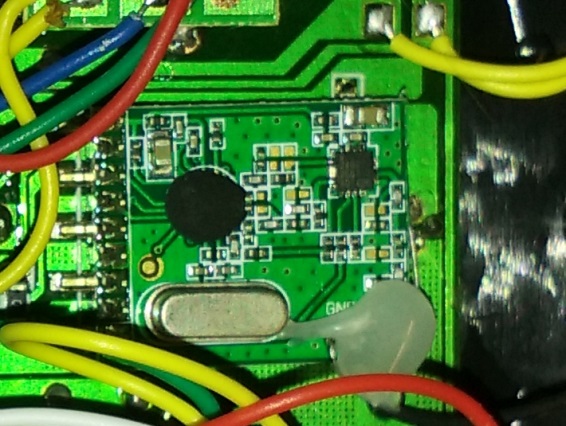
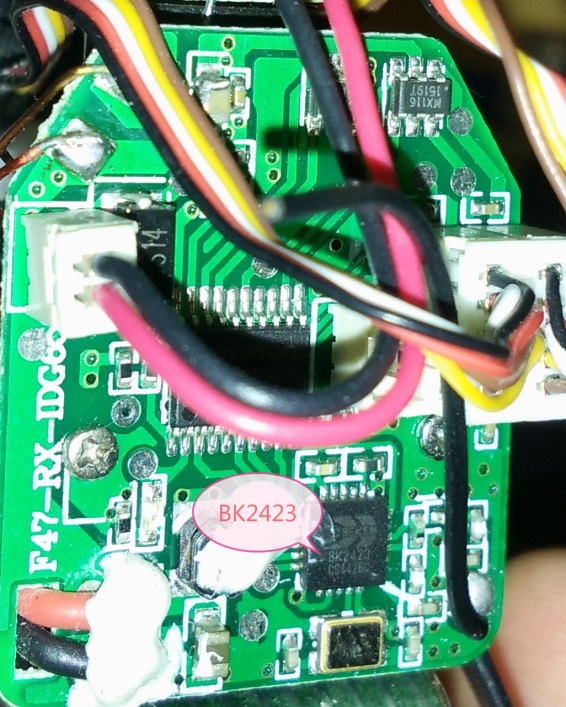
**2.4G无线遥控通讯协议破解—美嘉欣**

2.4G无线遥控通讯协议破解—美嘉欣/nRF24L01+/BK2423

准备做平衡车，缺个遥控器，又不想用手机App。刚好手中有个之前买的遥控直升机，配套的2.4G遥控器看着还行，就打算拿来用在平衡车上。于是就开始了破解。先看图吧！

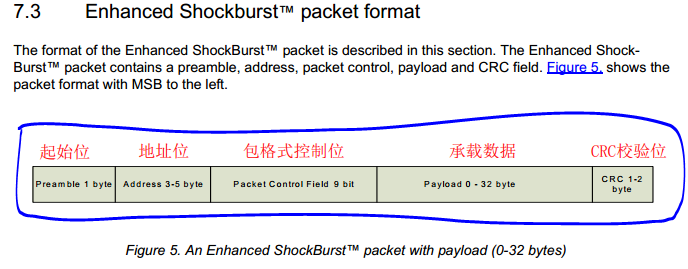


经拆解发现（遥控器端2.4G模块是COB封装的看不出，只能在接收端看芯片型号）为BK2423(和nRF24L01兼容)。于是就可以nRF24L01为接收机接收遥控器数据。当然反过来，也可以用nRF24L01来遥控直升机。

