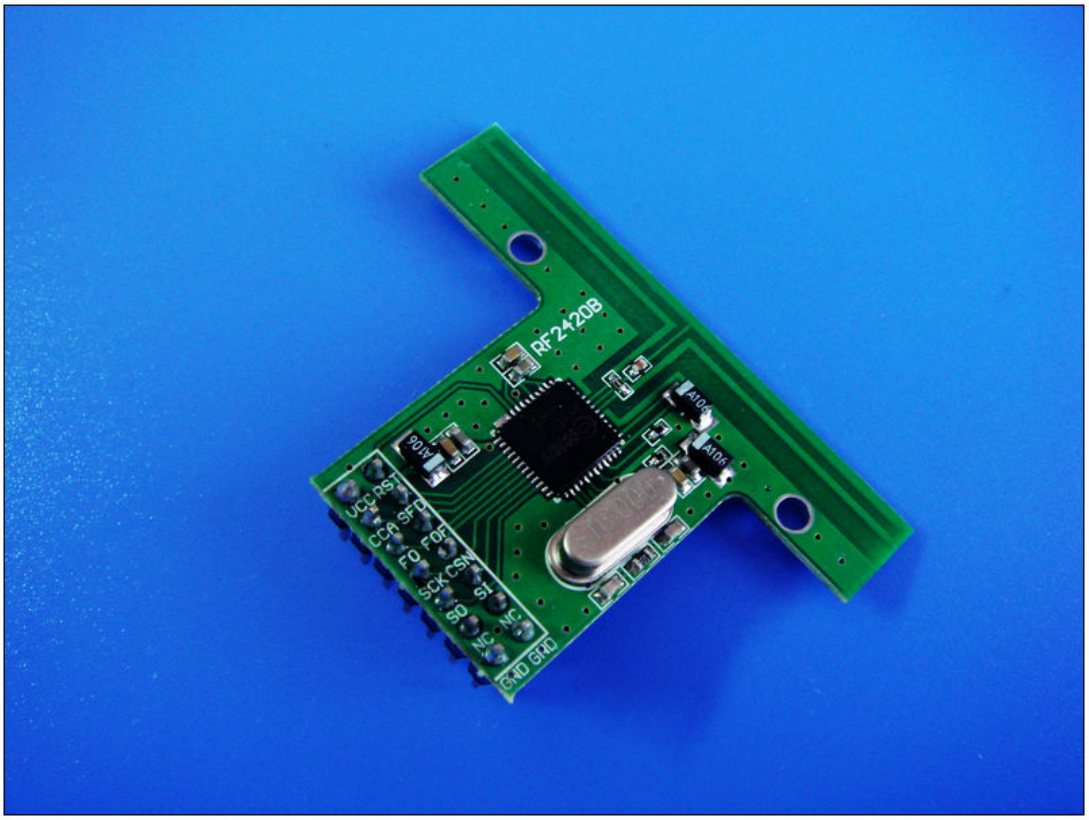
CC2420 无线模块

用户手册



CC2420 模块(尺寸：32*47mm　板厚：1mm)

联系电话：13704018223　陈 工　　E-mail: chj_006@sina.com
在线咨询：QQ:35625400　474882985　　MSN:1188mm88@hotmail.com

联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com

产品说明

CC2420 是 Chipcon（已被 TI 收购）公司推出的首款符合 2.4GHz IEEE802.15.4 标准的射频收发器。该器件包括众多功能，是第一款适用于 ZigBee 产品的 RF 器件。它基于 SmartRF 03 技术，以 0.18um CMOS 工艺制成,只需极少外部元器件，工作在 2400-2483.5MHZ 的 ISM 频段由一个完全集成的频率调制器一个带解调器的接收器一个功率放大器一个晶体震荡器和一个调节器组成。可自动产生前导码，CRC 可以很容易通过 SPI 接口进行编程配置，电流消耗低。性能稳定且功耗极低。CC2420 的选择性和敏感性指数可确保短距离通信的有效性和可靠性。利用此芯片开发的无线通信设备支持数据传输率高达 250kbps 可以实现多点对多点的快速组网。

基本特点：

(1) 工作在 2400-2483.5 MHz 的 ISM 和 SRD 频段.

