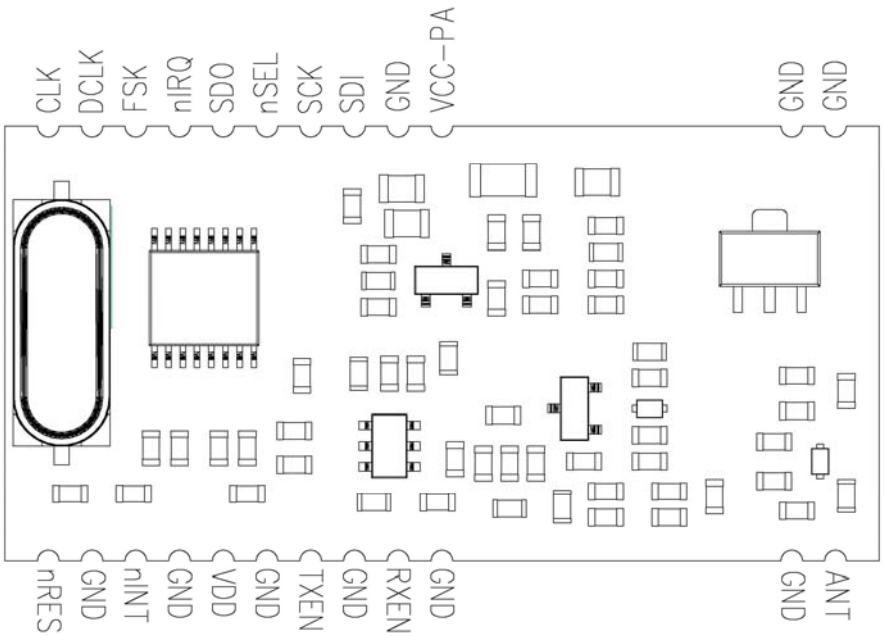
Features:

- High output power with 500mW
- High input sensitivity with -116dBm
- Low costing, high performance and price ratio
- PLL and zero IF technology
- Fast PLL lock time
- High resolution PLL with 2.5 KHz step
- High data rate (up to 115.2 kbps with internal demodulator，with external RC filter highest data rate is 256 kbps)
- 50 OHM antenna input/output
- Programmable TX power
- Programmable TX frequency deviation (from 15 to 240 KHz)
- Programmable receiver bandwidth (from 67 to 400 kHz)
- Analog and digital signal strength indicator (ARSSI/DRSSI)
- Automatic frequency control (AFC)
- Data quality detection (DQD)
- Internal data filtering and clock recovery
- RX synchron pattern recognition
- SPI compatible serial control interface
- Clock and reset signal output for external MCU use
- 16 bit RX Data FIFO
- Two 8 bit TX data registers
- Standard 10 MHz crystal reference
- Wakeup timer
- 2.2V – 3.8V power supply for FSK IC, 12V power supply for power amplifier
- Standby current less than 0.3uA
- Supports very short packets (down to 3 bytes)

Typical Application：

- Remote control
- Remote sensor
- Wireless data collection
- Home security system
- Toys

Pin Definition：



PIN number	definition	Type	Function
1	GND	S	ground
2	GND	S	ground
3	VCC-PA	S	Positive power supply for power amplifier (12V)
4	GND	S	ground
5	SDI	DI	SPI data input
6	SCK	DI	SPI clock input
7	nSEL	DI	Chip select (active low)
8	SDO	DO	Serial data output with bus hold
9	nIRQ	DO	Interrupts request output（active low）
10	FSK/DATA/nFFS	DI/DO/DI	Transmit FSK data input/ Received data output (FIFO not used)/ FIFO select
11	DCLK/CFIL/FFIT	DO/AIO/DO	Clock output (no FIFO)/ external filter capacitor(analog mode)/ FIFO interrupts(active high)when FIFO level set to 1, FIFO empty interruption can be achieved
12	CLK	DO	Clock output for external microcontroller
13	nRES	DIO	Reset output（active low）
14	GND	S	ground
15	nINT/VDI	DI/ DO	Interrupt input (active low)/Valid data indicator

16	GND	S	ground
17	VDD	S	Positive power supply for FSK IC(2.2V-3.8V)
18	GND	S	ground
19	TXEN	AI	TX enable(active high)
20	GND	S	ground
21	RXEN	AI	RX enable(active high)
22	GND	S	ground
23	GND	S	ground
24	ANT	AIO	RF signal output/input (50 OHM)

Electrical Parameter：

Maximum（not at working mode）

symbol	parameter	minimum	maximum	Unit
VDD	Positive power supply for FSk IC	-0.5	4.0	V
VCC-PA	Positive power supply for power amplifier	-0.5	13	V
V_{in}	All pin input level except VDD,VCC-PA,GND,ANT	-0.5	VDD+0.5	V
I_{in}	Input current except VDD,VCC-PA,GND,ANT	-25	25	mA
ESD	Human body model		1000	V
T_{st}	Storage temperature	-55	125	℃
T_{ld}	Soldering temperature(10s)		260	℃

