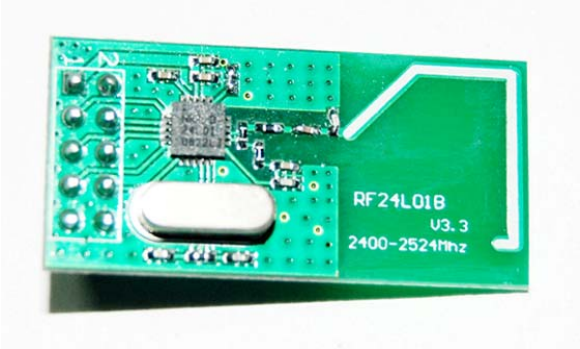
RF24L01无线通讯模块
开发指南

作者　　刘春伟　胡文明

一、模块介绍

RF24L01模块有两个型号：RF24L01B和RF24L01SE



RF24L01B（PCB板载天线）（尺寸：37mm*18mm*1.6mm）

RF24L01SE （外置天线）(尺寸：30mm*18mm*1.6mm)

RF24L01特点:

(1) 2.4Ghz 全球开放ISM 频段免许可证使用

(2) 最高工作速率2Mbps，高效GFSK调制，抗干扰能力强，特别适合工业控制场合

(3) 126 频道，满足多点通信和跳频通信需要

(4) 内置硬件CRC 检错和点对多点通信地址控制

(5) 低功耗1.9 - 3.6V 工作，待机模式下状态为22uA；掉电模式下为900nA

(6) 内置2.4Ghz 天线，体积小巧 34mm X 17mm

(7) 模块可软件设地址，只有收到本机地址时才会输出数据（提供中断指示），可直接接各种单片机使用，软件编程非常方便

(8) 内置专门稳压电路，使用各种电源包括DC/DC 开关电源均有很好的通信效果

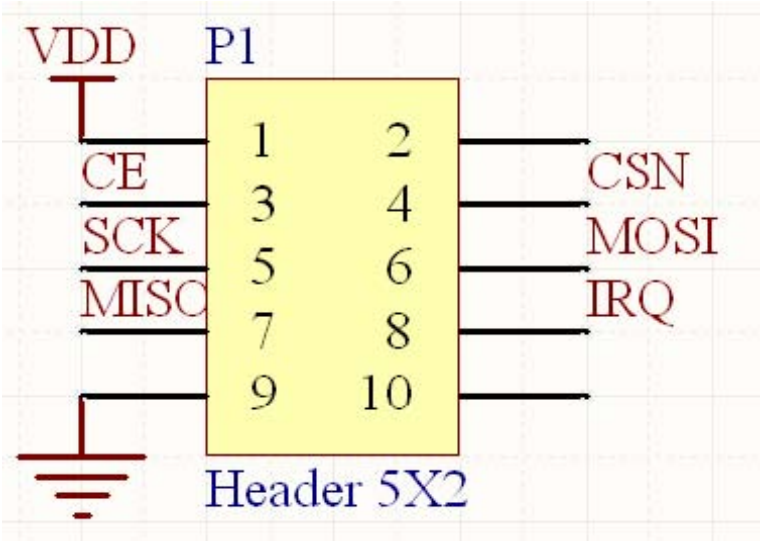
(9) 标准DIP间距接口，便于嵌入式应用

(10) 工作于Enhanced ShockBurst 具有Automatic packet handling, Auto packet transaction handling,具有可选的内置包应答机制，极大的降低丢包率。

（11） 与51系列单片机P0口连接时候，需要加10K的上拉电阻，与其余口连接不需要。

（12） 其他系列的单片机，如果是5V的，请参考该系列单片机IO口输出电流大小，如果超过10mA，需要串联电阻分压，否则容易烧毁模块！如果是3.3V的，可以直接和RF2401模块的IO口线连接。比如AVR系列单片机如果是5V的，一般串接2K的电阻。

二、接口电路



说明:

(1) VCC脚接电压范围为 1.9V~3.6V之间，不能在这个区间之外，超过3.6V将会烧毁模块。推荐电压3.3V左右。

(2) 除电源VCC和接地端，其余脚都可以直接和普通的5V单片机IO口直接相连，无需电平转换。当然对3V左右的单片机更加适用了。

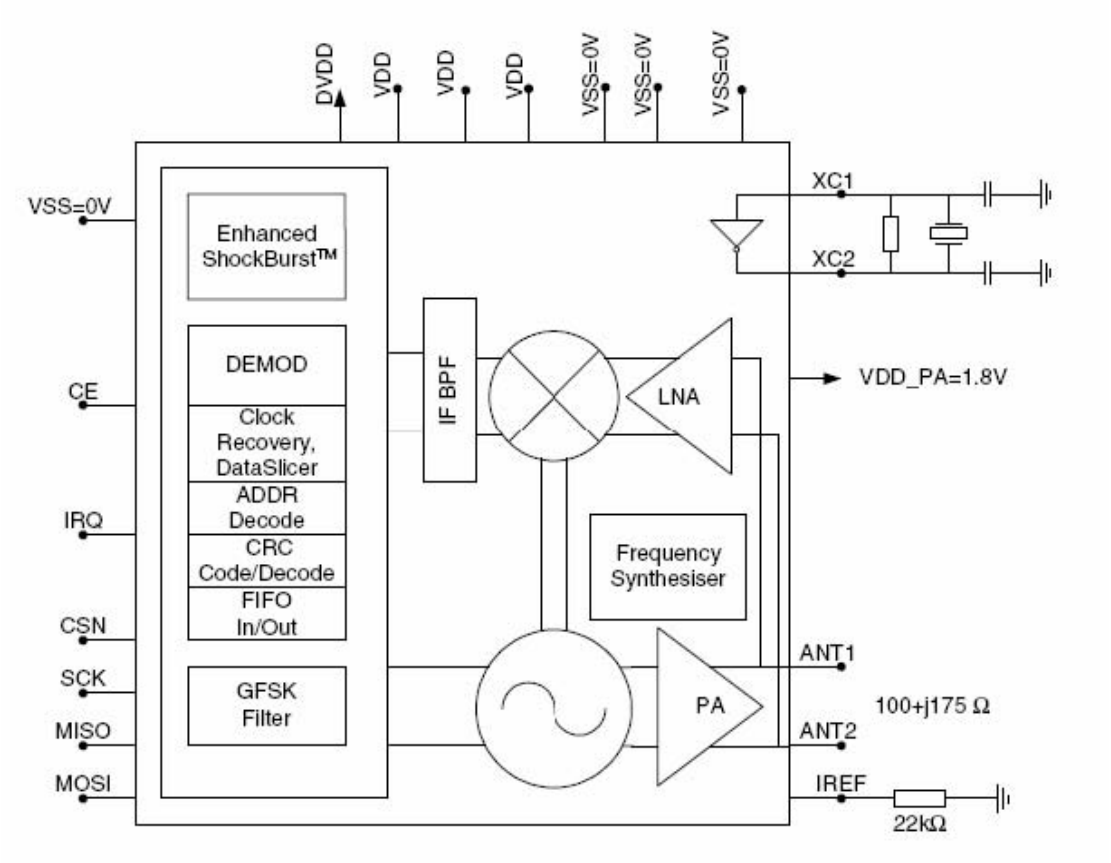
(3) 硬件上面没有SPI的单片机也可以控制本模块，用普通单片机IO口模拟SPI不需要单片机真正的串口介入，只需要普通的单片机IO口就可以了，当然用串口也可以了。

