RF01 programming guide

1. Brief description

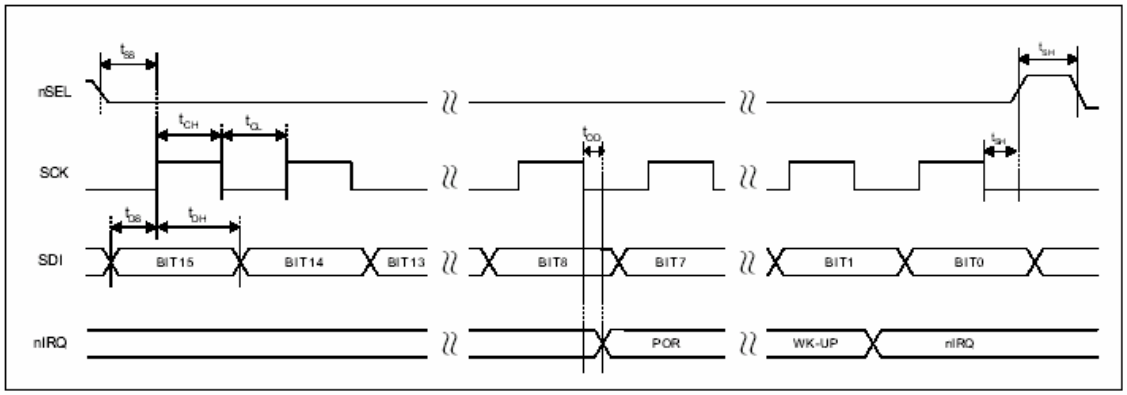
RF01 is a low cost FSK receive IC witch integrated all RF functions in a single chip. It only need a MCU, a crystal, a decouple capacitor and antenna to build a hi reliable FSK receiver. The operation frequency can cover 300 to 1000MHz.

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: Fc=310+F*0.0025 MHz

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, F is frequency parameter and 36≤F≤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:

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

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：

Vlb=2.2+T*0.1 [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	rl1	rl0	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