Recommended working range

symbol	parameter	min	max	Unit
VDD	Positive power supply	2.2	3.8	V
T_{op}	Working temperature	-40	+85	℃

DC characteristic

symbol	parameter	Remark	min	typ	max	Unit
$I_{dd_TX_PMAX}$	Supply current (TX mode, P_{out} = 500mW)	433MHz band 868/915MHz band		185 200		mA
I_{dd_RX}	Supply current (RX mode)	433MHz band 868/915MHz band		20 25		mA
I_x	Idle current	Crystal oscillator on		0.62	1.2	mA
I_{pd}	Sleep mode current	All blocks off		0.3		uA
I_{lb}	Low battery detection			0.5		uA
V_{lb}	Low battery detect threshold	0.1V per step	2.2		3.8	V
V_{lba}	Low battery detection accuracy		0		5	%
V_{il}	Low level input				$0.3*V_{dd}$	V
V_{ih}	High level input		$0.7*V_{dd}$			V
I_{il}	Leakage current	V_{il}=0V	-1		1	uA

I_{ih}	Leakage current	$V_{ih}=V_{dd}$, $V_{dd}=4V$	-1		1	uA
V_{ol}	Low level output	$I_{ol}=2mA$			0.4	V
V_{oh}	High level output	$I_{oh}=-2mA$	V_{dd}-0.4			V

AC characteristic

symbol	parameter	remark	min	typical	max	Unit
f_{ref}	PLL frequency		9	10	11	MHz
f_{LO}	frequency (10MHz crystal used)	433 MHz band,2.5KHz step 868 MHz band,5KHz step 915 MHz band,7.5KHz step	430.24 860.48 900.72		439.75 879.51 929.27	MHz
f_{LO}	frequency (9MHZ crystal used)	433 MHz band,2.5KHz step 868 MHz band,5KHz step 915 MHz band,7.5KHz step	387.22 774.43 810.65		395.76 791.56 836.34	MHz
f_{LO}	frequency (11MHZ crystal used)	433 MHz band,2.5KHz step 868 MHz band,5KHz step 915 MHz band,7.5KHz step	473.26 946.53 990.79		483.73 967.46 1022.2	MHz
BW	Receiver bandwidth	mode 0 mode 1 mode 2 mode 3 mode 4 mode 5	60 120 180 240 300 360	67 134 200 270 350 400	75 150 225 300 375 450	KHz
t_{lock}	PLL lock time	After 10MHz step hopping, frequency error <10 kHz		30		us
tst, P	PLL startup time	With a running crystal oscillator		200	300	us
BR	Data rate	With internal digital demodulator	0.6		115.2	kbps
BR_A	Data rate	With external RC filter			256	kbps
P_{min}	sensitivity	BER 10^{-3}, BW=134KHz,BR=1.2kbps		-117	-115	dBm
AFC_{range}	AFC working range	df_{FSK} : FSK deviation in the received signal		0.8* df_{FSK}		

AC characteristic(Transmitter)

symbol	parameter	remark	min	typical	max	Unit
P_{max_50}	Max. output power delivered to 50Ohm load	433MHZ band 868/915MHZ band		500 500		mW
BR_{TX}	FSK bit rate	Via internal TX data register			172	kbps

BRA_{TX}	FSK bit rate	TX data connected to the FSK input			256	kbps
df_{fsk}	FSK frequency deviation	Programmable in 15 kHz steps	15		240	kHZ

