

## OM6220 做玩具小车应用说明

**V1.1**

**2022/07/28**

## 修改历史

版本	历史版本	日期	作者
V1.0	初始版本	2022-06-28	钟 科
V1.1	增加了 vco 幅度值设置说明	2022-07-28	钟 科

OnMicro Confidential

## 目录

修改历史.....	1
图表目录.....	3
表格目录.....	4
1. 概述.....	5
2. OM6220 的初始化简介.....	6
2.1 初始化介绍.....	6
2.2 应用配置介绍.....	7
2.3 关于切换 BANK 的操作.....	8
3. OM6220 发送端介绍.....	9
3.1 发送端时序等介绍.....	9
3.2 发送端正确操作流程.....	9
4. OM6220 接收端操作介绍.....	10
4.1 接收端正确操作流程.....	10
4.2 接收端正常跳频介绍.....	10
4.3 接收端丢包之后跳频介绍.....	11
4.4 接收端收到一包数据之后的操作.....	12
4.5 接收到第一包数据的处理.....	12
4.6 长时间收不到数据的处理.....	13
4.7 接收要经常拉 CE.....	13
4.8 接收端收不到数据的时候马达操作.....	14
5. OM6220 在操作是常出现的一些问题.....	14
5.1 关于拉 CE 的操作.....	14
5.2 修改晶体负载电容值.....	14
5.3 关于通信地址的问题.....	15
5.4 关于通信频点的问题.....	15
5.5 关于通信不上或者通信效果不好的问题.....	15
5.6 关于接收数据包长度不对的问题.....	16
5.7 关于跳频的问题.....	16
5.8 关于 3 线 SPI 的 SDA 线做输入输出功能说明 .....	16

## 图表目录

图表 1 OM6220 初始化部分代码.....	7
图表 2 OM6220 初始化部分时序图.....	7
图表 3 初始化完成之后根据用户的应用需要进行配置.....	8
图表 4 关于切换 BANK 的正确操作.....	8
图表 5 发送端跳频和发送间隔.....	9
图表 6 发包的正确操作流程.....	9
图表 7 接收端的正确操作流程.....	10
图表 8 接收端正常跳频.....	11
图表 9 接收端丢了 2 包之后后面采取快跳的方式去追发送端.....	11
图表 10 同步跳频时序错不开的情况.....	12
图表 11 收到第一包数据之后设置 VCO 为最高档.....	13
图表 12 接收端长时间收不到数据的处理.....	13

## 表格目录

表 1-1 xxxx.....	6
表 1-2 xxxx.....	6

OnMicro Confidential

## 1. 概述

本文主要介绍 OM6220 在玩具小车应用上面的一些使用技巧和注意事项，避免客户在使用过程中重复之前出现的问题，加快客户的开发进度和提高产品性能。其中文中提到的这些点，用户一定要特别注意，要认真阅读本文。本文会结合程序还有时序图为大家讲解，这样能让开发者有直观的感受，更能理解里面说讲的内容。