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

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	VDI & Sync-word
1	1	Always

ff: Enable FIFO fill

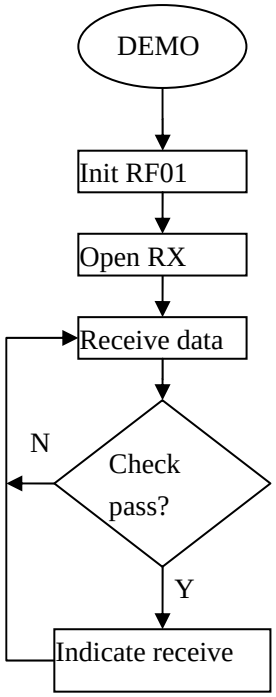
fe: Enable FIFO function

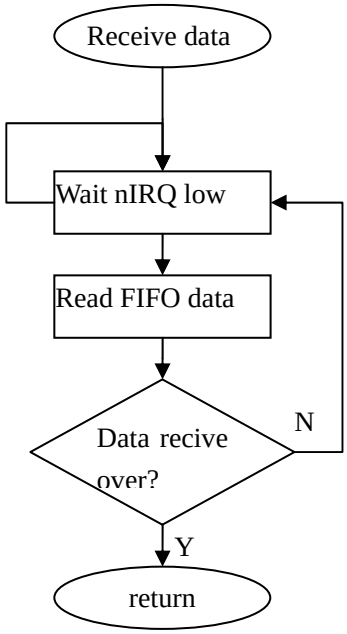
12. Status Read Command

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

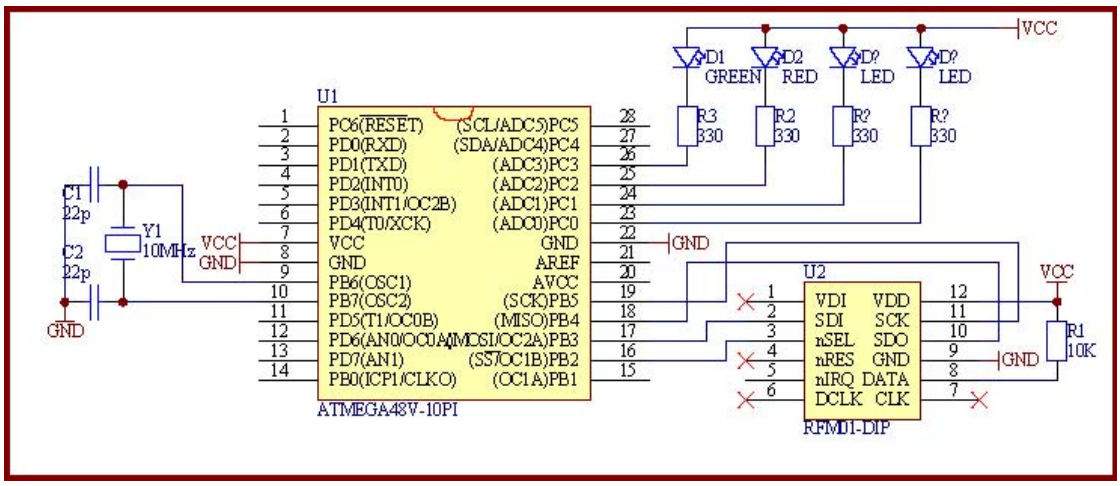




Note: After RF01 initialization, Open FIFO receive mode and wait nIRQ low, only then MCU can

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

4. Example 1（for AVR microcontroller）




```
/*********************************************************************
    copyright (c) 2006
Title:              RF01 simple example based on AVR C
Current version:    v1.0
Function:           Package Receive 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        RF01 SIDE
         SCK---------------->SCK
        MISO<----------------SDO
        MOSI---------------->SDI
          SS---------------->nSEL
                             DATA:Pull up to VDD
        INT0<----------------nIRQ
```