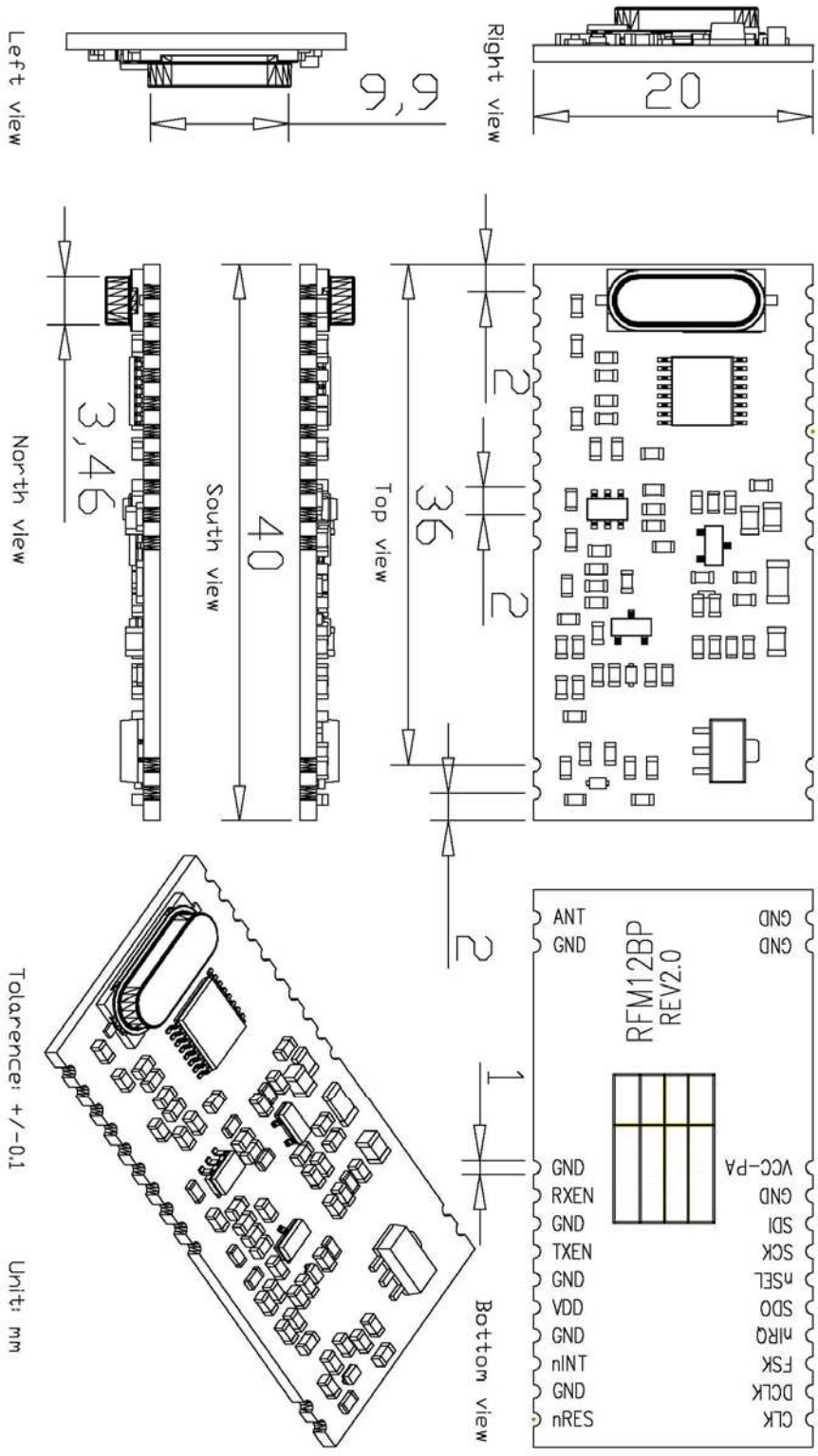
AC characteristic(Turn-on/Turnaround timings)

symbol	parameter	remark	min	typical	max	Unit
T_{st}	Crystal oscillator startup time	Crystal ESR < 100		1	5	ms
$T_{tx_XTAL_ON}$	Transmitter turn-on time	Synthesizer off, crystal oscillator on with 10 MHz step		250		us
$T_{rx_XTAL_ON}$	Receiver turn-on time	Synthesizer off, crystal oscillator on with 10 MHz step		250		us
$T_{tx_rx_SYNT_ON}$	Transmitter – Receiver turnover time	Synthesizer and crystal oscillator on during TX/RX change with 10 MHz step		150		us
$T_{rx_tx_SYNT_ON}$	Receiver – Transmitter turnover time	Synthesizer and crystal oscillator on during RX/TX change with 10 MHz step		150		us
C_{xl}	Crystal load capacitance	Programmable in 0.5 pF steps, tolerance+/- 10%	8.5		16	pf
t_{POR}	Internal POR timeout	After V_{dd} has reached 90% of final value			100	ms
t_{PBt}	Wake-up timer clock period	Calibrated every 30 seconds	0.96		1.05	ms
$C_{in,\ D}$	Digital input capacitance				2	pf
$t_{r,\ f}$	Digital output rise/fall time	15pF pure capacitive load			10	ns

Field testing range

Band	Test condition	Distance
433MHz band	Receiver bandwidth =67KHz, data rate=1.2kbps, transmitter frequency deviation =45KHZ（matches with RFM12）In free open area	>3000M
868MHz band	Receiver bandwidth=67KHz,data rate =1.2kbps,Transmitter frequency deviation =45KHZ（matches with RFM12）in free open area	>3000M
915MHz band	Receiver bandwidth=67KHz,data rate =1.2kbps,Transmitter frequency deviation =45KHZ（matches with RFM12）in free open area	>3000M

Mechanical Dimension



Module Model Definition

model=module-operation band

RFM12BP – 433

module type operation band

example：1，RFM12BP module at 433MHz band, RFM12BP-433.