OnMicro Confidential

## 2. OM6220 的初始化简介

### 2.1 初始化介绍

OM6220 的初始化过程，建议客户按照我们给的流程初始化，且一定要注意是否校准完成；必须要加超时没有查到校准标志要重新复位初始化，从 CE 拉高开始到校准完成一般需要 30ms 左右的时间。如果还有 IRQ 脚特殊应用的，初始化会一点改动，主要是 BANK0 的 DYNPD 这个寄存器。下图有详细的注释说明。

```
void HS6220_Init(void)
{
    unsigned char temp[3];
    RE_INIT_RF:
    SPI_write_byte(HS6220_BANK0_FEATURE, SOFT_RST); // 软件复位RF

    /* 默认是3线SPI的，如果要使用4线SPI，则上电之后要设置一下DYNPD这个寄存器，
    这个寄存器的bit3是控制SPI线数的，不管在哪里用到这个DYNPD寄存器一定要注意这点。*/
    #if (RF_SPI_NWIRE == SPI_4_WIRE)
    // 如果要用6220的IRQ脚控制马达，要设置DYNPD的bit7,6，并且config寄存器的bit6~4不能设置将RF收发中断信号映射到IRQ脚
    SPI_write_byte(HS6220_BANK0_DYNPD, 0x08); // 0x08 | 0x40; //
    #else
    // 如果要用6220的IRQ脚控制马达，要设置DYNPD的bit7,6，并且config寄存器的bit6~4不能设置将RF收发中断信号映射到IRQ脚
    SPI_write_byte(HS6220_BANK0_DYNPD, 0x00); // 0x00 | 0x40; //
    #endif

    HS6220_CE_Low();
    SPI_write_byte(HS6220_BANK0_CONFIG, 0x8b); // 给芯片上电
    delay_ms(20); // 因为后面单端晶体的芯片，如果晶体的负载电容大则需要起振时间长，建议20ms的时间
    SPI_write_byte(HS6220_BANK0_PMU_CTL, 0xac); // 设置芯片处于工作模式
    delay_ms(2); // 上面这2个时间不要少。

    HS6220_Bank_Switch(HS6220_Bank1); // 切换到BANK1
    temp[0] = 0x01; // 设置VCO幅度值
    SPI_wr_buffer(HS6220_BANK1_FAGC_CTRL_1, temp, 1);

    HS6220_Bank_Switch(HS6220_Bank0); // 切换回BANK0
    // 脉冲方式激活内部校准RF
    HS6220_CE_High();
    delay_us(100);
    HS6220_CE_Low();

    SysTick->VAL = 0; // 清零定时器当前值，
    tim2_lms_cnt = 0; // 定时计数器清零

    while(1)
    {
        /*从CE拉高到查到校准完成标志需要29ms左右，一定要注意这个时间。如果没有校准完成就出去，那后面RF工作的性能不是最佳的*/
        // 如果32ms还没有查到校准完成标志，则重新开始复位初始化，避免因为SPI通信失败导致一直查询不到校准标志。
        if (tim2_lms_cnt > 31)
        {
            goto RE_INIT_RF;
        }
        if ((SPI_read_byte(HS6220_BANK0_RF_SETUP) & 0x20) != 0x00) // 看看是否查到校准标志
        {
            break; // 如果校准完成，则继续往下执行
        }

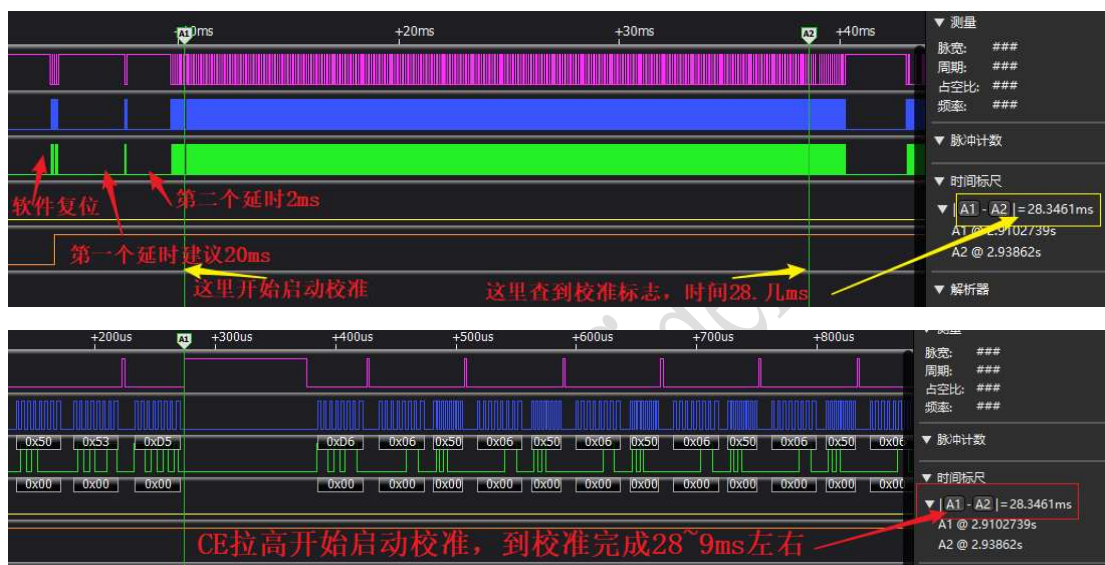
        delay_ms(2); // 可以不用那么频繁的去读取标志，这句是可以不要的。
    }
    SPI_write_byte(HS6220_BANK0_RF_SETUP, 0x40); // 清除校准完成位，写bit6为1是清除校准标志

    HS6220_Bank_Switch(HS6220_Bank1); // 切换到BANK1
    temp[2] = 0x75;
    temp[1] = 0x98;
    temp[0] = 0x20;
    SPI_wr_buffer(HS6220_BANK1_CAL_CTL, temp, 3);

    HS6220_Bank_Switch(HS6220_Bank0); // 注意一定要切换回BANK0，如果不在BANK0，则不能收发数据
}
```