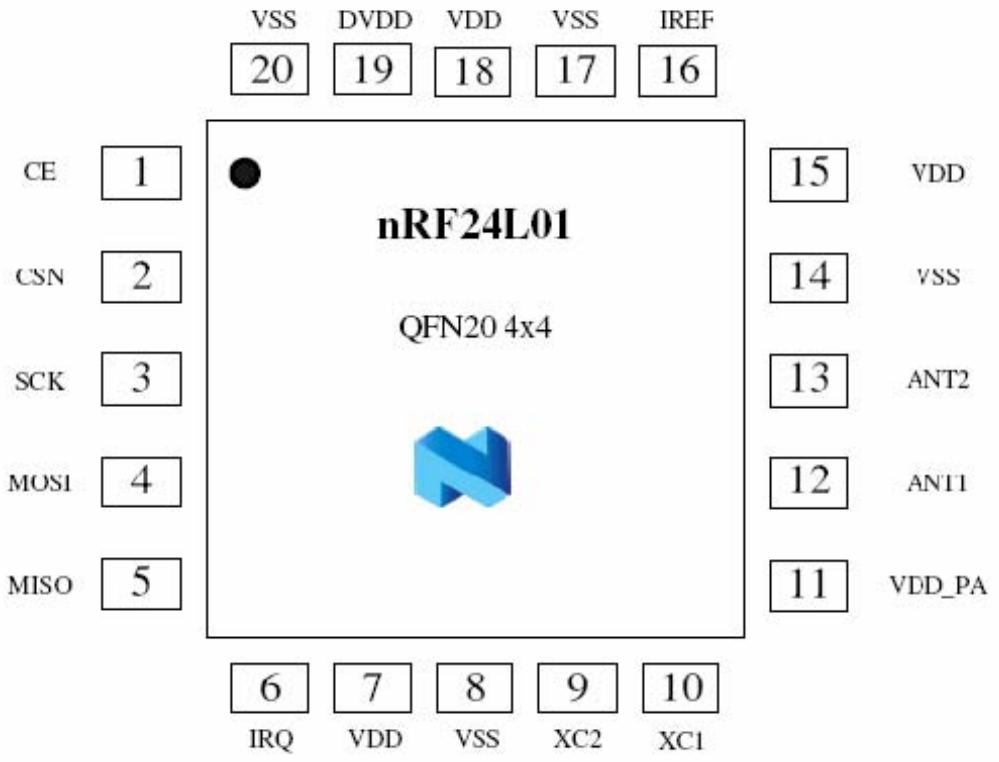
(4) 9脚接地脚,需要和母板的逻辑地连接起来；2脚和9脚悬空。

(5) 排针间距为100mil,标准DIP插针，如果需要其他封装接口，比如密脚插针，或者其他形式的接口，可以联系我们定做。

三、模块结构和引脚说明

RF24L01模块使用Nordic公司的nRF24L01芯片开发而成。





Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin
7	VDD	Power	Power Supply (+3V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply (+1.8V) to Power Amplifier
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+3V DC)
16	IREF	Analog Input	Reference current
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+3V DC)
19	DVDD	Power Output	Positive Digital Supply output for de-coupling purposes
20	VSS	Power	Ground (0V)

四、工作方式

RF24L01有工作模式有四种:

收发模式

配置模式

空闲模式

关机模式

工作模式由PWR_UP register 、PRIM_RX register和CE决定，详见下表。

Mode	PWR_UP register	PRIM_RX register	CE	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO
TX mode	1	0	1→0	Stays in TX mode until packet transmission is finished
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

4.1 收发模式

收发模式有Enhanced ShockBurstTM收发模式、ShockBurstTM收发模式和直接收发模式三种，收发模式由器件配置字决定，具体配置将在器件配置部分详细介绍。

4.1.1 Enhanced ShockBurstTM收发模式

Enhanced ShockBurstTM收发模式下，使用片内的先入先出堆栈区，数据低速从微控制器送入，但高速(1Mbps)发射，这样可以尽量节能，因此，使用低速的微控制器也能得到很高的射频数据发射速率。与射频协议相关的所有高速信号处理都在片内进行，这种做法有三大好处：尽量节能；低的系统费用(低速微处理器也能进行高速射频发射)；数据在空中停留时间短，抗干扰性高。Enhanced ShockBurstTM技术同时也减小了整个系统的平均工作电流。