2，RFM12BP module at 868MHZ band, RFM12BP-868.

HOPE MICROELECTRONICS CO.,LTD
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

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 MECHANTABILITY OR FITNESS FOR A ARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.

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

RF12 programming guide

1. Brief description

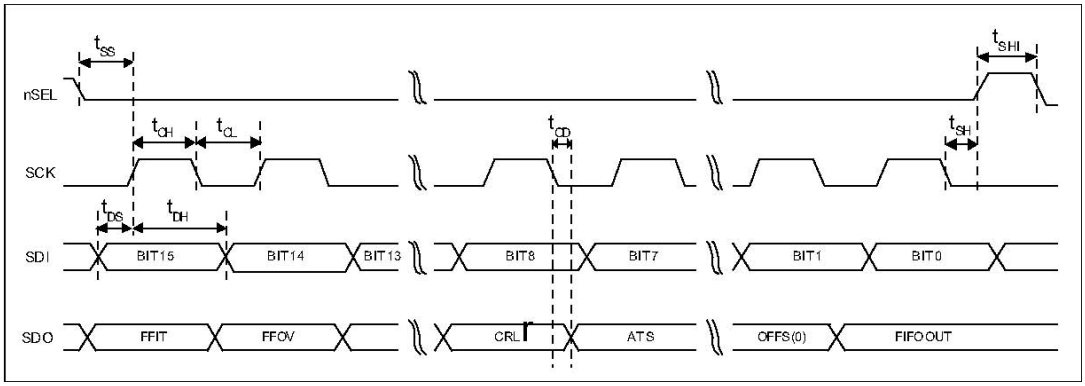
RF12B is a low cost FSK transceiver IC witch integrated all RF functions in a single chip. It only need a MCU, a crystal, a decouple capacitor and antenna to build a hi reliable FSK transceiver system. The operation frequency can cover 400 to 1000MHz.

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	el	ef	b1	b0	x3	x2	x1	x0	8008h

e l: 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: Fc=430+F*0.0025 MHz
868band: Fc=860+F*0.0050 MHz
915band: Fc=900+F*0.0075 MHz
Fc is carrier frequency and F is the frequency parameter. 36≤F≤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	rl1	rl0	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

rl1..rl0: 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 funcition

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	lpx	ddy	ddit	1	bw0	CC67h

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

lpx: select low power mode of the crystal oscillator.

lpx	Crystal start-up time (typ)	Power consumption (typ)
0	1 ms	620 uA
1	2 ms	460 uA

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:

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

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

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	C8OEh

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：

Vlb=2.2+V*0.1 [V]

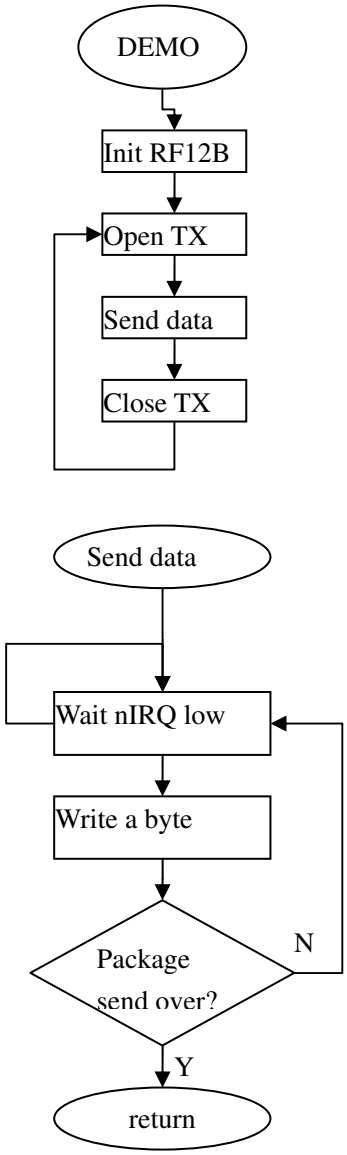
18. Status Read Command

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

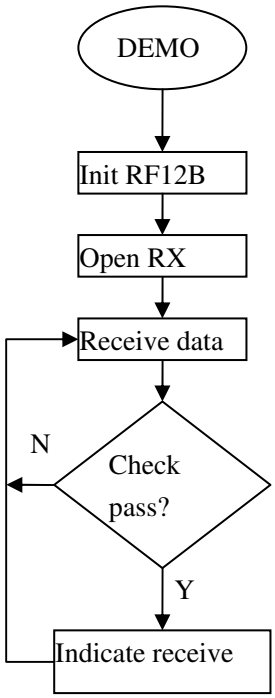
3. Demo flow diagram

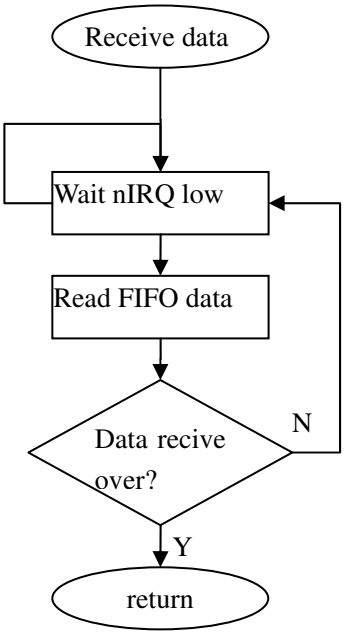
Transmitter:



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:

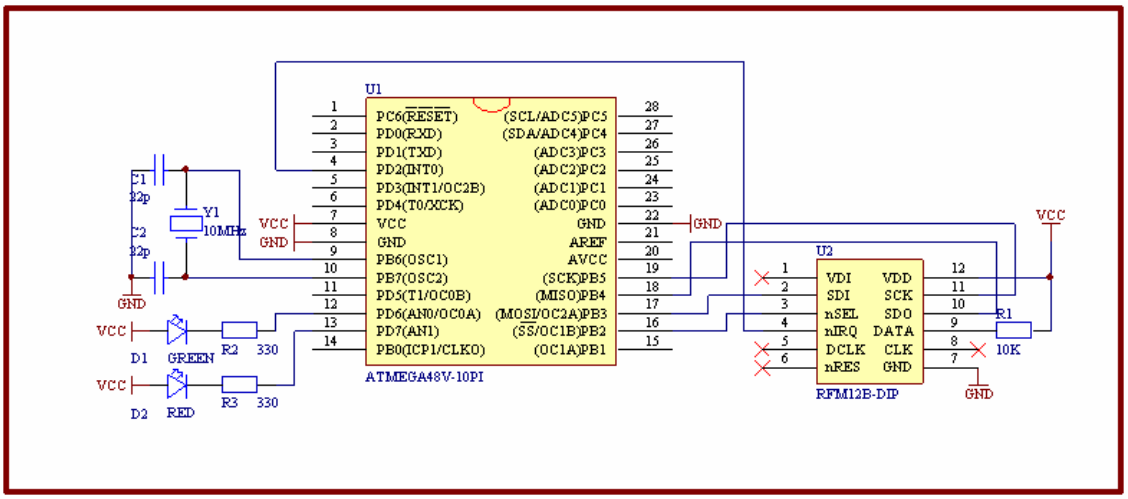




Note: After RF12B initialization, Open FIFO receive mode and wait nIRQ low, only then MCU

can read received and stored in FIFO data. For next package receive, please reset FIFO.