如果要过认证，这里写0x01，  
如果不需要过认证，这里写0x0C

图表 1 OM6220 初始化部分代码



图表 2 OM6220 初始化部分时序图

## 2.2 应用配置介绍

初始化完成之后,就是用户的应用配置,根据用户需要自己配置。下图也有详细的注释。



```

/* 初始化 hs6220 */
HS6220_Init();

/* 用户应用配置*/
#if (RF_SPI_NWIRE == SPI_4_WIRE)
    SPI_write_byte(HS6220_BANK0_DYNPD, 0x08);
#else
    SPI_write_byte(HS6220_BANK0_DYNPD, 0x00);
#endif
SPI_write_byte(HS6220_BANK0_FEATURE, 0x10); // 不需要回ack, 不需要动态数据包长度
SPI_write_byte(HS6220_BANK0_EN_AA, 0x00); // 不需要自动应答
SPI_write_byte(HS6220_BANK0_EN_RXADDR, 0x09); // bit3:为0则关闭白化功能, 1为打开白化功能, bit2~0:对应3个接收pipe
SPI_write_byte(HS6220_BANK0_SETUP_RETR, 0x00); // 不需要开启自动重发 rx_addr的bit3为1是开白化功能
SPI_write_byte(HS6220_BANK0_CONFIG, 0x8a); // 发送模式
SPI_write_byte(HS6220_BANK0_RX_PW_P0, sizeof(pt_struct)); // 设置通信数据包的长度。
ch_cnt = 0;
SPI_write_byte(HS6220_BANK0_RF_CH, rf_comm_ch[ch_cnt]); // 设置通信信道
HS6220_Change_Addr(rf_comm_addr, 5); // 设置通信地址
SPI_write_byte(HS6220_BANK0_RF_SETUP, 0x47); // 设置发射功率, 0x47=8dB, 最大8dB

HS6220_Flush_Tx();
HS6220_Flush_Rx();
HS6220_Clear_All_Irq();

/* 测载波看频偏的时候频谱仪的SPAN值要先设置100M抓到载波在哪个点之后再调整成1M看看大概偏多少。
最后设置到200k左右看频偏更准确。*/
// HS6220_ModeSwitch(HS6220_Carrier_Mode); // 设置载波模式
// HS6220_Change_Xtalcc(15); // 修改晶体的负载电容值来调整频偏, 数值从0~31, 数值越大频率越低, 数值越小频率越高。
// HS6220_CE_High(); // CE拉高才能发出载波
// while(1);

```

图表 3 初始化完成之后根据用户的应用需要进行配置

## 2.3 关于切换 BANK 的操作

关于切换 BANK 这个事情, 很多用户习惯直接切换 BANK, 而不是先读取状态寄存器看看当前是处于哪一个 BANK, 然后再判断是否需要切换 BANK, 这样虽然能省一点代码, 但是如果芯片由于掉电不完全, 或者其他异常导致 BANK 并不是你认为的那个 BANK, 就会出现切错的情况, 从而导致后面数据收发不了, 这点强烈建议一定要先读再判断要不要切换 BANK。

```

// 切换OM6220 BANK
void HS6220_Bank_Switch(HS6220_Bank_TypeDef bank)
{
    unsigned char sta;

    sta = SPI_read_byte(HS6220_BANK0_STATUS); // 1. 先读取状态寄存器

    if(bank != HS6220_Bank0) // 2. 再判断要不要切换BANK
    {
        if(!(sta & HS6220_Bank1))
        {
            HS6220_wr_cmd(HS6220_ACTIVATE, HS6220_ACTIVATE_DATA);
        }
    }
    else
    {
        if(sta & HS6220_Bank1)
        {
            HS6220_wr_cmd(HS6220_ACTIVATE, HS6220_ACTIVATE_DATA);
        }
    }
}

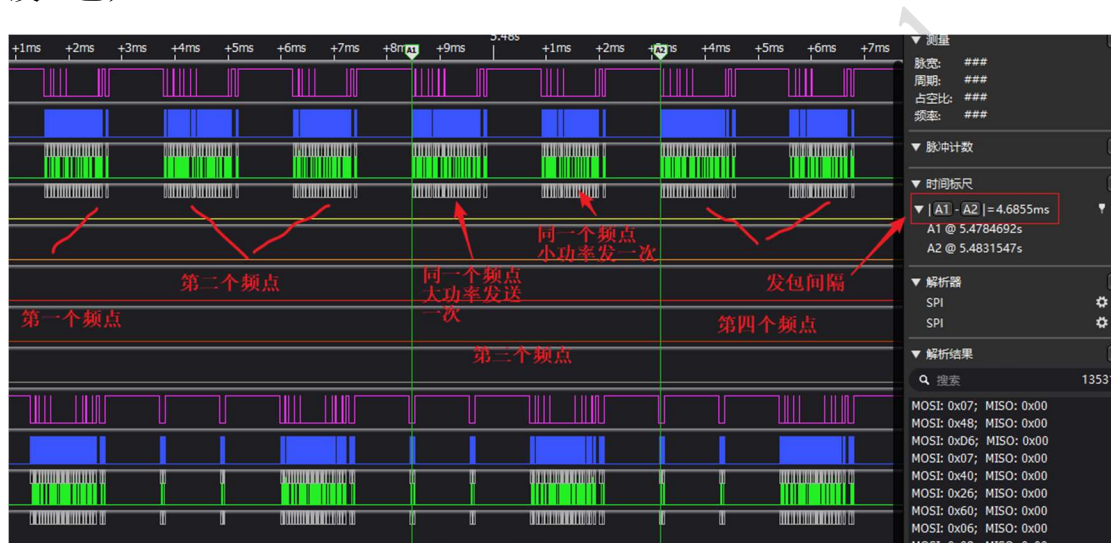
```