在Enhanced ShockBurstTM收发模式下， RF24L01自动处理字头和CRC校验码。在接收数据时，自动把字头和CRC校验码移去。在发送数据时，自动加上字头和CRC校验码，在发送模式下，置CE为高，至少10us，将时发送过程完成后。

4.1.1.1 Enhanced ShockBurstTM发射流程

A. 把接收机的地址和要发送的数据按时序送入RF24L01;

B. 配置CONFIG寄存器，使之进入发送模式。 C. 微控制器把CE置高（至少10us），激发RF24L01进行Enhanced ShockBurstTM发射； D. RF24L01的Enhanced ShockBurstTM发射 (1) 给射频前端供电； (2)射频数据打包(加字头、CRC校验码)； (3)高速发射数据包； (4)发射完成，RF24L01进入空闲状态。

4.1.1.2 Enhanced ShockBurstTM接收流程 A. 配置本机地址和要接收的数据包大小； B. 配置CONFIG寄存器，使之进入接收模式，把CE置高。

C. 130us后，RF24L01进入监视状态，等待数据包的到来； D. 当接收到正确的数据包(正确的地址和CRC校验码)，RF2401自动把字头、地址和CRC校验位移去;

E. RF24L01通过把STATUS寄存器的RX_DR置位(STATUS一般引起微控制器中断)通知微控制器； F. 微控制器把数据从RF2401读出； G. 所有数据读取完毕后，可以清除STATUS寄存器。RF2401可以进入四种主要的模式之一。

4.1.2 ShockBurstTM收发模式

ShockBurstTM收发模式可以与Nrf2401a,02,E1及E2兼容，具体表

述前看本公司的-RF2401文档。

4.2 空闲模式

RF24L01的空闲模式是为了减小平均工作电流而设计，其最大的优点是，实现节能的同时，缩短芯片的起动时间。在空闲模式下，部分片内晶振仍在工作，此时的工作电流跟外部晶振的频率有关。

4.4 关机模式

在关机模式下，为了得到最小的工作电流，一般此时的工作电流为900nA左右。关机模式下，配置字的内容也会被保持在RF2401片内，这是该模式与断电状态最大的区别。

五、配置RF24L01模块

RF2401的所有配置工作都是通过SPI完成，共有30字节的配置字。

我们推荐RF24L01工作于Enhanced ShockBurstTM 收发模式，这种工作模式下，系统的程序编制会更加简单，并且稳定性也会更高，因此，下文着重介绍把RF24L01配置为Enhanced ShockBurstTM收发模式的器件配置方法。

ShockBurstTM的配置字使RF24L01能够处理射频协议，在配置完成后，在RF24L01工作的过程中，只需改变其最低一个字节中的内容，以实现接收模式和发送模式之间切换。

ShockBurstTM的配置字可以分为以下四个部分:

数据宽度: 声明射频数据包中数据占用的位数。这使得RF24L01能够区分接收数据包中的数据和CRC校验码;

地址宽度：声明射频数据包中地址占用的位数。这使得RF24L01能够区分地址和数据；

地址：接收数据的地址，有通道0到通道5的地址；

CRC：使RF24L01能够生成CRC校验码和解码。

当使用RF24L01片内的CRC技术时，要确保在配置字(CONFIG的EN_CRC)中CRC校验被使能，并且发送和接收使用相同的协议。

RF24L01配置字的CONFIG寄存器的位描述如下表所示。
RF24L01 CONFIG配置字描述

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the IRQ pin 0: Reflect RX_DR as active low interrupt on the IRQ pin
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' – 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	1: PRX, 0: PTX