4. Example 1（for AVR microcontroller）



RF12 transmitter demo:

```
/*
                ;   copyright (c) 2006
                ;Title              RF12 TX simple example based on AVR C
                ;Company:           Hope microelectronic Co.,Ltd.
                ;Author:            Tank
                ;Current version:   v1.0
                ;Date:              2006-11-13
                ;processor          ATMEGA48
                ;Clock:             10MHz Crystal
                ;Contact:           +86-0755-86106557
                ;E-MAIL:            hopefsk@hoperf.com


                Connections

      ATMEGA48 SIDE       RF12 SIDE
        SCK---------------->SCK
       MISO---------------->SDO
       MOSI---------------->SDI
         SS---------------->nSEL
       INT0<----------------nIRQ

       PD6: LED GREEN
       PD7: LED RED
```

```
*/


#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 PORT_DATA       PORTD
#define PIN_DATA        PIND
#define DDR_DATA        DDRD

#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 NC              1//  |
#define PB0             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)

#define RF12_DATA        4//PD4
#define DATA_OUT         DDR_DATA|=1<<RF12_DATA
#define HI_DATA          PORT_DATA|=1<<RF12_DATA

#define LEDG_OUTPUT()    DDRD|=~(1<<6)
#define LEDR_OUTPUT()    DDRD|=~(1<<7)

#define LEDG_ON()        PORTD&=~(1<<6)
#define LEDG_OFF()       PORTD|=~(1<<6)
#define LEDR_ON()        PORTD&=~(1<<7)
#define LEDR_OFF()       PORTD|=~(1<<7)


void RFXX_PORT_INIT(void){
  HI_SEL();
  HI_SDI();
  LOW_SCK();
  SEL_OUTPUT();
  SDI_OUTPUT();
  SDO_INPUT();
  SCK_OUTPUT();
}
unsigned int RFXX_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 RF12_INIT(void){
  RFXX_WRT_CMD(0x80D7);//EL,EF,433band,12.0pF
  RFXX_WRT_CMD(0x8239);//!er,!ebb,ET,ES,EX,!eb,!ew,DC
  RFXX_WRT_CMD(0xA640);//434MHz
  RFXX_WRT_CMD(0xC647);//4.8kbps
  RFXX_WRT_CMD(0x94A0);//VDI,FAST,134kHz,0dBm,-103dBm
  RFXX_WRT_CMD(0xC2AC);//AL,!ml,DIG,DQD4
  RFXX_WRT_CMD(0xCA81);//FIFO8,SYNC,!ff,DR
  RFXX_WRT_CMD(0xCED4);//SYNC=2DD4;
  RFXX_WRT_CMD(0xC483);//@PWR,NO RSTRIC,!st,!fi,OE,EN
  RFXX_WRT_CMD(0x9850);//!mp,90kHz,MAX OUT
  RFXX_WRT_CMD(0xCC77);//OB1，OB0，  LPX,！ddy，DDIT，BW0
  RFXX_WRT_CMD(0xE000);//NOT USE
  RFXX_WRT_CMD(0xC800);//NOT USE
  RFXX_WRT_CMD(0xC040);//1.66MHz,2.2V
}
void RF12_SEND(unsigned char aByte){
  while(PIND&(1<<2));//wait for previously TX over
  RFXX_WRT_CMD(0xB800+aByte);
}

void Delay_ms(unsigned char amS){
  unsigned char i;
  unsigned int j;
  for(i=0;i<amS;i++)for(j=0;j<914;j++);
}
void main(void)
{
  unsigned int i,j;
  unsigned char ChkSum;

  #asm("cli");
  DDRB=0x00;//PB INPUT;
```

```
DDRD=0x00;//PD INPUT;

//POWER ON indication: both LEDR and LEDG blink 3 times
LEDG_OFF();
LEDR_OFF();
LEDG_OUTPUT();
LEDR_OUTPUT();
for(i=0;i<3;i++){
  Delay_ms(200);
  LEDG_ON();
  LEDR_ON();
  Delay_ms(200);
  LEDG_OFF();
  LEDR_OFF();
}

  LEDG_OFF();
  LEDR_OFF();

RFXX_PORT_INIT();

RF12_INIT();

DDRD|=(1<<RF12_DATA);
PORTD|=(1<<RF12_DATA);// SET nFFS pin HI when using TX register
DDRD&=~(1<<2);        //PD2(INT0)


while(1){
  LEDR_ON();
  RFXX_WRT_CMD(0x0000);//read status register
  RFXX_WRT_CMD(0x8239);//!er,!ebb,ET,ES,EX,!eb,!ew,DC

  ChkSum=0;
  RF12_SEND(0xAA);//PREAMBLE
  RF12_SEND(0xAA);//PREAMBLE
  RF12_SEND(0xAA);//PREAMBLE
  RF12_SEND(0x2D);//SYNC HI BYTE
  RF12_SEND(0xD4);//SYNC LOW BYTE
  RF12_SEND(0x30);//DATA BYTE 0
  ChkSum+=0x30;
  RF12_SEND(0x31);//DATA BYTE 1
  ChkSum+=0x31;
  RF12_SEND(0x32);
```

```
    ChkSum+=0x32;
    RF12_SEND(0x33);
    ChkSum+=0x33;
    RF12_SEND(0x34);
    ChkSum+=0x34;
    RF12_SEND(0x35);
    ChkSum+=0x35;
    RF12_SEND(0x36);
    ChkSum+=0x36;
    RF12_SEND(0x37);
    ChkSum+=0x37;
    RF12_SEND(0x38);
    ChkSum+=0x38;
    RF12_SEND(0x39);
    ChkSum+=0x39;
    RF12_SEND(0x3A);
    ChkSum+=0x3A;
    RF12_SEND(0x3B);
    ChkSum+=0x3B;
    RF12_SEND(0x3C);
    ChkSum+=0x3C;
    RF12_SEND(0x3D);
    ChkSum+=0x3D;
    RF12_SEND(0x3E);
    ChkSum+=0x3E;
    RF12_SEND(0x3F);    //DATA BYTE 15
    ChkSum+=0x3F;
    RF12_SEND(ChkSum);  //send chek sum
    RF12_SEND(0xAA);//DUMMY BYTE
    RF12_SEND(0xAA);//DUMMY BYTE
    RF12_SEND(0xAA);//DUMMY BYTE

    RFXX_WRT_CMD(0x8201);
    LEDR_OFF();
    LEDG_OFF();
    for(i=0;i<10000;i++)for(j=0;j<123;j++);//sleep 1 second appr.

  };
}
```