图表 4 关于切换 BANK 的正确操作

### 3. OM6220 发送端介绍

#### 3.1 发送端时序等介绍

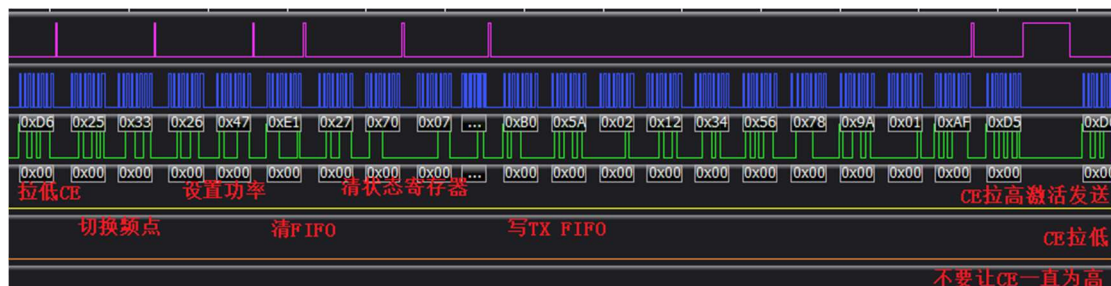
发送端发送间隔建议大于等于 5ms，这个时间接收端可以有比较多的时间做其他处理，接收端的跳频时序也好做一点；建议 4 个频点发送，高中低频点都要有，单双数频点都要有；建议一个频点上大功率挡发一包，等 1ms 再切小功率挡发一包；