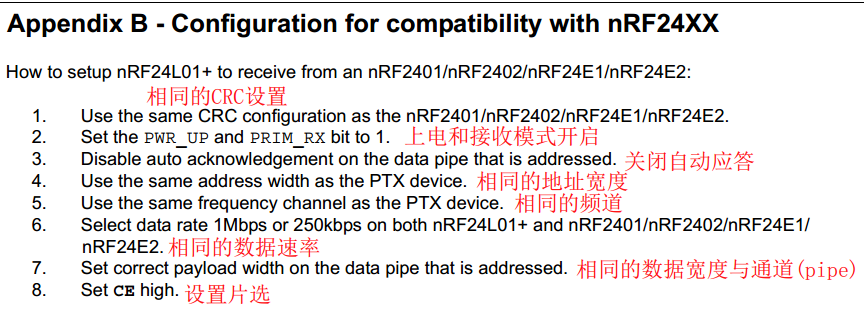


**查数据手册：（这里用nRF24L01数据手册、BK2423多了一个BANK寄存器但是只有初始化时使用、其余通用）（BK2423的数据手册做的不好，毫无美感，看地眼痛心累）**

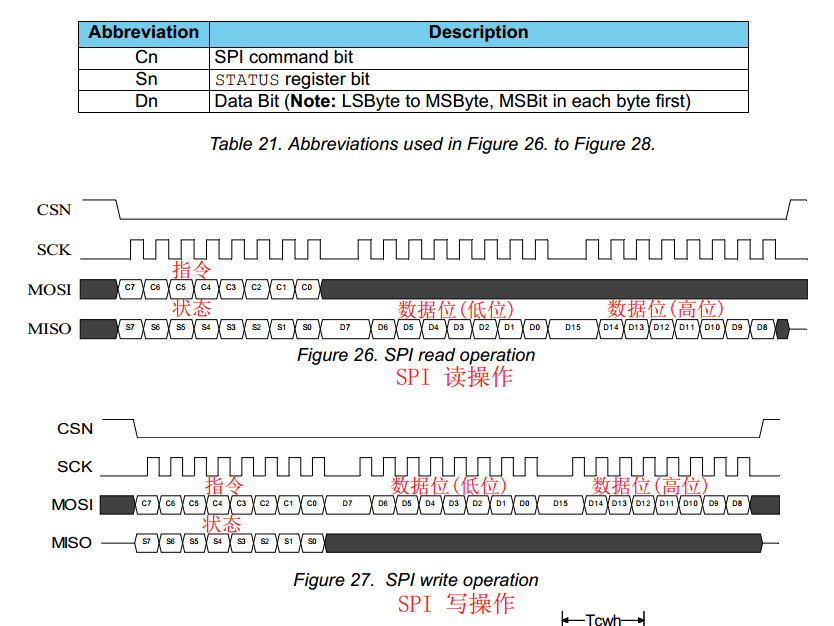
1. 数据格式



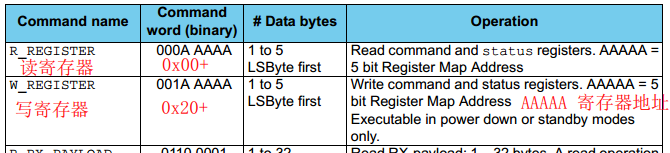
1. 兼容模式（如何配置接收端）



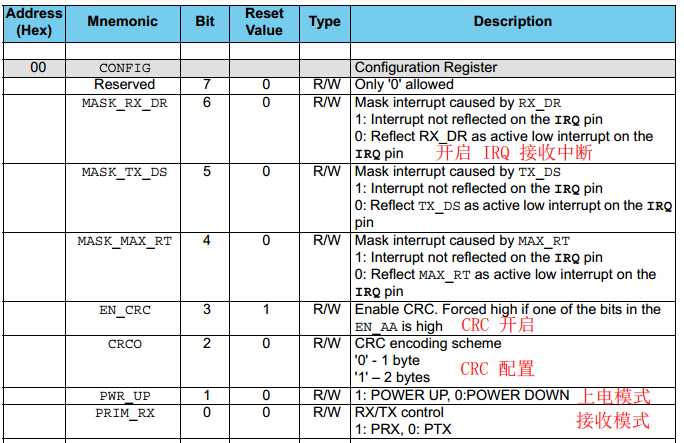
1. SPI时序（看懂时序图）

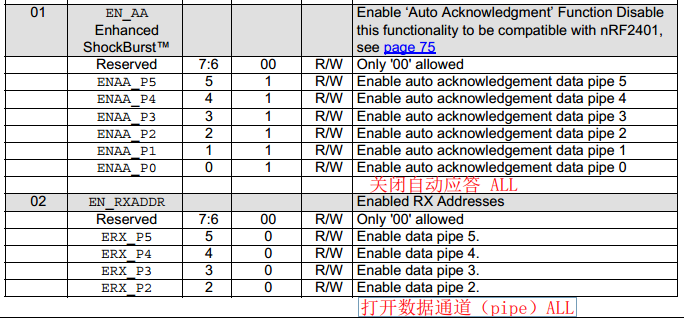


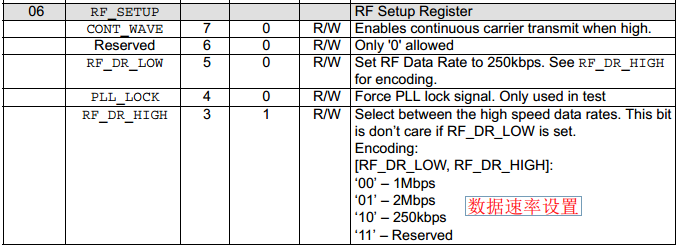
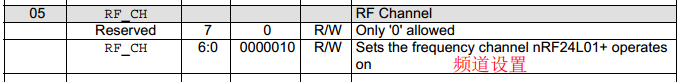
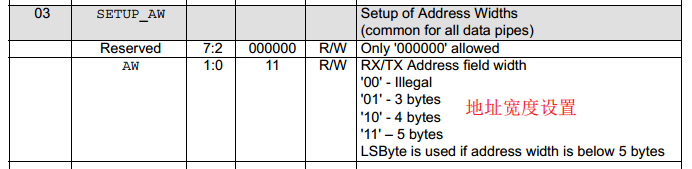
1. 指令（根据3看4、5）

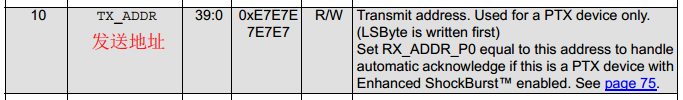
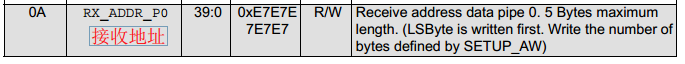