RF12 receiver demo

```
/*
```

```
;   copyright (c) 2006
;Title              RF12 RX simple example based on AVR C
;Company:           Hope microelectronic Co.,Ltd.
;Author:            Tank
;Current version:   v1.0
;Date:              2006-11-17
;processor          ATMEGA48
;Clock:             10MHz Crystal
;Contact:           +86-0755-86106557
;E-MAIL:            hopefsk@hoperf.com


                Connections

      ATMEGA48 SIDE       RF12 SIDE
        SCK---------------->SCK
       MISO<----------------SDO
       MOSI---------------->SDI
         SS---------------->nSEL
        PD4---------------->FSK/DATA
       INT0<----------------nIRQ

       PD6: LED GREEN
       PD7: LED RED

*/

#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 PORT_IRQ        PORTD
#define PIN_IRQ         PIND
#define DDR_IRQ         DDRD

#define PORT_DATA       PORTD
#define PIN_DATA        PIND
#define DDR_DATA        DDRD

#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 NC              1//  |
#define PB0             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 LOW_SDO()       PORT_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)

#define RF12_IRQ        2
#define IRQ_IN()        DDR_IRQ &=~(1<<RF12_IRQ)
#define WAIT_IRQ_LOW()  while(PIND&(1<<RF12_IRQ))

#define RF12_DATA       4//PD4
#define DATA_OUT()      DDR_DATA|=1<<RF12_DATA
```

```
#define HI_DATA()       PORT_DATA|=1<<RF12_DATA

#define LEDG_OUTPUT()   DDRD|=~(1<<6)
#define LEDR_OUTPUT()   DDRD|=~(1<<7)

#define LEDG_ON()       PORTD&=~(1<<6)
#define LEDG_OFF()      PORTD|= (1<<6)
#define LEDR_ON()       PORTD&=~(1<<7)
#define LEDR_OFF()      PORTD|= (1<<7)


void RFXX_PORT_INIT(void){
  HI_SEL();
  HI_SDI();
  LOW_SCK();
  //SET nFFS pin HI when using FIFO
  HI_DATA();
  SEL_OUTPUT();
  SDI_OUTPUT();
  SDO_INPUT();
  SCK_OUTPUT();
  IRQ_IN();
  DATA_OUT();
}
unsigned int RFXX_WRT_CMD(unsigned int aCmd){
  unsigned char i;
  unsigned int temp;
  temp=0;
  LOW_SCK();
  LOW_SEL();
  for(i=0;i<16;i++){
    if(aCmd&0x8000){
      HI_SDI();
    }else{
      LOW_SDI();
    }
    HI_SCK();
    temp<<=1;
    if(SDO_HI()){
      temp|=0x0001;
    }
    LOW_SCK();

    aCmd<<=1;
```

```
  };
  HI_SEL();
  return(temp);
}
void RF12_INIT(void){
  RFXX_WRT_CMD(0x80D7);//EL,EF,433band,12.0pF
  RFXX_WRT_CMD(0x8239);//!er,!ebb,ET,ES,EX,!eb,!ew,DC
  RFXX_WRT_CMD(0xA640);//434MHz
  RFXX_WRT_CMD(0xC647);//4.8kbps
  RFXX_WRT_CMD(0x94A0);//VDI,FAST,134kHz,0dBm,-103dBm
  RFXX_WRT_CMD(0xC2AC);//AL,!ml,DIG,DQD4
  RFXX_WRT_CMD(0xCA81);//FIFO8,SYNC,!ff,DR
  RFXX_WRT_CMD(0xCED4);//SYNC=2DD4;
  RFXX_WRT_CMD(0xC483);//@PWR,NO RSTRIC,!st,!fi,OE,EN
  RFXX_WRT_CMD(0x9850);//!mp,90kHz,MAX OUT
  RFXX_WRT_CMD(0xCC77);//OB1，OB0,  LPX,！ddy，DDIT，BW0
  RFXX_WRT_CMD(0xE000);//NOT USE
  RFXX_WRT_CMD(0xC800);//NOT USE
  RFXX_WRT_CMD(0xC040);//1.66MHz,2.2V
}
unsigned char RF12_RECV(void){
  unsigned int FIFO_data;
  WAIT_IRQ_LOW();
  RFXX_WRT_CMD(0x0000);
  FIFO_data=RFXX_WRT_CMD(0xB000);
  return(FIFO_data&0x00FF);
}
void Delay_ms(unsigned char amS){
  unsigned char i;
  unsigned int j;
  for(i=0;i<amS;i++)for(j=0;j<914;j++);
}
void main(void)
{
  unsigned char i;
  unsigned char ChkSum;

  //POWER ON indication: both LEDR and LEDG blink 3 times

  LEDG_OFF();
  LEDR_OFF();
  LEDG_OUTPUT();
  LEDR_OUTPUT();
```

```
for(i=0;i<3;i++){
  Delay_ms(200);
  LEDG_ON();
  LEDR_ON();
  Delay_ms(200);
  LEDG_OFF();
  LEDR_OFF();
}

  LEDG_OFF();
  LEDR_OFF();

//Initialize command port
RFXX_PORT_INIT();

//Initialize RF12 chip
RF12_INIT();

//Init FIFO
RFXX_WRT_CMD(0xCA81);

while(1){
  //Enable FIFO
  RFXX_WRT_CMD(0xCA83);
  ChkSum=0;

  //Receive payload data
  for(i=0;i<16;i++){
    ChkSum+=RF12_RECV();
  }

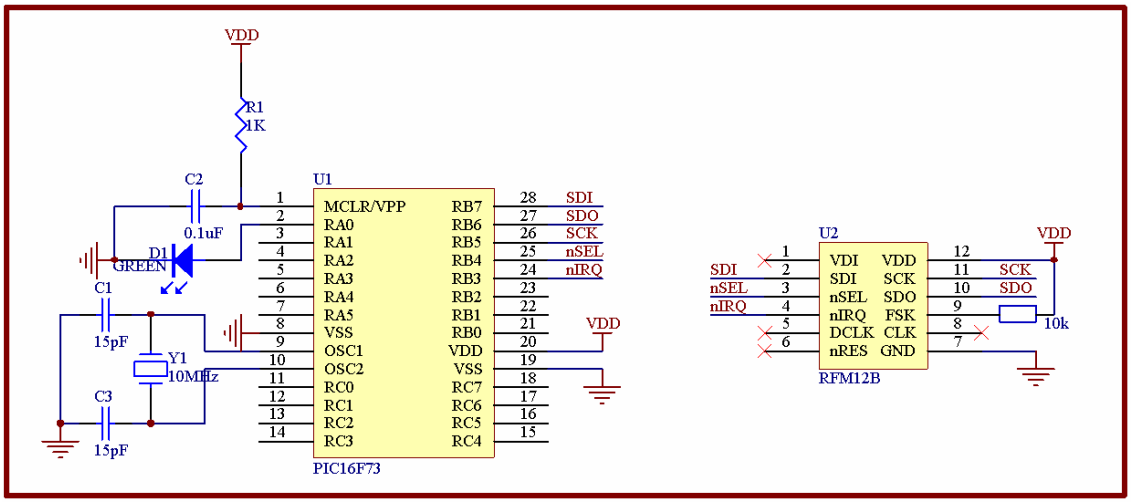
  //Receive Check sum
  i=RF12_RECV();

  //Disable FIFO
  RFXX_WRT_CMD(0xCA81);

  //Package chkeck
  if(ChkSum==i){
    LEDG_ON();
    Delay_ms(200);
    LEDG_OFF();
  }
}
```

```
}
```