六、参考源代码

```
#include <reg51.h>

//<nRF2401_Pins 对应引脚>
sbit    MISO =P1^3;
sbit    MOSI =P1^4;
sbit SCK         =P1^5;
sbit CE         =P1^6;
sbit CSN    =P3^7;
sbit IRQ    =P1^2;

sbit       LED2       =P3^5;
sbit       LED1       =P3^4;
sbit       KEY1       =P3^0;
sbit       KEY2       =P3^1;


// SPI(nRF24L01) commands
#define READ_REG              0x00    // Define read command to register
#define WRITE_REG             0x20    // Define write command to register
#define RD_RX_PLOAD           0x61    // Define RX payload register address
#define WR_TX_PLOAD           0xA0    // Define TX payload register address
#define FLUSH_TX              0xE1    // Define flush TX register command
#define FLUSH_RX              0xE2    // Define flush RX register command
#define REUSE_TX_PL           0xE3    // Define reuse TX payload register command
#define NOP                   0xFF    // Define No Operation, might be used to read status
register


//*****************************************************//
// SPI(nRF24L01) registers(addresses)
#define CONFIG                0x00    // 'Config' register address
#define EN_AA                 0x01    // 'Enable Auto Acknowledgment' register address
#define EN_RXADDR             0x02    // 'Enabled RX addresses' register address
#define SETUP_AW              0x03    // 'Setup address width' register address
#define SETUP_RETR            0x04    // 'Setup Auto. Retrans' register address
#define RF_CH                 0x05    // 'RF channel' register address
#define RF_SETUP              0x06    // 'RF setup' register address
#define STATUS                0x07    // 'Status' register address
#define OBSERVE_TX            0x08    // 'Observe TX' register address
#define CD                    0x09    // 'Carrier Detect' register address
#define RX_ADDR_P0            0x0A    // 'RX address pipe0' register address
#define RX_ADDR_P1            0x0B    // 'RX address pipe1' register address
#define RX_ADDR_P2            0x0C    // 'RX address pipe2' register address
#define RX_ADDR_P3            0x0D    // 'RX address pipe3' register address
#define RX_ADDR_P4            0x0E    // 'RX address pipe4' register address
#define RX_ADDR_P5            0x0F    // 'RX address pipe5' register address
#define TX_ADDR               0x10    // 'TX address' register address
#define RX_PW_P0              0x11    // 'RX payload width, pipe0' register address
#define RX_PW_P1              0x12    // 'RX payload width, pipe1' register address
#define RX_PW_P2              0x13    // 'RX payload width, pipe2' register address
#define RX_PW_P3              0x14    // 'RX payload width, pipe3' register address
#define RX_PW_P4              0x15    // 'RX payload width, pipe4' register address
```

```
#define RX_PW_P5             0x16    // 'RX payload width, pipe5' register address
#define FIFO_STATUS    0x17        // 'FIFO Status Register' register address
//-----------------------------------------------------------------

// 写一个字节到24L01，同时读出一个字节
uchar SPI_RW(uchar byte)
{
 uchar bit_ctr;
    for(bit_ctr=0;bit_ctr<8;bit_ctr++) // output 8-bit
    {
  MOSI = (byte & 0x80);              // output 'byte', MSB to MOSI
  byte = (byte << 1);                // shift next bit into MSB..
  SCK = 1;                           // Set SCK high..
  byte |= MISO;                      // capture current MISO bit
  SCK = 0;                           // ..then set SCK low again
    }
    return(byte);                    // return read byte
}

//   向寄存器reg写一个字节，同时返回状态字节
uchar SPI_RW_Reg(BYTE reg, BYTE value)
{
 uchar status;

 CSN = 0;                            // CSN low, init SPI transaction
 status = SPI_RW(reg);               // select register
 SPI_RW(value);                      // ..and write value to it..
 CSN = 1;                            // CSN high again

 return(status);                     // return nRF24L01 status byte
}

// 读出bytes字节的数据
uchar SPI_Read_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
{
 uchar status,byte_ctr;

 CSN = 0;                            // Set CSN low, init SPI tranaction
 status = SPI_RW(reg);               // Select register to write to and read status byte

 for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)
  pBuf[byte_ctr] = SPI_RW(0);       //

 CSN = 1;

 return(status);                     // return nRF24L01 status byte
}

// 写入bytes字节的数据
uchar SPI_Write_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
{
 uchar status,byte_ctr;

 CSN = 0;
 status = SPI_RW(reg);
 for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) //
  SPI_RW(*pBuf++);
```

```
  CSN = 1;                          // Set CSN high again
  return(status);                 //
}

// 接收函数，返回1表示有数据收到，否则没有数据接受到
unsigned char nRF24L01_RxPacket(unsigned char* rx_buf)
{
      unsigned char revale=0;
      // set in RX mode
  SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);        // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
      CE = 1; // Set CE pin high to enable RX device
      dalay130us();
  sta=SPI_Read(STATUS); // read register STATUS's value
  if(RX_DR)       // if receive data ready (RX_DR) interrupt
  {
       CE = 0;     // stand by mode
    SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);// read receive payload from RX_FIFO buffer
    revale =1;
  }
  SPI_RW_Reg(WRITE_REG+STATUS,sta);// clear RX_DR or TX_DS or MAX_RT interrupt flag

  return revale;
}

// 发送函数
void nRF24L01_TxPacket(unsigned char * tx_buf)
{
  CE=0;
  //SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);      // Writes TX_Address to nRF24L01
  //SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); // RX_Addr0 same as TX_Adr for Auto.Ack
  SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH); // Writes data to TX payload
  SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);        // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
  CE=1;
  dalay10us();
      CE=0;
}



// 配置函数
void nRF24L01_Config(void)
{
  //initial io
  CE=0;     // chip enable
    CSN=1;     // Spi disable
    SCK=0;     // Spi clock line init high
      CE=0;
      SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);        // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
  SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);
```

```
 SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);    // Enable Pipe0
 SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x02);       // Setup address width=5 bytes
 SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a);    // 500us + 86us, 10 retrans...
 SPI_RW_Reg(WRITE_REG + RF_CH, 0);
 SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);     // TX_PWR:0dBm, Datarate:1Mbps,
LNA:HCURR
 SPI_RW_Reg(WRITE_REG          +          RX_PW_P0,          RX_PLOAD_WIDTH);
SPI_Write_Buf(WRITE_REG    +    TX_ADDR,    TX_ADDRESS,    TX_ADR_WIDTH);
SPI_Write_Buf(WRITE_REG    +    RX_ADDR_P0,    TX_ADDRESS,    TX_ADR_WIDTH);
CE=1;   //
 }
```

