AN1201
应用笔记

UM2052 应用手册

版本：V1.0

UNICMICRO
广芯微电子

广芯微电子（广州）股份有限公司
http://www.unicmicro.com/

条款协议

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

版本修订

版本	日期	描述
V1.0	2022.01.12	初始版

目录

1 概述

UM2052 是广芯微电子（广州）股份有限公司（以下简称“广芯微”）研制的一款 2.4G 收发芯片。这个使用手册是针对做玩具的客户应用或者是普通的 2.4G 的应用场景所写，所以这里只介绍 2.4G 的基本功能。

此芯片是可以和广芯微公司之前所出的所有的 2.4G 芯片互通的，不存在不兼容的问题，只需要配置好相应的寄存器。这个芯片有很多的新功能和新特点，这里也主要是讲玩具和其他 2.4G 普通应用的特点，主要的新特点有以下几点：

- 可以做 3 线或者 4 线 SPI 通信，只需要配置一下寄存器。
- 用寄存器控制或者是命令控制的形式。
- IRQ 引脚可以通过SPI 来控制输出，驱动马达或者 LED 灯。
- 简化初始化流程。

2 2.4G 应用初始化简介

此芯片初始化流程大致概括为以下几点：

1. 上电延时 10ms。
2. 进行一次软件复位是芯片确保在默认寄存器值下。
3. 关掉IO 复用功能，配置SPI 线数。
4. 给芯片上电，让晶振起振。
5. CE 脉冲(至少 40μs)。
6. 等待RF 校准完成。
7. 配置用户要使用的其他寄存器。

3 普通 2.4G 模式

3.1 2.4G 模式初始化配置流程

下面的代码是 SDK 中的初始化过程，初始化完成后再根据您的需要进行其他的配置。要注意，DYNPD 寄存器的选择 SPI 线数的位，不要在应用配置中改掉，还有 FEATRUE 寄存器的 Bit4，CONFIG 寄存器的 Bit7 一定要注意置”1”。

```
void UM2052_Init(void)
{
    unsigned char temp[5];

    //上电进行软件复位，复位之后左右的寄存器恢复默认值
    UM2052_write_byte(UM2052_BANK0_FEATURE, SOFT_RST);  //soft_reset
#if (UM2052_SPI_NWIRE == SPI_4_WIRE)

    //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
    //如果要用UM2052的IRQ脚控制马达的话，要设置IRQ脚，而且必须是在最前面设置它
    UM2052_write_byte(UM2052_BANK0_DYNPD, 0x08);
#else

    //根据需要配置是3线还是4线SPI
    //如果要用UM2052的IRQ脚控制马达的话，要设置IRQ脚，而且必须是在最前面设置它
    UM2052_write_byte(UM2052_BANK0_DYNPD, 0x00);
-#endif

    UM2052_CE_L0w();
    UM2052_write_byte(UM2052_BANK0_CONFIG, 0x8b);  //power up 给芯片上电
    delay_ms(3);  // wait 3 ms 必须至少等待3ms

    //UM2052_PWRDWN = 00  是芯片处于工作模式
    UM2052_write_byte(UM2052_BANK0_PMU_CTL, 0xac);
    delay_ms(2);  //必须至少等待2ms

    UM2052_Bank_Switch(UM2052_Bankl);
```

```
    UM2052_write_byte(UM2052_BANK1_TEST_PKDET, 0x20);  // pll_vdiv2_sel = 0

    /* increase fitler agc threshold start */
    temp[0] = 0x01;
    UM2052_wr_buffer(UM2052_BANK1_FAGC_CTRL_1, temp, 1);
    UM2052_Bank_Switch(UM2052_Bank0);
    UM2052_CE_High();
    delay_us(100);  //CE脉冲至少40us，使芯片启动校准流程
    UM2052_CE_Low();  //一定要注意，校准的时候CE是低的

    //读取 RF_SETUP直到芯片校准完成
    while((UM2052_read_byte(UM2052_BANK0_RF_SETUP) & 0x20) == 0x00);
    UM2052_write_byte(UM2052_BANK0_RF_SETUP, 0x40);  //cal_en = 0 清除校准使能位
    UM2052_Bank_Switch(UM2052_Bank1);
    temp[2] = 0x75;  // bp_dac = 1 bp_rc = 1
    temp[1] = 0x98;  // bp_vco_amp = 1 bp_vco_1do = 1
    temp[0] = 0x20;
    UM2052_wr_buffer(UM2052_BANK1_CAL_CTL, temp, 3);

    //设置TX RX的通信地址，属于用户应用配置的内容
    UM2052_Bank_Switch(UM2052_Bank0);
    temp[0] = 0x46;
    temp[1] = 0x0b;
    temp[2] = 0xaf;
    temp[3] =0x43;
    temp[4] = 0x98;
    UM2052_wr_buffer(UM2052_BANK0_RX_ADDR_P0, temp, 5);  //set address
    UM2052_wr_buffer(UM2052_BANK0_TX_ADDR, temp, 5);  //set address

    //这里可以在使用的时候去写，直到这里，目前CE还是处于拉低的状态
    UM2052_Clear_All_Irq();
    UM2052_Flush_Tx();
}
```