1. 寄存器（部分）









**抓数据：通过遥控器的2.4G模块SPI接口获取配置信息，通讯协议跳频列表。**

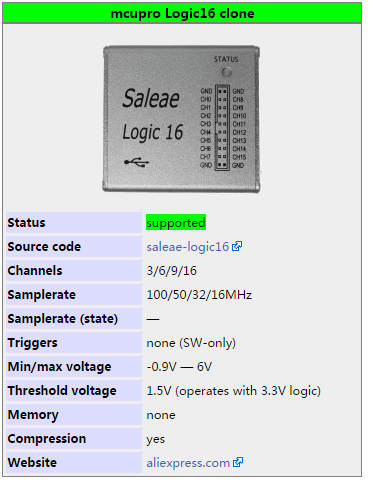
**工具：** Saleae Logic 16 Clone（逻辑分析仪）、

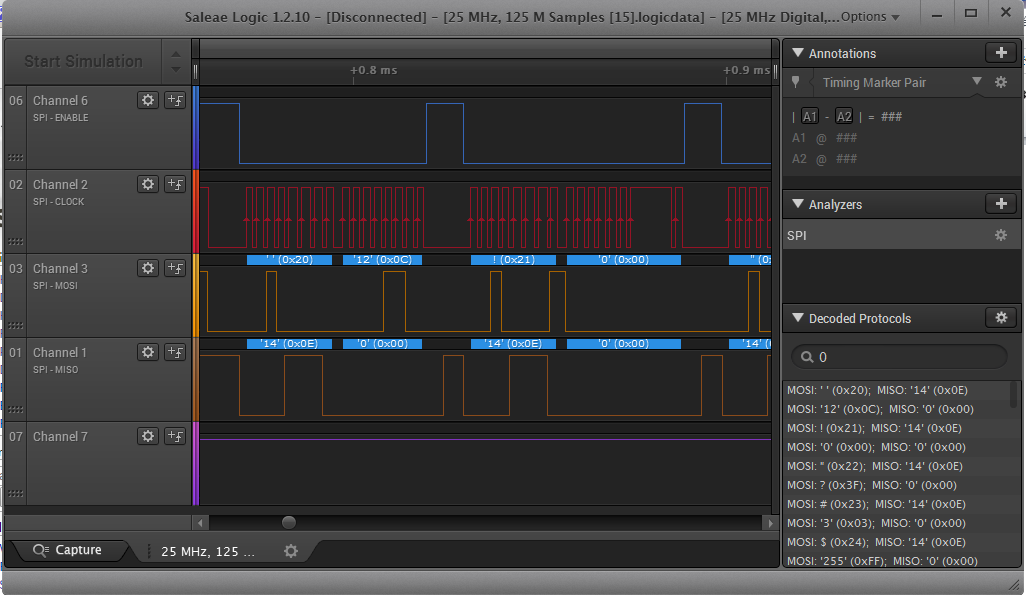
Saleae Logic 1.2.10（官方软件）、

Sigrok PulseView（开源软件）。

（逻辑分析仪这个不是必须的，例：Arduino SPI 来捕获数据，就是在确定引脚的时候会有点麻烦，因为发送端模块的引脚定义不明，但是可以通过单片机新唐N79E814AT20的SPI引脚来查,遥控器的主控用的这个单片机）

下面是硬件和软件的截图：

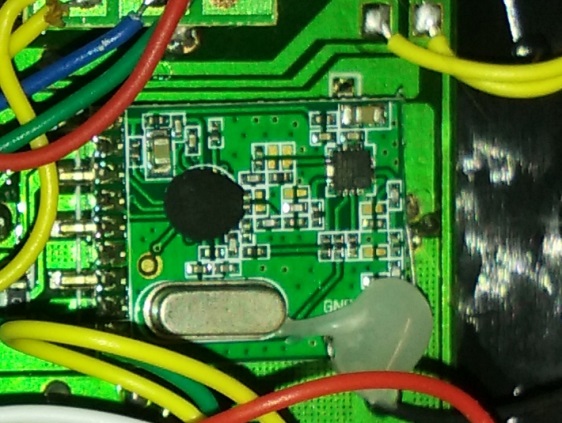
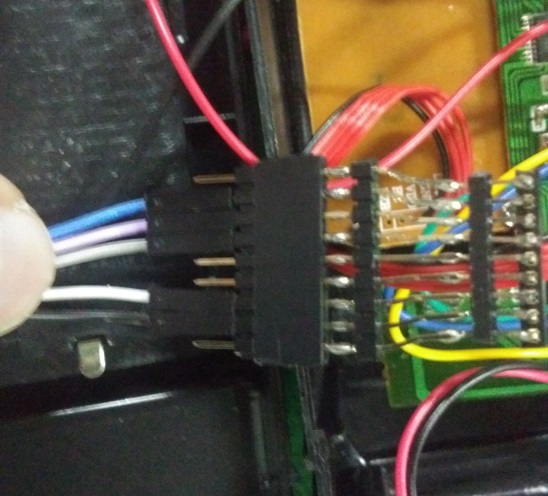