-采用直接序列扩频方式.

-工作速率 250kbps，码片速率 2 MChip/s.

-使用 O-QPSK 调制方式.

-高灵敏度（-95dBm）.

-较低的电流消耗（RX :13.3 mA TX:17.4 mA).

-抗邻频道干扰能力强(39dB)

-内部集成有VCO、LNA、PA以及电源整流器.

-采用低电压供电(2.1～3.6V).

-输出功率编程可控.

(2) IEEE802.15.4-2003 标准 MAC 层硬件支持.

-前导码与同步字段自动生成与检测.

-CRC-16 自动生成与检测.

-空闲信道检测.

-能量检测、接收信号强度与链路质量指示.

-MAC 层安全保护(CTR,CBC－MAC,CCM)支持.

(3) 采用 4 线 SPI 标准接口,便于 MCU 配置.

(4) 独立的128字节RX和128字节TX数据FIFO.

典型应用场合：

无线传感器网络

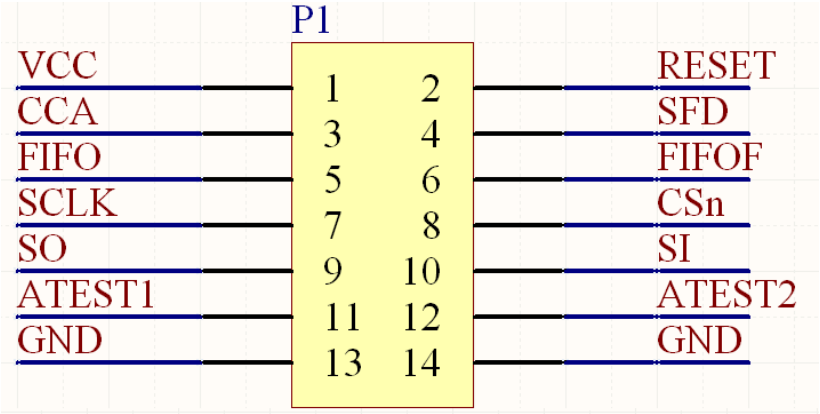
住宅、建筑物（智能家居）控制

工业仪器仪表无线数据采集和控制

无线鼠标、无线键盘、无线类玩具等消费电子

无线门禁、物流跟踪、仓库巡检等 RFID 有源电子标签

模块接口说明



引脚功能说明

引脚	引脚名	引脚类型	描述
1	VCC	电源输入	1.9V-3.6V之间
2	RESET	数字输入	复位，低电平有效
3	CCA	数字输出	空闲通道指示，可配置为其他功能
4	SFD	数字输出	帧定界符接收指示，可配置为其他功能
5	FIFO	数字输出/输出	RX FIFO存在数据指示，串行模式时作为数据输入输出口
6	FIFOF	数字输出	RX FIFO内数据量超过门限值指示，串行模式时作为时钟输出

联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com

7	SCLK	数字输入	四线SPI，时钟输入
8	CSn	数字输入	四线SPI，片选
9	SO	数字输出	四线SPI，数据输出
10	SI	数字输入	四线SPI，数据输入
11，12	ATEST1 ATEST2		无用
13,14	GND	地(模拟)	模拟接地