1. 芯片上电等待10ms，然后进行软复位(Soft_rst)。
2. 在bank0的DYNPD寄存器中把Bypass_io置0，关掉IO复用功能，否则MOSI会有电平冲突；如果需要4线SPI，把Spi4_en置1，如果需要3线SPI，Spi4_en用默认值0。

3. bank0的CONFIG寄存器中把PWR_UP置1，打开晶振，然后延时3ms让晶振启动。
4. bank0的PMU_CTL中把RF_PWRDWN置00，让芯片进入工作模式，延时2ms，让数字开电。
5. bank0的FEATURE寄存器中把vco_amp_tx_mux置0。
6. bank1的TEST_PKDET中把pll_vdiv2_sel置00。
7. 拉CE pulse，CE pulse宽度大于40μs，然后在bank0的RF_SETUP中查询CAL_DONE，CAL_DONE=1表示校准完成。
8. bank0的RF_SETUP中把CAL_EN置0。
9. bank1的CAL_CTL寄存器配置为0x28 0x75 0x98 0x20（高字节在前）。

3.2 2.4G 使用时常用寄存器配置

下面为2.4G 应用时寄存器配置代码：

```
/* 初始化 UM2052 模块，注意初始化里面，CE已经是拉低的 */
UM2052_Init();

//feature和dynmic寄存器要根据实际应用配置，而且收发端要对应上，否则会导致收到的数据不对
//这些设置都是在CE拉低的状态下进行的
UM2052_write_byte(UM2052_BANK0_FEATURE, 0x10);
UM2052_write_byte(UM2052_BANK0_EN_AA, 0x00);  //是否使能自动应答

//注意config寄存器的最高位是“1”，其他位根据需要配置，
//不建议把中断映射到IRQ脚，因为IRQ脚可能还要做输出脚用
UM2052_write_byte(UM2052_BANK0_CONFIG, 0xfa);

//配置数据包长度，如果是定长的话，只能收发所设置的长度的数据包，
//动长的话就1~32字节都可以
UM2052_write_byte(UM2052_BANK0_RX_PW_P0, 10);
UM2052_write_byte(UM2052_BANK0_RF_CH, 5);

//0x03, scramble_en = 0, 0x0b, scramble_en = 1
UM2052_write_byte(UM2052_BANK0_EN_RXADDR, 0x03);
UM2052_Flush_Tx();
UM2052_Flush_Rx();
UM2052_Clear_All_Irq();
```

3.3 2.4G 配置需要注意的一些事项

3.3.1 软件复位

芯片的复位标志是在 BANK0 的 FEATURE 寄存器的Bit5。

FEATURE (RW) Address: 1DH

表 3-1：FEATURE 寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Long_packet_en	Ble_en	Soft_rst	BP_GAU	Vco_amp_tx_mux	EN_DPL	EN_ACK_PAY	EN_DYN_ACK
0	0	0	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

3.3.2 CONFIG 寄存器

芯片的 CONFIG 寄存器的最高位要置”1”，最高位不置”1”的话不能进行收发数据。CONFIG 寄存器设置发送模式要设置为 0x8A。UM2052 没有配置 CRC 是 1 字节还是 2 字节的位。