图表 5 发送端跳频和发送间隔

#### 3.2 发送端正确操作流程

发包的正确操作流程，CE 拉低，切换频点，切换地址，切换功率，清 FIFO，清状态寄存器，等操作，然后写 TX FIFO，然后 CE 拉高激活发送。

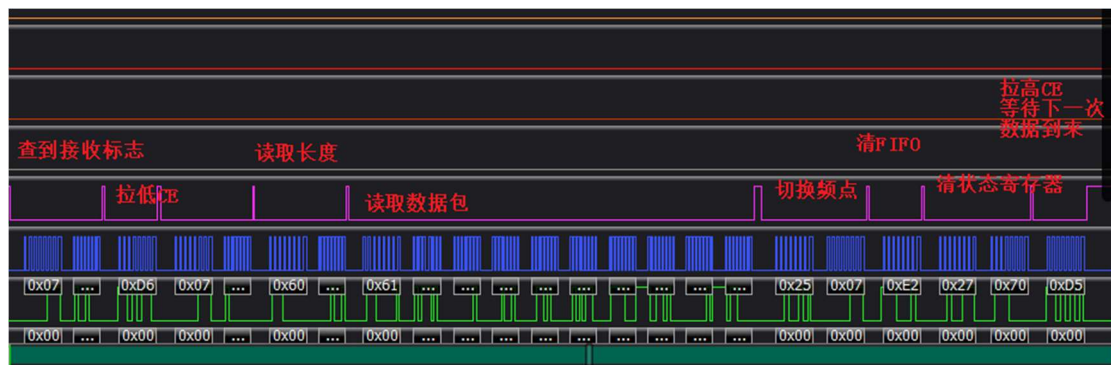


图表 6 发包的正确操作流程

## 4. OM6220 接收端操作介绍

### 4.1 接收端正确操作流程

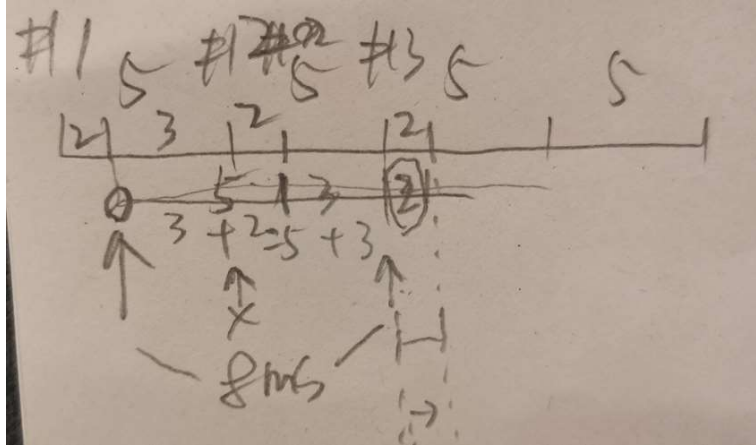
接收端正确操作流程，查到接收标志，拉低 CE，需要判断长度的话可以读取数据包长度，读取数据包内容，清 RX FIFO，清状态寄存器，切换频点，如果需要切换地址就切换，还有其他操作，拉高 CE 等待下一次数据到来。



图表 7 接收端的正确操作流程

### 4.2 接收端正常跳频介绍

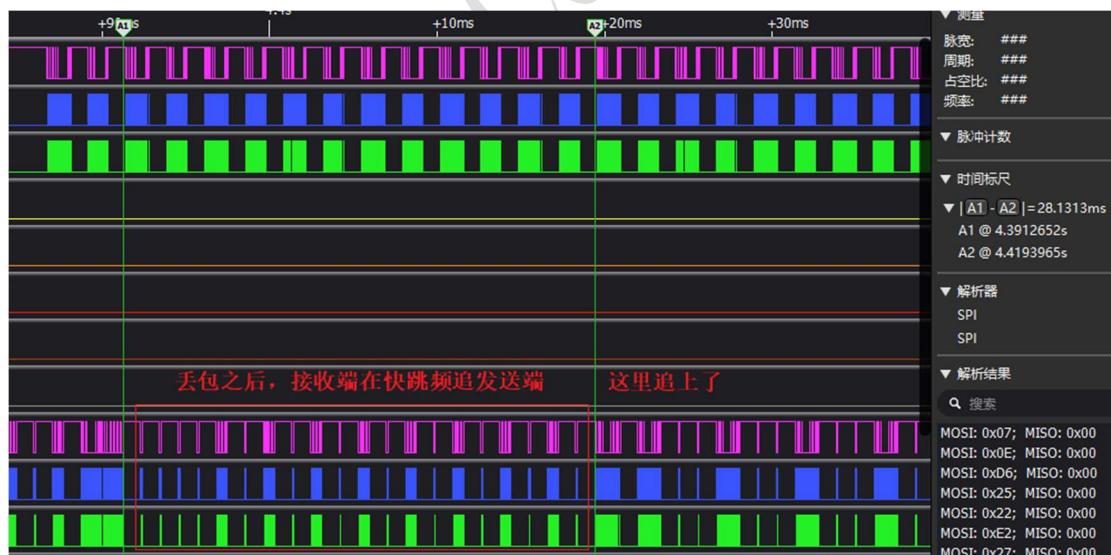
接收到数据之后的处理，接收到一包数据之后，就切换到下一个频点，拉高 CE 等待下一包数据的到来，并且设置超时跳频的时间为 2 个发送间隔的时间在减少 2ms，主要是在本次切换的频点等不到数据之后，在下下个频点的数据到来之前的 2ms 要提前进入下下个频点等到数据到来。举个例子，比如发送周期为 5ms。当前接收到数据的频点为 1 号频点，那接收完之后接收端是马上切换到 2 号频点的，但是此时发送端在 1 号频点刚发完数据，还没有等到下一个周期的到来，比如还差 3ms 才到 2 号频点发送。那设置超时时间为  $5 \times 2 - 2 = 8\text{ms}$ ，如果 2 号频点收不到，则等到 8ms 之后切换到 3 号频点，则此时发送端就刚好准备进入 3 号频点发送，时序就能对上。