备注

1.VCC 引脚的电压范围为1.9-3.6V 之间，不能在这个区间之外，如超过 3.6V 将会烧毁模块。推荐电压 3.3V 左右；

2.硬件没有集成SPI功能的单片机也可以控制本模块，用普通单片IO口模拟 SPI 时序进行读写操作即可；

3. 模块接口采用标准2.54mmDIP插针，13 脚、14 脚为接地脚,需要和系统电路的逻辑地连接起来；

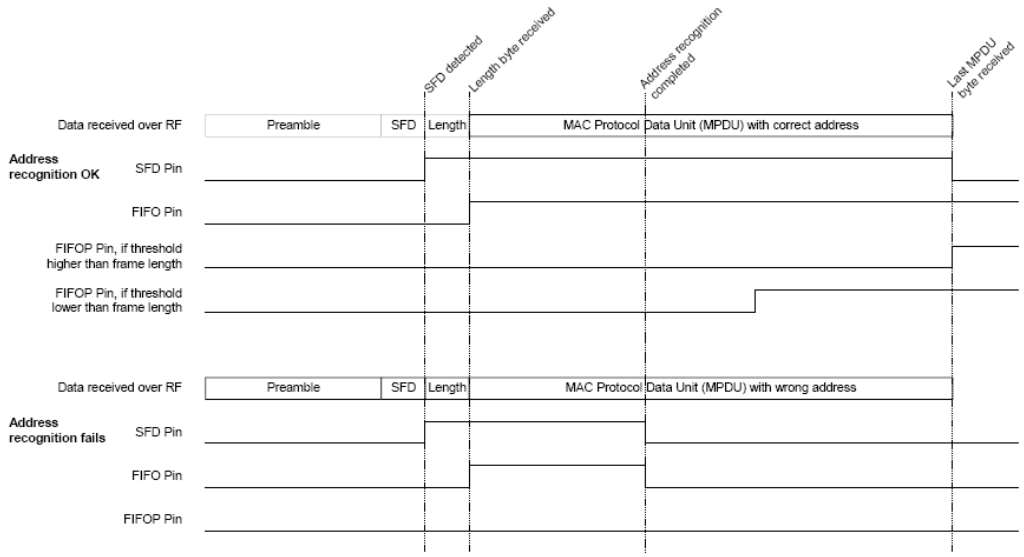
4.与 51 系列单片机 P0 口连接时候，需要加 10K 的上拉电阻,与其余口连接不需要。其他系列的5V单片机，如AVR、PIC，请参考该系列单片机 IO 口输出电流大小，如果超过 10mA，需要串联2-5K电阻分压，否则容易烧毁模块！如果是 3.3V 的MCU，可以直接和IO口连接。

联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com

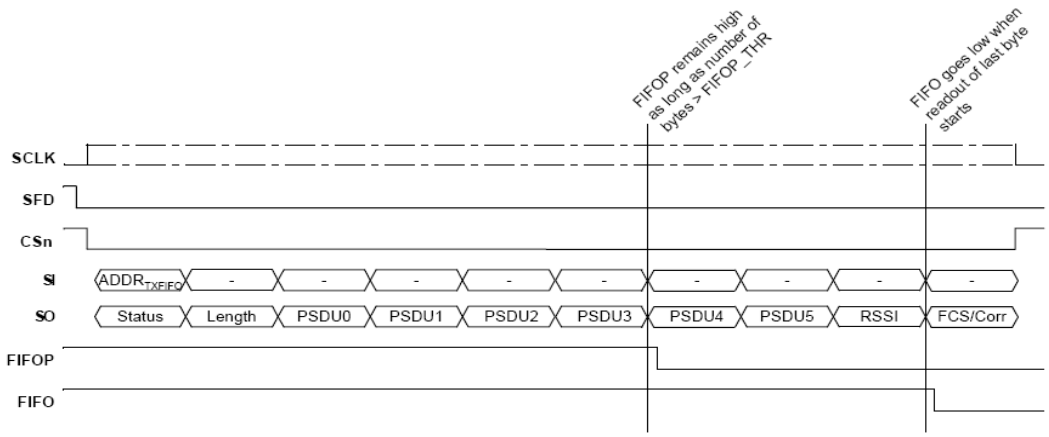
CC2420 模块工作方式

所有配置参数和收发数据都是单片机通过 SPI 接口对 CC2420 进行读写操作来完成的。SIP 接口的待机模式、发送模式以及接收等工作模式都通过 SPI 指令进行设置。