**开整：**

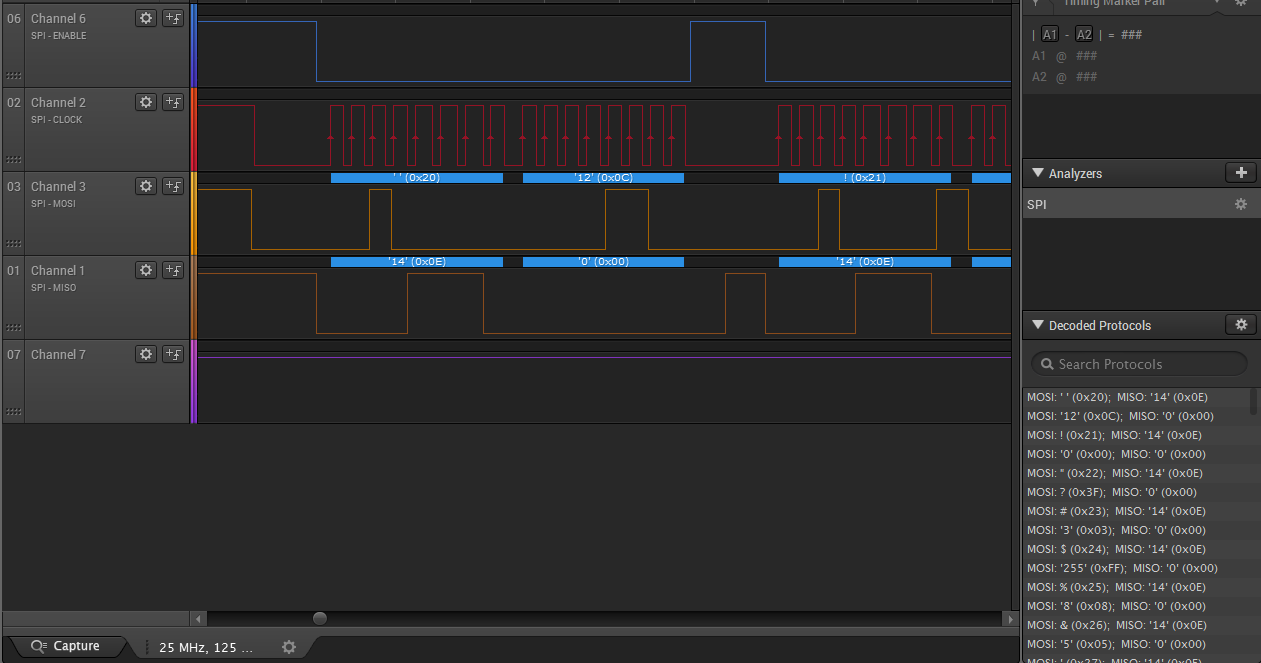
1. **连线：（做了个接头直接卡在模块的引脚上，共9个引脚）**

****

1. **连接逻辑分析仪、打开软件开始捕获、打开遥控器电源、开始配对、配对成功、操控摇杆一段时间、终止。以25MHz的采样率采集15s，九个引脚对应九个通道。（把开机配置、对频、工作跳频的整个过程都采集下来进行后续的分析）**
2. **根据SPI时序图找出MOSI、MISO、CS、CLK。**