PC0~PC3: LED0~LED3

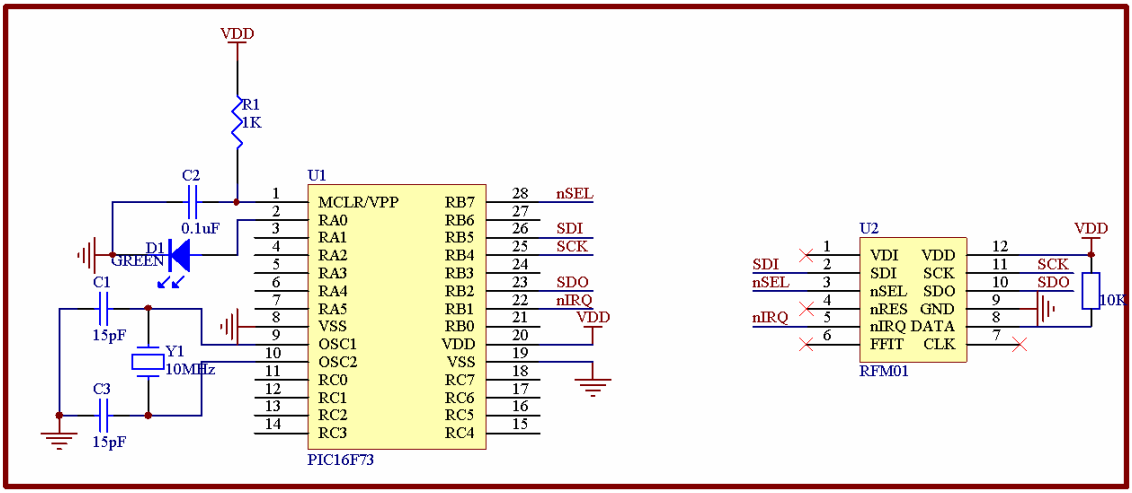
```
*****************************************************************************/
#include <mega48.h>

#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 PORT_LED        PORTC
#define DDR_LED         DDRC

#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 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 LED_OUTPUT()     DDR_LED |=0x0F
#define LED0_ON()        PORT_LED&=~(1<<0)
#define LED0_OFF()       PORT_LED|= (1<<0)
#define LED0_TRG()       PORT_LED^= (1<<0)

#define LED1_ON()        PORT_LED&=~(1<<1)
#define LED1_OFF()       PORT_LED|= (1<<1)
#define LED1_TRG()       PORT_LED^= (1<<1)

#define LED2_ON()        PORT_LED&=~(1<<2)
#define LED2_OFF()       PORT_LED|= (1<<2)
#define LED2_TRG()       PORT_LED^= (1<<2)

#define LED3_ON()        PORT_LED&=~(1<<3)
#define LED3_OFF()       PORT_LED|= (1<<3)
#define LED3_TRG()       PORT_LED^= (1<<3)

unsigned char RF_RXBUF[22];
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);
}
unsigned char RF01_RDFIFO(void){
  unsigned char i,Result;
  LOW_SCK();
  LOW_SDI();
  LOW_SEL();
  for(i=0;i<16;i++){//skip status bits
    HI_SCK();
    HI_SCK();
    LOW_SCK();
    LOW_SCK();
  }
  Result=0;
  for(i=0;i<8;i++){//read fifo data byte
    Result<<=1;
    if(SDO_HI()){
      Result|=1;
    }
    HI_SCK();
    HI_SCK();
    LOW_SCK();
    LOW_SCK();
  };
  HI_SEL();
  return(Result);
}
void main(void)
{
  unsigned int intI,intJ;
  unsigned char i,j,ChkSum;
```

```
  for(intI=0;intI<10000;intI++)for(intJ=0;intJ<123;intJ++);
  RFXX_PORT_INIT();

  RFXX_WRT_CMD(0x0000);
  RFXX_WRT_CMD(0x898A);//433BAND,134kHz
  RFXX_WRT_CMD(0xA640);//434MHz
  RFXX_WRT_CMD(0xC847);//4.8kbps
  RFXX_WRT_CMD(0xC69B);//AFC setting
  RFXX_WRT_CMD(0xC42A);//Clock recovery manual control,Digital filter,DQD=4
  RFXX_WRT_CMD(0xC240);//output 1.66MHz
  RFXX_WRT_CMD(0xC080);
  RFXX_WRT_CMD(0xCE88);//use FIFO
  RFXX_WRT_CMD(0xCE8B);
  RFXX_WRT_CMD(0xC081);//OPEN RX
  DDRB|=(1<<RFXX_DATA);

  DDRD&=~(1<<2);


  LED_OUTPUT();

  i=0;
  while(1){
    while(!(PIND&(1<<2))){//polling the nIRQ data
      RF_RXBUF[i++]=RF01_RDFIFO();//read FIFO data
      if(i==18){
        i=0;
        RFXX_WRT_CMD(0xCE88);   //reset FIFO for next frame recognition
        RFXX_WRT_CMD(0xCE8B);
        ChkSum=0;
        for(j=0;j<16;j++){
          ChkSum+=RF_RXBUF[j];  //calculate checksum
        }
        if(ChkSum==RF_RXBUF[16]){//frame check
          LED0_TRG();//receive indication
        }
      }
    }
  }
}
```

5. Example 2（for PIC microcontroller）




```
/*************************************************************
    copyright (c) 2006
Title:              RF01 simple example based on PIC C
Current version:    v1.0
Function:           Package receive 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:               2006-11-14
*************************************************************/
#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
#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(0xCE88);//use FIFO
  WriteCMD(0xCE8B);
  WriteCMD(0xC081);//OPEN RX
  while(1)
   {
     while(!nIRQ)
     {
      RF_RXBUF[i++]=RF01_RDFIFO();
      if(i==17)
       {
        i=0;
        WriteCMD(0xCE88);
        WriteCMD(0xCE8B);
        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();
  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 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);
   }
}
```

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