CONFIG (RW) Address: 00H

表 3-2：CONFIG寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Tx_gurd_en	MASK_RX_DR	MASK_TX_DS	MASK_MAX_RT	EN_CRC	CE_REG	PWR_UP	PRIM_RX
1	0	0	0	1	0	0	1
RW	RW	RW	W	RW	RW	RW	RW

位编号	位符号	说明
7	Tx_gurd_en	最高位置“1”
3	EN_CRC	是否使能CRC
2	CE_REG	是否寄存器控制CE，还是命令控制CE

3.3.3 数据白化功能

白化标志位，此芯片是放在 BANK0 的 EN_RXADDR 的 Bit3 这个位，配置的时候要开白化功能。

EN_RXADDR (RW) Address: 02H

表 3-3：EN_RXADDR寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved				scramble_en	ERX_P2	ERX_P1	ERX_P0
0				1	0	1	1
RW				RW	RW	RW	RW

3.3.4 FEATURE 寄存器

FEATURE寄存器，在 2.4G 模式下，Bit3位要置”1”。低 3bit 根据实际使用配置。

FEATURE (RW) Address: 1DH

表 3-4：FEATURE寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Long_packet_en	Ble_en	Soft_rst	BP_GAU	Vco_amp_tx_mux	EN_DPL	EN_ACK_PAY	EN_DYN_ACK
0	0	0	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

3.3.5 DYNPD 寄存器

芯片的 DYNPD 寄存器低 3Bit 是对应 3 个 pipe，剩下的是其他功能 Bit。这里需要注意的是，Bit3 是 3 线/4 线 SPI 的控制位，如果使用的是 3 线 SPI 的话，这个 Bit 要置”0”，如果是使用 4 线的话，要置”1”。如果配置不对的话， SPI 线用逻辑分析仪抓的时候抓不到所读的数据。

DYNPD (RW) Address: 1CH

表 3-5：DYNPD寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reg_output	Reg_output_en	Bypass_io	Xn_en	Spi4_en	DPL_P2	DPL_P1	DPL_P0
0	0	1	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

3.3.6 接收端防死机处理

在接收端的接收函数，需要做下面的 2 个处理，一个是 20～50ms 时间收不到数据，要进行一次睡眠唤醒；如果 300～500ms 的时间一次数据都收不到，则要软件复位重新初始化一次。

```
/* 以下的复位RF和让RF睡眠唤醒的动作是必须的，不可缺少 */
if(reset_1ms_cnt > 300)  //如果300ms都没有收到1包数据，则复位RF重新初始化
{
    UM2052_Init();
    UM2052_write_byte(UM2052_BANK0_FEATURE, 0x10)；
    UM2052_write_byte(UM2052_BANK0_EN_AA, 0x00);
    UM2052_write_byte(UM2052_BANK0_CONFIG, 0xfb);
    UM2052_write_byte(UM2052_BANK0_RX_PW_P0, 10);
    UM2052_write_byte(UM2052_BANK0_RF_CH, 5);
    UM2052_write_byte(UM2052_BANK0_EN_RXADDR, 0x03);  //scramble_en = 0
```

```
    UM2052_CE_High();
    Reset_1ms_cnt = 0;  //清零复位RF计时器
}

// 发送端是4ms一个周期发一包数据，这里20ms的话就有5包数据的时间了
if((tim2_1ms_cnt & 20) == 0)  //如果20ms都没有收到1包数据，则让RF睡眠唤醒一次
{
  UM2052_CE_Low();
  UM2052_Flush_Rx();
  UM2052_Clear_All_Irq();
  UM2052_write_byte(UM2052_BANK0_PMU_CTL, 0xAE);  //RF进入深睡眠
  Delay_ms(1);  //这里的1ms延时不可缺少，也不能少
  UM2052_write_byte(UM2052_BANK0_PMU_CTL, 0xAC);  //唤醒RF
  Delay_ms(1);  //这里的1ms延时不可缺少，也不能少
  UM2052_CE_High();
}
```