图表 8 接收端正常跳频

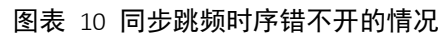
### 4.3 接收端丢包之后跳频介绍

当然，如果 3 号频点还是没有通信上，你可以在等 5-1ms 提前进入 4 号频点等待接收。我例程里面没有这么做，我是错过一个接收包，后面就 2ms 一跳去追发送端。这里一定要注意错开时序。举个例子，如果你是 4ms 一跳，那你快跳的时候不能设置为 2ms 一跳，因为这样有可能会同步一直跳导致要到长时间收不到数据之后软件复位了才错开时序才能对上。也不能 1ms 一跳，因为太快了，数据在发的时候你可能已经切换到下一个频点去了。

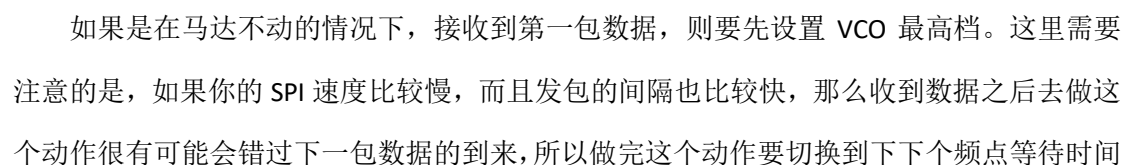


图表 9 接收端丢了 2 包之后后面采取快跳的方式去追发送端

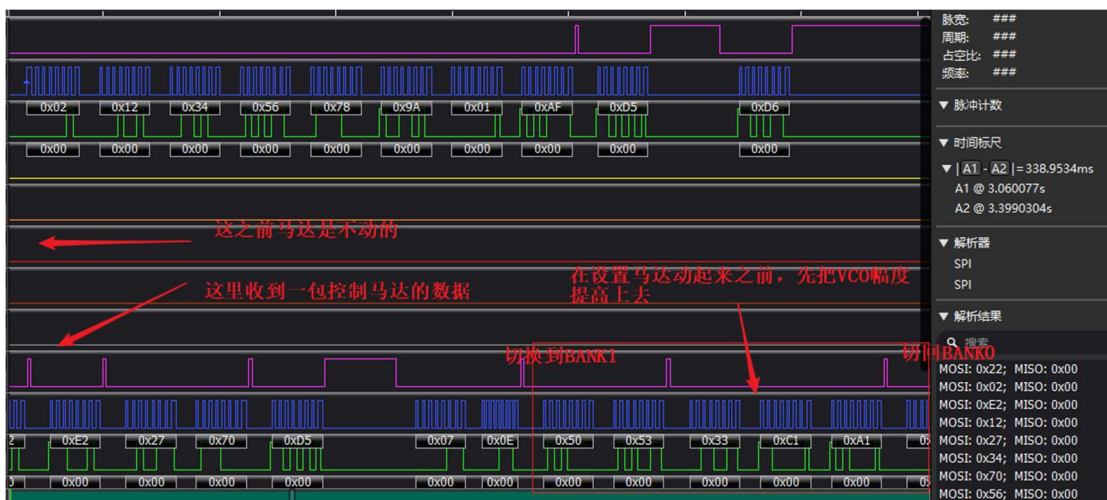




收到一包数据之后，切换频点，清 FIFO，清状态寄存器之后。接收端做一次休眠唤醒，休眠时间 1ms，唤醒时间 1ms。然后拉高 CE 等待下一次数据到来。



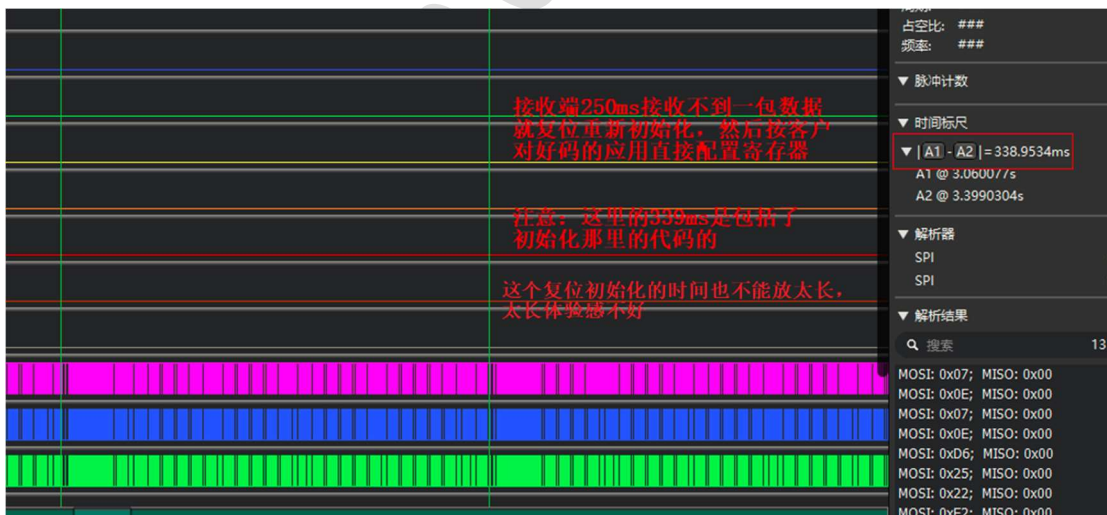
长一点等数据到来，具体需要你抓时序图来看。



图表 11 收到第一包数据之后设置 VCO 为最高档

## 4.6 长时间收不到数据的处理

假设现在设置的是 250ms 超时收不到一包数据，则马达停止转动，RF 复位初始化，然后根据对好码的应用配置配置好，快跳频接收数据。复位初始化 VCO 幅度按第一档设置，后面马达需要再次动的时候再设置 VCO 最高档。这里注意为什么是 250ms，因为一个字节是 8bit 为 255，所以定时器计数值可以使用 1 个字节计数就够用了。



图表 12 接收端长时间收不到数据的处理

## 4.7 接收要经常拉 CE

在切换频点的时候会先拉低 CE，然后切换完频点之后再拉高 CE 等待数据到来。如果是

只有 1 个频点进行通信的情况，正常收到数据的时候因为是先拉低了 CE 再去读取数据的，所以已经有这个动作了，没有收到也要经常拉 CE，建议 10 到 20ms 这样拉一次 CE。拉 CE 是能解决一部分被干扰导致收不到数据的情况。