****

1. **用SPI协议进行分析**

****

1. **查找关键数据：**

**形式： 指令：写指令(0x20) +寄存器地址(0xXX)**

**数据：D7、D6、D5、D4、D3、D2、D1、D0、**

**指令：**

**W\_TX\_PAYLOAD: 0xA0**

**CRC ：** **0x20 + 0x00 = 0x20**

**AutoAck: 0x20 + 0x01 = 0x21**

**地址宽度 : 0x20 + 0x03 = 0x23**

**频道： 0x20 + 0x05 = 0x25**

**数据速率： 0x20 + 0x06 = 0x26**

**数据位宽度： 没有查到，可通过0xA0查找W\_TX\_PAYLOAD，**

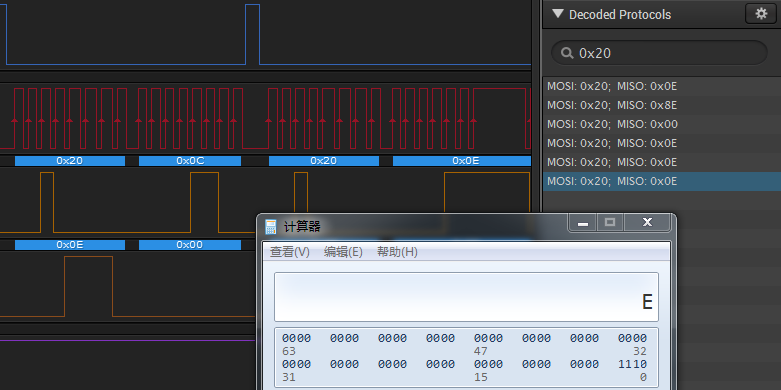
**查看入数据的字节数**

**发送地址： 0x20 + 0x10 = 0x30**

**数据：**

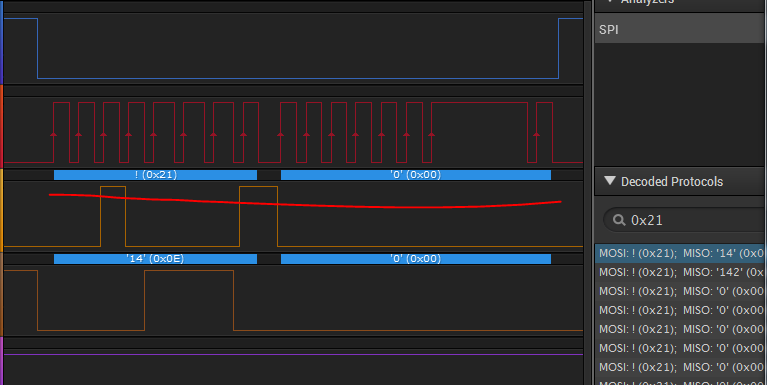
**CRC ：0x0E : 0000 1110**

1. **开启CRC**
2. **2 Bytes**
3. **PWR\_UP**

****

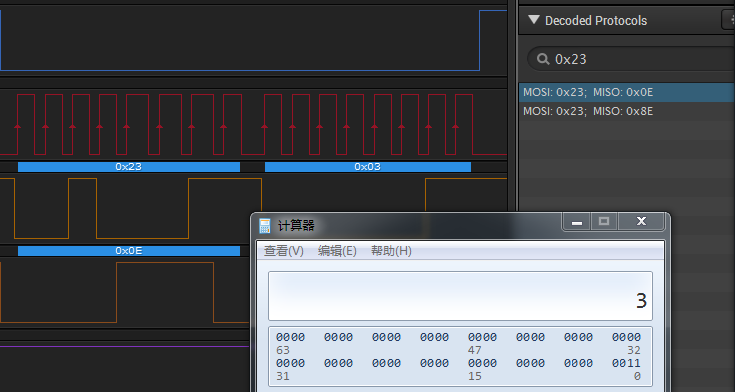
**AutoAck: 0x00 : 0000 0000**

1. **关闭所有**

****

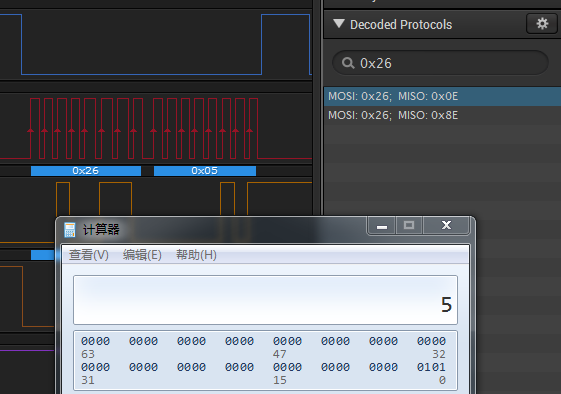
**地址宽度：0x03 : 0000 0011**

1. **5 Bytes**

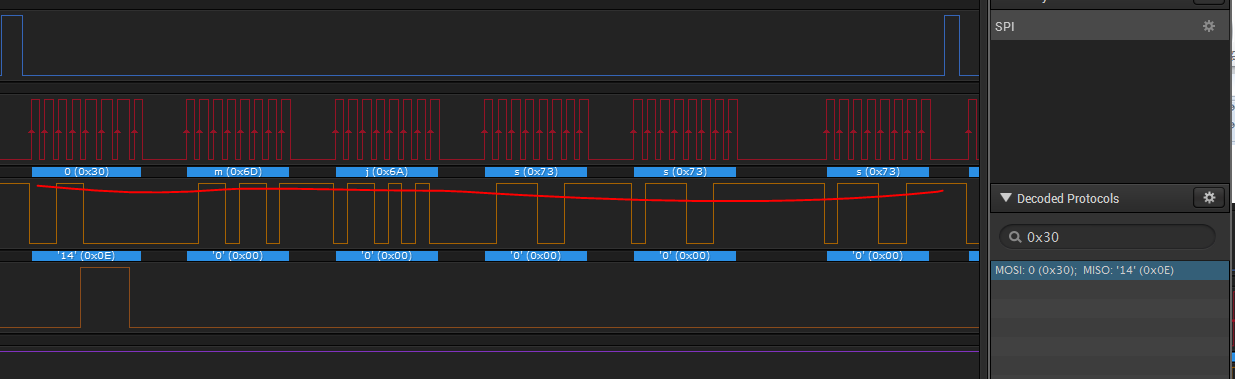
****

**数据速率：0x05 : 0000 0101**

1. **1Mbps**

****

**发送地址：0x6D 0x6A 0x73 0x73 0x73 : “mjsss”**

****

**注：右下角的Decoded Protocol 列表中配置命令有多个，如有两个则**

**下一个是BK2423独有的寄存器配置。多个有部分为数据。**

**频道：经检查发现有两个跳频列表**

1. **配对未完成时：（至今也没搞明白这个是干什么的，猜测是对频的部分，但是不清楚对频的方式）**

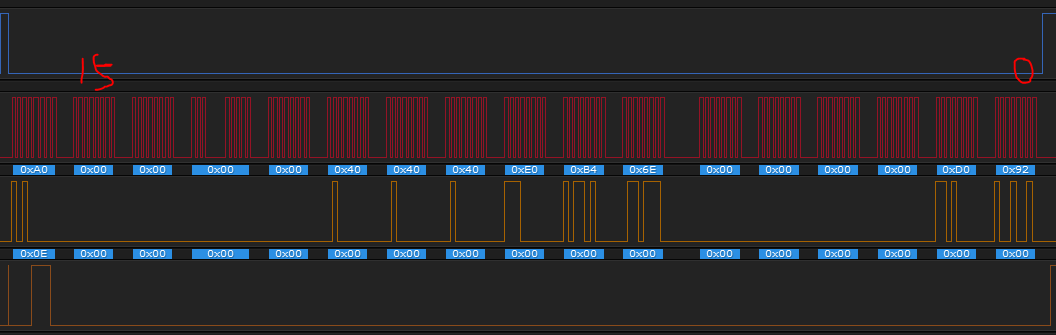


1. **配对成功后：**



**Cycle列表每个频道发送两次，循环管往复。**

**发送数据位宽度由W\_TX\_PAYLOAD: 0xA0 发现：16 Bytes**

****

**所有必须的信息已经知道了：接下来开始接收数据进行验证！**

**Arduino 程序 （RF24 Lib）**

**/\***

**\* 美嘉欣 2.4G 无线遥控接收程序**

**\* 作者：anchang**

**\* RX\_Address：mjsss**

**\* RC\_ChannelList: 0x25, 0x2A, 0x32, 0x2C, 0x14, 0x27, 0x36, 0x34,**

**\* 0x1C, 0x17, 0x11, 0x1A, 0x35, 0x24, 0x28, 0x18；**

**\* CRC: 2 Bytes**

**\* AutoAck: Disable**

**\* DataRate: 1MBPS**

**\* PayloadSize: 16 Bytes**

**\***

**\*/**

**#include <SPI.h>**

**#include <nRF24L01.h>**

**#include <printf.h>**

**#include <RF24.h>**

**#include <RF24\_config.h>**

**RF24 radio(7, 8);**

**// RX\_Address**

**const byte rxAddr[] = "mjsss";**

**// RX\_ChannelList**

**const byte