5. Example 2（for PIC microcontroller）



RF12 transmitter demo:

```
/*********************************************************
    copyright (c) 2006
Title:              RFM12A transmitter simple example based on PIC C
Current version:    v1.0
Function:           Package send Demo
Processor           PIC16F73 DIP-28
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:               2006-11-21
*********************************************************/
#include "pic.h"

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

```
#define  SDI           RB7
#define  SDO           RB6
#define  SCK           RB5
#define  nSEL          RB4

#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 );
void DelayUs( uint us );
void DelayMs(uint ms);
void WriteFSKbyte( uchar DATA );

__CONFIG(0x3FF2);

void Init_RF12(void)
{
  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(0x94C0);//VDI,FAST,134kHz,0dBm,-103dBm
  WriteCMD(0xC2AC);
  WriteCMD(0xCA80);
  WriteCMD(0xCED4);
  WriteCMD(0xCA83);//FIFO8,SYNC,
  WriteCMD(0xC49B);
  WriteCMD(0x9850);//!mp,90kHz,MAX OUT
  WriteCMD(0xCC77);
  WriteCMD(0xE000);//NOT USE
  WriteCMD(0xC80E);//NOT USE
  WriteCMD(0xC000);//1.0MHz,2.2V
```

```
}

void main()
{
  uint ChkSum=0;
  Init_RF12();
  while(1)
   {
     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
      DelayMs(1000);

   }
}

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

void Write1( 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)
      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--)
   {
     DelayUs(1);
   }
  }
}
```

RF12 receiver demo:

```
/*************************************************************
    copyright (c) 2006
Title:              RFM12A transmitter simple example based on PIC C
Current version:    v1.0
Function:           Package send Demo
Processor           PIC16F73 DIP-28
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:               2006-11-17
*************************************************************/
#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(0x94C0);//VDI,FAST,134kHz,0dBm,-103dBm
  WriteCMD(0xC2AC);
  WriteCMD(0xCA80);
  WriteCMD(0xCED4);
  WriteCMD(0xCA83);//FIFO8,SYNC,
  WriteCMD(0xC49B);
  WriteCMD(0x9850);//!mp,90kHz,MAX OUT
  WriteCMD(0xCC77);
  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 )
{
  SDI=0;
  SCK=0;
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
  NOP();
```

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

void Write1( void )
{
  SDI=1;
  SCK=0;
  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;
}



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
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

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 MECHANTABILITY OR FITNESS FOR A ARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.

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