接收模式流程

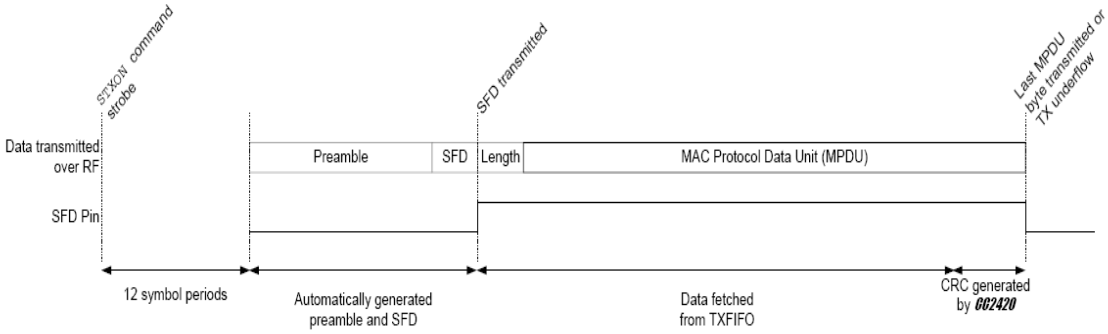


读取 FIFO 数据流程



联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com

数据发送流程



程序参考设计

用CC2420模块无需掌握任何专业无线或高频方面的理论，读者只需要具备一定的C语言程序基础即可。本文档没有涉及到的问题，读者可以参考CC2420官方手册或向我们寻求技术支持。

同时，为便于用户开发，我们提供系列配套评估套件，为产品开发保驾护航，使无线应用开发大大加速，并避免不必要的误区。以下为范例程序中的部分相关代码段。

CC2420 寄存器读写配置

CC2420 通过 SPI 接口与单片机通讯，因此必须首先了解 SPI 接口。标配 SPI 外围串行接口由四条线构成：

MOSI 主机输出从机输入 （主机写操作）

MISO 主机输入从机输出 （主机读操作）

SCK 串行时钟信号，由主机控制

CSN 片选信号，低电平有效

SPI 读操作代码

```
uint8 SPI_Read(void)
{
    uint8 i,rxdata;
    rxdata = 0x00;
    for (i = 0;i < 8;i++)
    {
        rxdata = rxdata<<1;
        SCLK_ON();
        if (MISO_IN)
       {
            rxdata |= 0x01;
        }
        else
        {
            rxdata &= ~0x01;
        }
        SCLK_OFF();
     }
     return rxdata;
}
```

SPI 写操作代码

```
void SPI_Write(uint8 txdata)
{
    uint8 i;
    for (i = 0;i < 8;i++)
    {
        if (txdata&0x80)
        {
            MOSI_ON();
```

```
        }
        else
        {
            MOSI_OFF();
        }
        SCLK_ON();
        txdata = txdata<<1;
        SCLK_OFF();
    }
}
```

CC2420 配置寄存器读操作

```
uint16 CC2420_ReadReg(uint8  addr)
{
    uint16 value;
    CSN_OFF();
    SPI_Write(addr|REG_READ);
    value = SPI_Word_Read();
    CSN_ON();
    return value;
}
```

CC2420 配置寄存器写操作

```
void  CC2420_WriteReg(uint8  addr, uint16 value)
{
    CSN_OFF();
    SPI_Write(addr|REG_WRITE);
    SPI_Word_Write(value);
    CSN_ON();
}
```

CC2420 RAM 读操作

```
uint8 CC2420_RAM_Read(uint8 addr,uint8 block)
{
    uint8 value;
    CSN_OFF();
    SPI_Write(addr|RAM);
    SPI_Write((block<<6)|RAM_READ);
    value = SPI_Read();
```

```
    CSN_ON();
    return value;
}
```