七、公司介绍

杭州飞拓电子科技有限公司组建于2003年3月，位于杭州高新技术开发区东部软件园，拥有一支以博士、硕士为主的软硬件开发团队，并与杭州电子科技大学、浙江大学的相关电子院系、省部级重点实验室建立了紧密合作关系，技术开发能力强大，开发经验丰富。

公司专注于无线通信、单片机开发应用、RFID产品及系统、集成电路设计等领域。主要产品有:

微功率、中大功率、USB系列无线通信/数传模块;

M1卡读写模块、非接触式感应锁识别模块;

高频RFID读写器、有源电子标签;

远距离RFID射频识别系统;

自主开发消费类ＩＣ产品（遥控系列IC、音响系列IC、电源管理IC等）

工业ＩＣ委托设计;

单片机开发应用设计

公司坚持技术自主创新，并依托省内外著名大学的国家级科研机构，紧跟本领域的高新技术发展潮流，以长久合作，持续共赢为宗旨，为客户提供成熟、可靠的无线通信、单片机应用、IC设计等解决方案。欢迎无线通讯、集成电路等相关领域同仁前来访问和洽谈项目合作，欢迎提出新产品需求。

八、联系方式

公司名称：杭州飞拓电子科技有限公司
公司网址：http://www.fytoo.com

电话：0571-87207271
传真：0571-87207271
Email:jimmy.zh@tom.com
MSN: fred.zh@163.com
QQ: 1009531258
地址：浙江省杭州市西湖区华星路99号创业大厦A402室
邮编：310012