rxChannelList[] = {0x25, 0x2A, 0x32, 0x2C, 0x14, 0x27, 0x36, 0x34,**

**0x1C, 0x17, 0x11, 0x1A, 0x35, 0x24, 0x28, 0x18};**

**void setup()**

**{**

**while (!Serial);**

**Serial.begin(250000);**

**printf\_begin();**

**// Receive Interrupt**

**attachInterrupt(digitalPinToInterrupt(2), Receiver, FALLING);**

**radio.begin();**

**radio.setCRCLength(RF24\_CRC\_16); // CRC-2 bytes**

**radio.powerUp(); // PWR\_UP & PRIM\_RX**

**radio.setAutoAck(false); // Disable auto Acknowlage**

**radio.setChannel(0x25); // channel - 0x25(start Channel)**

**radio.setDataRate(RF24\_1MBPS); // date rate - 1M**

**radio.setPayloadSize(16); // payload width - 16**

**radio.openReadingPipe(0, rxAddr); // open read pip**

**radio.startListening(); // start listening**

**radio.printDetails(); // print Module Setiing Details**

**}**

**void loop()**

**{**

**}**

**void Receiver()**

**{**

**static char times = 0;**

**static byte nextChannel = 0;**

**if(radio.available())**

**{**

**byte text[16] = {0};**

**radio.read(&text, sizeof(text));**

**for(int i = 0; i < 16; i++)**

**{**

**Serial.print(text[i], HEX);**

**Serial.print('\t');**

**}**

**Serial.print('\n');**

**times++;**

**}**

**else**

**{**

**times = 0;**

**nextChannel = 0;**

**}**

**if(times >= 1)**

**{**

**times = 0;**

**nextChannel++;**

**if(nextChannel == sizeof(rxChannelList))**

**{**

**nextChannel = 0;**

**}**

**// radio stop**

**radio.stopListening();**

**// change channel**

**radio.setChannel(rxChannelList[nextChannel]);**

**// radio start**

**radio.startListening();**

**}**

**}**

**由于不清楚对频过程所以可以不进行对频（漏洞：在对频过程中只要摇动任意摇杆就会打断对频直接进入工作状态）**

**连接硬件：**

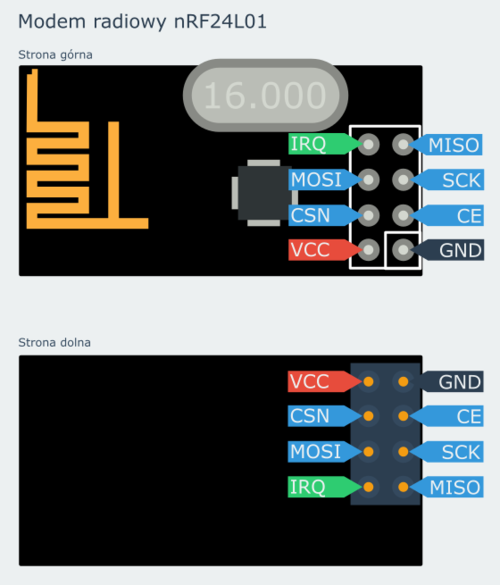
**VCC —>3.3V**

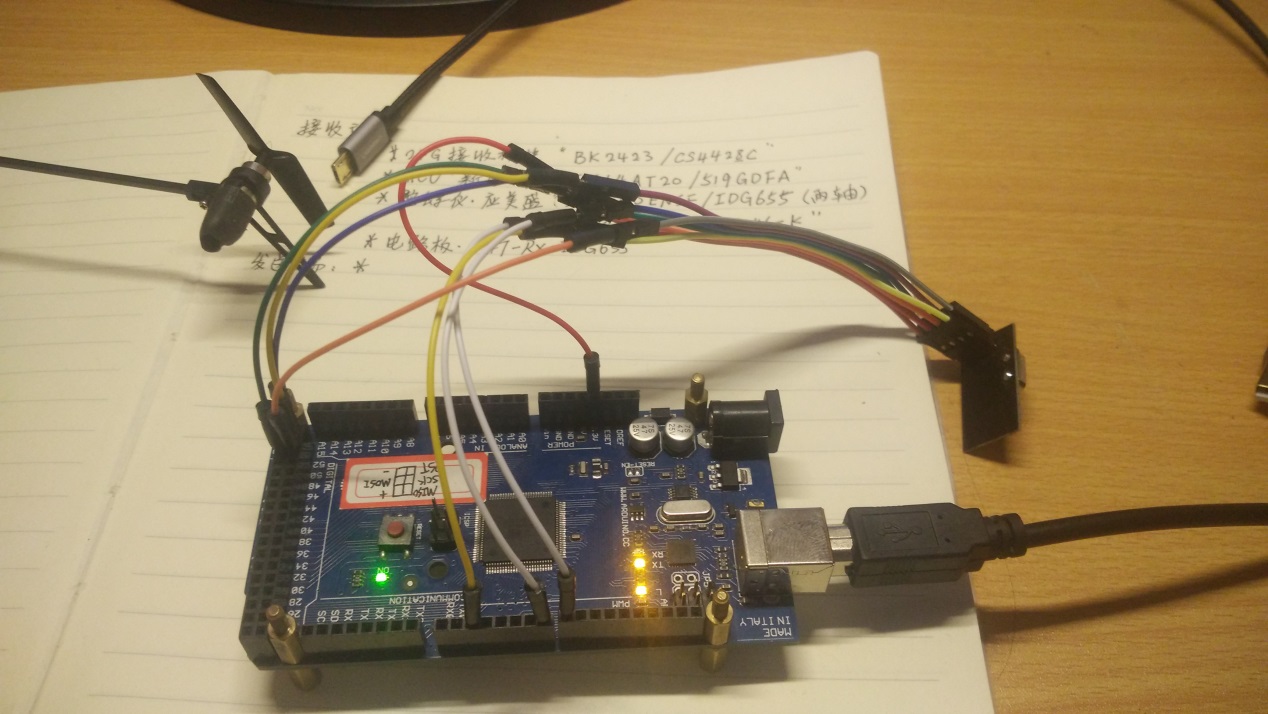
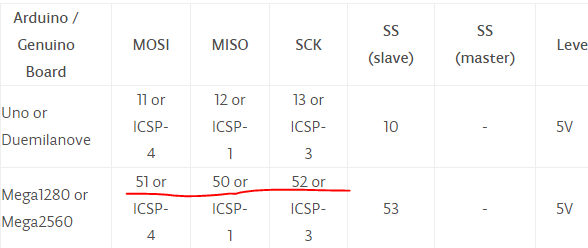
**GND —>GND**