CC2420 RAM 写操作

```
void CC2420_RAM_Write(uint8  addr,uint8 block, uint8 value)
{
    CSN_OFF();
    SPI_Write(addr|RAM);
    SPI_Write((block<<6)|RAM_WRITE);
    SPI_Write(value);
    CSN_ON();
}
```

CC2420 初始化

```
void CC2420_Init(void)
{
    RESET_OFF();
    delay_ms(10);
    RESET_ON();
    delay_ms(10);
    CC2420_Command(CMD_SXOSCON);
    delay_ms(10);
    CC2420_PSDU[ 1 ] =
    (PAN_ID_COMPRESSION<<6)|(ACKNOWLEDGMENT_REQUEST<<5)|
    (FRAME_PENDING<<4)|(SECURITY_ENABLE<<3)|(FRAME_TYPE_DATA<<0);
    CC2420_PSDU[ 2 ] =
    (SOURCE_ADDRESSING_MODE<<6)|(FRAME_VERSION<<4)|
    (DEST_ADDRESSING_MODE<<2);
    CC2420_PSDU[ 3 ] = SEQUENCE_NUMBER;
    CC2420_PSDU[ 4 ] =  CC2420_Destination_PANID[0];
    CC2420_PSDU[ 5 ] =  CC2420_Destination_PANID[1];
    CC2420_PSDU[ 6 ] =  CC2420_Destination_IEEEAddr[0];
    CC2420_PSDU[ 7 ] =  CC2420_Destination_IEEEAddr[1];
    CC2420_PSDU[ 8 ] =  CC2420_Destination_IEEEAddr[2];
    CC2420_PSDU[ 9 ] =  CC2420_Destination_IEEEAddr[3];
    CC2420_PSDU[10] =  CC2420_Destination_IEEEAddr[4];
    CC2420_PSDU[11] =  CC2420_Destination_IEEEAddr[5];
    CC2420_PSDU[12] =  CC2420_Destination_IEEEAddr[6];
```

```
    CC2420_PSDU[13] =  CC2420_Destination_IEEEAddr[7];
    CC2420_PSDU[14] =  CC2420_Source_PANID[0];
    CC2420_PSDU[15] =  CC2420_Source_PANID[1];
    CC2420_RAM_Write(RAM_PANID, 2,  CC2420_Source_PANID[0]);
    CC2420_RAM_Write(RAM_PANID+1, 2, CC2420_Source_PANID[1]);
    CC2420_PSDU[16] =  CC2420_Source_IEEEAddr[0];
    CC2420_PSDU[17] =  CC2420_Source_IEEEAddr[1];
    CC2420_PSDU[18] =  CC2420_Source_IEEEAddr[2];
    CC2420_PSDU[19] =  CC2420_Source_IEEEAddr[3];
    CC2420_PSDU[20 ] = CC2420_Source_IEEEAddr[4];
    CC2420_PSDU[21] =  CC2420_Source_IEEEAddr[5];
    CC2420_PSDU[22] =  CC2420_Source_IEEEAddr[6];
    CC2420_PSDU[23] =  CC2420_Source_IEEEAddr[7];
    CC2420_RAM_Write(RAM_IEEEADR, 2, CC2420_Source_IEEEAddr[0]);
    CC2420_RAM_Write(RAM_IEEEADR+1, 2, CC2420_Source_IEEEAddr[1]);
    CC2420_RAM_Write(RAM_IEEEADR+2, 2, CC2420_Source_IEEEAddr[2]);
    CC2420_RAM_Write(RAM_IEEEADR+3, 2, CC2420_Source_IEEEAddr[3]);
    CC2420_RAM_Write(RAM_IEEEADR+4, 2, CC2420_Source_IEEEAddr[4]);
    CC2420_RAM_Write(RAM_IEEEADR+5, 2, CC2420_Source_IEEEAddr[5]);
    CC2420_RAM_Write(RAM_IEEEADR+6, 2, CC2420_Source_IEEEAddr[6]);
    CC2420_RAM_Write(RAM_IEEEADR+7, 2, CC2420_Source_IEEEAddr[7]);
    CC2420_WriteReg(REG_MDMCTRL0,
    CCA_HYST|CCA_MODE|PREAMBLE_LENGTH|AUTOCRC|ADR_DECODE);
    CC2420_WriteReg(REG_SYNCWORD,SYNCWORD);
    CC2420_WriteReg(REG_SECCTRL0,0);
    CSN_OFF();
    SPI_Write(REG_RXFIFO|REG_READ);
    SPI_Read();
    CSN_ON();
    CC2420_Command(CMD_SFLUSHRX);
    CC2420_Command(CMD_SFLUSHTX);
    delay_ms(10);
}
```