## 4.8 接收端收不到数据的时候马达操作

有些客户反映车子会溜一段距离。那是因为没有收到数据的时候，他的马达没有做超时停下来，导致马达会持续转动出现溜车的现象。但是有不能丢一包数据就停止马达，要不然经常丢包的时候或者丢包严重的时候回看到车子一抖一抖的。建议是几十 ms 收不到数据的时候马达可以停止转动了。这个只是建议，具体根据你的车型来设置超时时间。

## 5. OM6220 在操作是常出现的一些问题

### 5.1 关于拉 CE 的操作

在支持客户的过程中，发现有很多客户是不习惯去拉 CE 这个动作的，CE 一直处于拉高的状态，或者是 CE 拉的很频繁，或者 CE 拉的位置不对，所以这里指出正确的拉 CE 的问题，其实上面的发送和接收的介绍里面也有说了，只是需要再次强调一下。拉 CE 的好处是时序完全有用户自己掌控，什么时候想发数据或者收数据，就拉 CE。

发送状态：在还没有需要发送数据的时候 CE 是为低的，等你设置好频点，地址，功率，TX FIFO，清状态寄存器等这些操作之后，再拉高 CE 激活发送/或者是 CE 脉冲激活发送。发送完成 CE 就是为低的了。

接收状态：设置完频点，地址，清状态寄存器，清 RX FIFO 等等操作之后，CE 拉高等待数据到来，查询到接收数据标志之后拉低 CE，读取数据包长度，读取数据包内容，清状态，清 FIFO，切换地址切换频点等操作之后再次拉高 CE 等待数据到来。接收端要时常去拉 CE，避免接收效果不好的情况。

### 5.2 修改晶体负载电容值

我们芯片内部是有晶体负载电容的，可以通过修改寄存器的值来修改这个负载电容值，

从而修改频偏，当然频偏是跟晶体本身的参数有关，也跟 PCB 板有关，如果偏的太多就没有办法通过修改寄存器的值修正回来。需要更换晶体或者是改 PCB 晶体走线或者是铺铜间距。修改晶体负载电容值得寄存器在 BANK1 的 1E 寄存器(RF\_IVGEN 这个寄存器)的 bit4~0 一共 5 个位，0~31，32 个档位。

### 5.3 关于通信地址的问题

通信地址是 RF 硬件解调数据需要的，所以一个地址好不好就一想到通信的效果。我们的通信地址要求是不能出现连续的 6 个 0 或者连续的 6 个 1 也不能 5 个字节地址全部相同。6220 只有 5 个字节地址，没有 3 字节或者 4 字节地址。