**CE —>Pin 7**

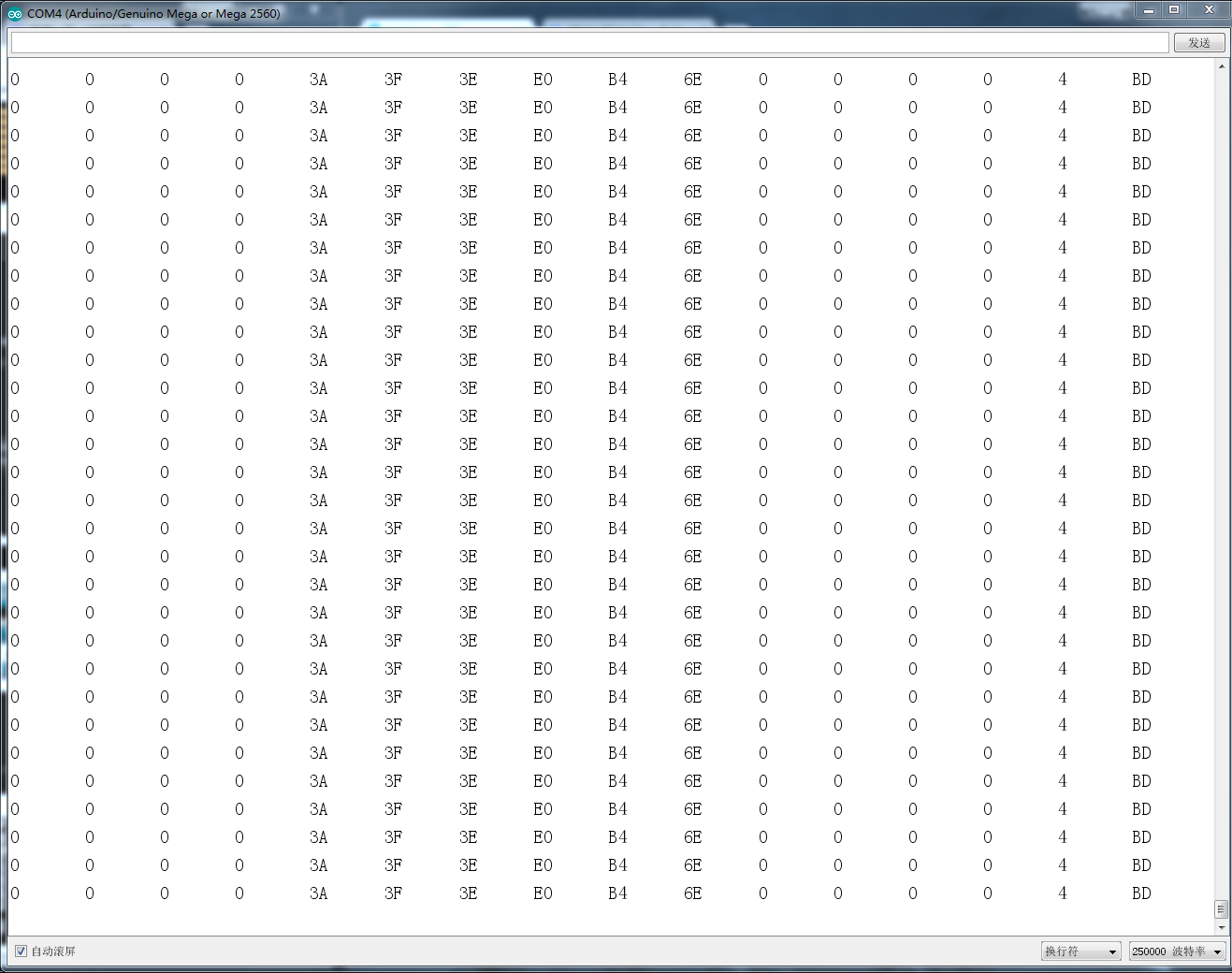
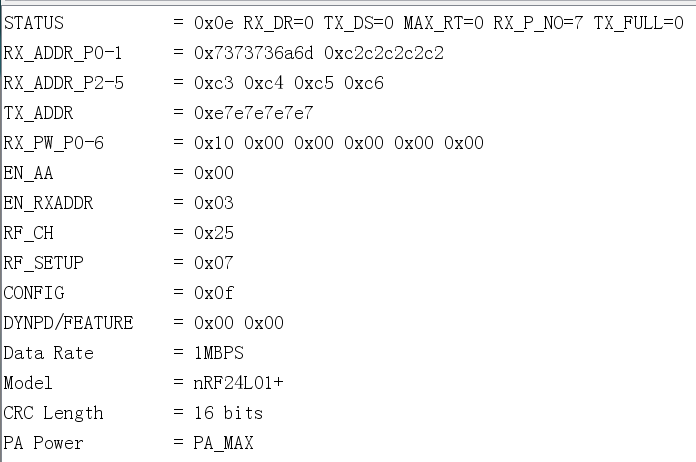
**CSN —>Pin 8**