CC2420 FIFO 发送流程

FIFO 写数据操作

```
void  CC2420_WriteTXFIFO(void)
{
    uint8 i;
```

```
    CC2420_Command(CMD_SFLUSHTX);
    CSN_OFF();
    SPI_Write(REG_TXFIFO|REG_WRITE);
    SPI_Write(CC2420_PSDU[0]);
    for(i=0;i<CC2420_PSDU[0];i++)
    {
        SPI_Write(CC2420_PSDU[1+i]);
    }
    CSN_ON();

}
```

FIFO 数据发送操作

```
void CC2420_TxPacket(void)
{
    CC2420_Command(CMD_SRFOFF);
    CC2420_Command(CMD_STXON);
    while(!SFD_IN);
    while(SFD_IN);
}
```

CC2420 FIFO 接收流程

接收模式设置

```
void CC2420_SetRxMode(void)

{

    CC2420_Command(CMD_SRFOFF);

    CC2420_Command(CMD_SRXON);

}
```

FIFO 接收数据

```
uint8 CC2420_RxPacket(void)
{
     if((!SFD_IN)&&(FIFO_IN))
    {
        return TRUE;
    }
      return FALSE;
}
```

收到数据后读取 FIFO 数据

```
void CC2420_ReadRXFIFO(void)
{
    uint8 i;
    CSN_OFF();
    SPI_Write(REG_RXFIFO|REG_READ);
    CC2420_PSDU[0] = SPI_Read();
    for(i=0;i<CC2420_PSDU[0];i++)
    {
        CC2420_PSDU[1+i] = SPI_Read();
    }
    CSN_ON();
    CC2420_Command(CMD_SFLUSHRX); }
```

联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com

无线应用注意事项

(1) 无线模块的 VCC 电压范围为 1.8V-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 51 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。模块按照接口提示和母板的逻辑地连接起来

(4) 标准 DIP 插针，如需要其他封装接口，或其他形式的接口，可联系我们定做。

(5) 任何单片机都可实现对无线模块的数据收发控制，并可根据我们提供的程序，然后结合自己擅长的单片机型号进行移植;

(6) 频道的间隔的说明：实际要想 2 个模块同时发射不相互干扰，两者频道间隔应该至少相差 1MHZ，这在组网时必须注意,否则同频比干扰。

(7) 实际用户可能会应用其他自己熟悉的单片机做为主控芯片，所以，建议大家在移植时注意以下 4 点：

A:确保 IO 是输入输出方式，且必须设置成数字 IO;

B:注意与使用的 IO 相关的寄存器设置，尤其是带外部中断、

带 AD 功能的 IO，相关寄存器一定要设置好;

C:调试时先写配置字，然后控制数据收发

D:注意工作模式切换时间

我们的承诺

最后，欢迎您使用我们的产品，在应用中有技术问题请及时向我们联系，我们会予以技术知道，同时运输中出现产品问题我们会全面责任并予以更换。

愿与您一起走向成功

联系电话：13704018223 陈 工 E-mail: chj_006@sina.com
在线咨询：QQ:35625400 474882985 MSN:1188mm88@hotmail.com