比如 0x01=b00000001,0x80=0x10000000 就出现了连续的 7 个 1，比如 0x50,0x17=01010000 0001 0777，也是出现连续的 6 个 0 的情况，也不行。0x33,0x33,0x33,0x33,0x33 这样 5 字节地址相同的也不行。通信双方的地址一定要对应上，要不然切乱了地址也通信不上的。

### 5.4 关于通信频点的问题

通信频点可以从 2.400G 到 2.483G，间隔 1M 一个频点，建议不要使用 2416,2432,2448,2464,2480 这样的 16 的倍数的频点。

### 5.5 关于通信不上或者通信效果不好的问题

第一点是先关注频偏和功率。建议先用频谱仪看频偏和传导测试功率，确保频偏没有问题，输出功率没有问题。不要认为是同一个厂家的同一个批次的晶体，出来的频偏就是一样的，这个跟 PCB 板上的晶体走线线宽，长度，铜皮厚度，以及铺地的间距都是有关系的。

第二点是看程序，确认所设置的地址，频点，对不对得上，最好是同时抓发送端和接收端的时序图出来看，这样能直观的看出时序上面对不对得上，地址，频点是否设置对。

第三点在看下其他寄存器是否配置对应，一般是关注 CONFIG,EN\_AA,EN\_RXADDR,PMU\_CTL,RF\_CH,RX\_ADDR\_P0~2,TX\_ADDR,RX\_PW\_P0,DYNPD,FEATURE 这些寄存器。

第四点，一定要看时序图，确认芯片是校准完成的。



## 5.6 关于接收数据包长度不对的问题

客户在使用这个芯片做接收功能的过程中，发现虽然设置了固定长度数据包功能，但是接收到的数据包长度不是字节需要的数据包长度。其原因是由于这个芯片要操作 SPI 才会锁住 FIFO。也就是说如果客户不是经常通过读取状态寄存器来操作 SPI，那么假如发送端 1ms 发送一包 10 字节的数据，接收端 4ms 操作一次 SPI，那么接收的 FIFO 会接收 3 次完整的 10 字节数据，最后一包数据只能进来 2 字节数据，FIFO 的 32 字节就满了，当你查到接收标志之后再去看长度时就会发现长度不是你想要的长度。那这种情况下就要一直不停的操作 SPI 去读取状态寄存器，这样数据来了之后 FIFO 就会被锁住，后面的数据不会串进来。

## 5.7 关于跳频的问题

关于跳频这个问题，大部分的应用都是采用的慢跳频的方式，即发送端 n 个频点发送一轮为一个周期，接收端会在这个周期内接收数据，收不到再跳下一个频点接收，反正不管怎么样在一个周期内是肯定有机会接收到其中一个频点的数据的。缺点也明显，如果跳频的频点数量比较多，那么如果中间本来有机会收到数据的，结果被干扰到之后，那就要至少 2 个周期才能收到一包数据，而且发送间隔还不能设置太长，太长容易导致接收到的数据很少。发送间隔太短则遥控端功耗会大，不耐用。而且慢跳频由于跳频很慢，所以拉 CE 的时间间隔会很长，对干扰后恢复接收功能不利。

快跳频的好处是，发送端一般发送间隔不会设置很短，可以设置长一点，接收端快跳的时候时间不要设置太短，比如 1ms 这样太短了，2ms 这样比较合适，还要注意一定要错开时序，要不然一直同步追有可能一直都追不到频点，因为错不开时序了。追的过程中，不确定什么时候回同步上，可能下一个频点就能同步上，可能 2 个周期才能同步上。但是由于是切频点的时候会拉 CE，拉 CE 对干扰后恢复接收功能有利。

## 5.8 关于 3 线 SPI 的 SDA 线做输入输出功能说明

做为 3 线 SPI 接口时，一定要注意 BANK0 的 1C 寄存器的 bit3 一定要是 0，做四线的时候 bit3 是 1，不管在哪里设置这个寄存器都要注意。由于做 3 线 SPI 的时候 SDA(MOSI)这个脚需要根据你是写还是读来切换输入(读)输出(写)功能，所以一定要注意这个脚的切换。另

外，有些 MCU 的 SDK 不一定是读对应的 pin 脚回来就是一个“1”值可能是 0x40 这样的值。什么意思呢，比如你想通过读取 GPIO 输入电平的时候，你是根据是否读到为“1”就确定 SDA 脚输入是高电平，而实际可能你调用的函数返回来的是 0x40(GPIO6)，那你根据读回来是否为“1”来判断 SDA 是否输入高电平那就不对了。

OnMicro Confidential