**IRQ —>Pin 2**

****

****

**开始接收数据！**

****

**自左向右依次：0：油门、1：左右转、2：前后飞、3：左右飞、4：左右转微调、5：前后飞微调、6：左右飞微调、14：操作模式/灵敏度、15：校验位。（其余数据含义未知）**

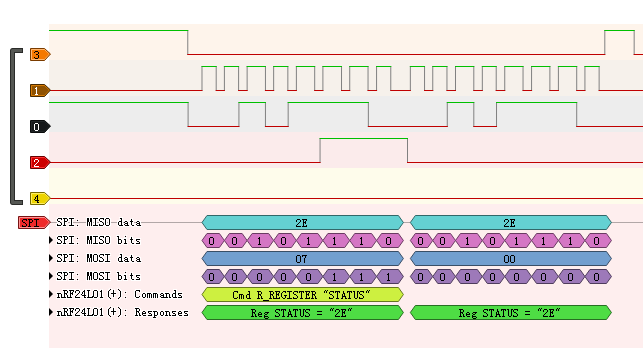
**至此全部工作已经完成。**

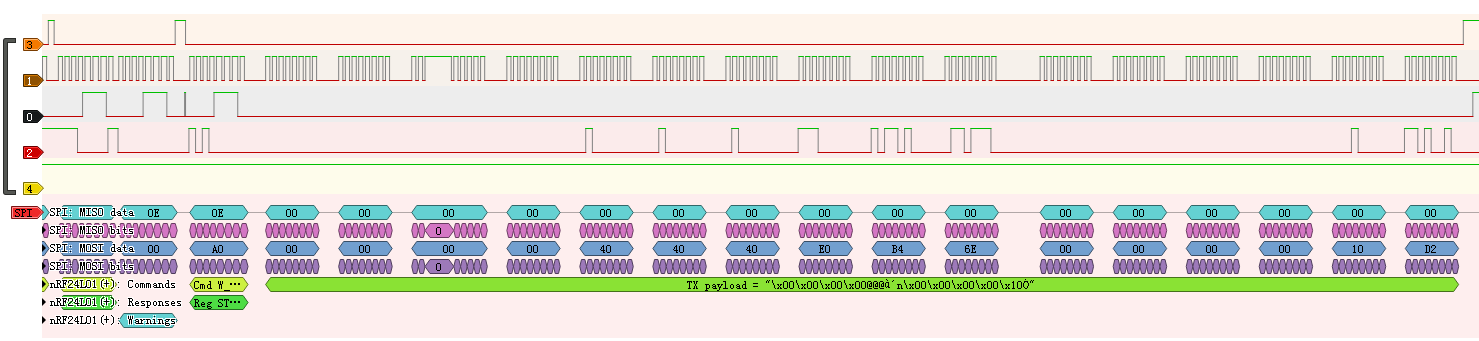
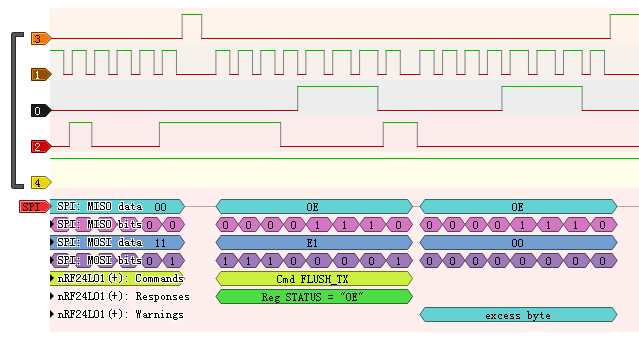
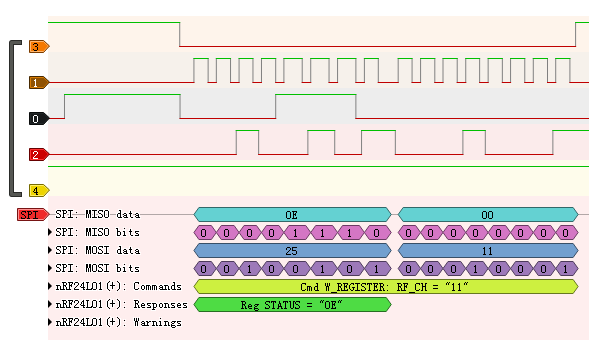
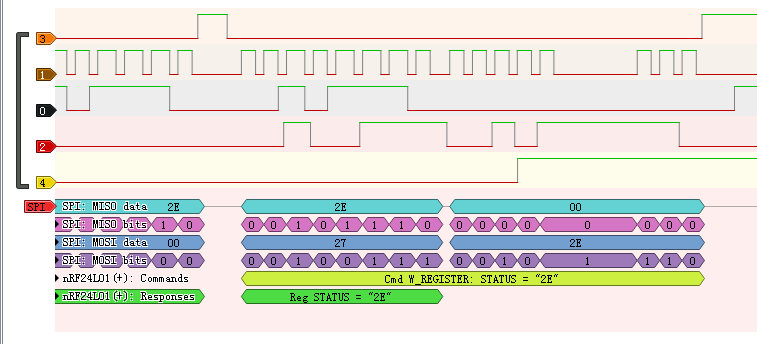
**总的做下来难道并不高，但是锻炼了自己。把它写下来一方面是总结一下自己所做的工作、步骤，另一方面希望能够对大家有所帮助。**

**附录：我在上文中提到了sigrok这个开源软件，但是并未提及（因为发现越写思路越清晰，就用不到了），这里作为补充。**

Sigrok是一个开源跨平台信号分析软件、支持大量设备。它不仅能够提供通讯协议分析、更可以具体到具体的芯片，比如这里用到的nRF24L01+,它以SPI作为通讯接口，那么就可以应用SPI分析数据内容，进而解析其含义。

这里以一个数据传输过程为例：

****

****

sigrok支持大量的通讯协议和芯片解码，有需要的可以了解一下。

注：暂不支持本文中的设备，折腾了很久，从win7到Ubuntu都试过了就是不行。我是通过官方软件导出为Raw Binary的形式再导入sigrok进行分析的，这个是最快捷的方法。 it依次为0~7Channel，在导入时选择的通道数n是指导入0~(n-1)。另外要填写采样率，单位为Hz。