3.3.7 接收端接收处理数据的正确顺序

```
UM2052_CE_High();  // CE拉高才能接收数据
for( ; ; )
{
  delay_us(100);  // 延时100μs不至于频繁读取状态寄存器
  status = UM2052_read_byte(UM2052_BANK0_STATUS);  //读状态寄存器看是否有收到数据
  if ((UM2052_STATUS_RX_DR & status))  //如果有收到数据
  {
    Len = UM2052_Receivepack(RX_Payload);  //读取FIFO里面数据的长度
    If(len ==10)  //如果FIFO里面是10个字节的数据
    {
    }

    // 接收到数据并读取FIFO之后，这个操作必须的
    //要注意收到数据之后的正确操作：CE先拉低，清FIFO，清标志，设置频点，设置地址，…
    //CE拉高进入接收状态
    UM2052_CE_Low();  //CE拉低
```

```
        UM2052_Flush_Rx();  清空RX FIFO
        UM2052_Clear_All_Irq();  //清状态寄存器标志
        UM2052_CE_High();  //CE拉高等待下次数据到来
    }
}
```

3.3.8 发送端正确的发送数据操作

```
UM2052_SendPack(UM2052_W_TX_PAYLOAD, Tx_Payload, 10);

// 发送端：只有在有数据需要发送的时候才给CE信号，其他时间CE都是拉低的
// 如果需要切换到接收的话，那么在接收状态下CE要一直为高
UM2052_CE_High();  // CE pulse 20μs
delay_us(35);
UM2052_CE_Low();
delay_us(3500);
UM2052_Flush_Tx();
UM2052_Clear_All_Irq();
```

说明：

- CE脉冲发送方式：正常状态下CE是拉低的
  1. 设置频点
  2. 设置地址
  3. 设置功率…等
  4. 写入TX FIFO数据
  5. CE拉高
  6. 延时至少20μs
  7. CE拉低
  8. 查询状态寄存器等待数据发送完成
  9. 下一次数据发送
- CE高电平发送方式：正常状态下CE是拉低的
  1. 设置频点
  2. 设置地址
  3. 设置功率…等
  4. 写入TX FIFO数据

5. CE拉高
6. 延时1ms（1ms时间数据是可以发完的）
7. CE拉低
8. 下一次数据发送

3.3.9 IRQ 脚的输出控制功能

要使用 IRQ 脚做输出控制功能，需要在芯片上电初始化的时候就配置IRQ 脚，以免误控制。IRQ 脚控制使能位在 DYNPD 寄存器的 Bit6。然后控制 IRQ 脚是高电平还是低电平是在 DYNPD 寄存器的 Bit7 这个位。

```
void UM2052_Init(void)
{
   unsigned char temp[5];
   UM2052_write_byte(UM2052_BANK0_FEATURE, SOFT_RST);  //soft_reset
#if (UM2052_SPI_NWIRE == SPI_4_WIRE)

   // 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
   UM2052_write_byte(UM2052_BANK0_DYNPD, 0x08 | 0x40);

   // 如果要用UM2052的IRQ脚控制马达的话，要设置IRQ脚，而且必须是在最前面设置它
#else
   UM2052_write_byte(UM2052_BANK0_DYNPD, 0x00 | 0x40);
   // 如果要用UM2052的IRQ脚控制马达的话，要设置IRQ脚，而且必须是在最前面设置它
-#endif
}
```

```
#define LED1_HIGH()
do
{
      UM2052_write_byte(UM2052_BANK0_DYNPD,
UM2052_read_byte(UM2052_BANK0_DYNPD) | 0x80);
   delay_us(10);
}while(0)
#define LED1_Low()
do
{
```

```
    UM2052_write_byte(UM2052_BANK0_DYNPD,
UM2052_read_byte(UM2052_BANK0_DYNPD) & 0x7f);
  delay_us(10);
}while(0)
```

3.4 常用寄存器的配置介绍

3.4.1 CONFIG (BANK0--00)寄存器

做普通 2.4G 使用是 Bit7 要置”1”。Bit6~Bit4 是 RF 有中断产生的时候，信号在 IRQ 脚出现，低电平有效。Bit3 是开 CRC 功能。Bit2 是 CE 控制方式，”0”是命令控制，“1”是寄存器控制。Bit1 是芯片上电掉电位，”1”是给芯片上电。Bit0是发送接收模式控制位，“0”是发送，“1”是接收。

CONFIG (RW) Address: 00H

表 3-6：CONFIG寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Tx_gurd_en	MASK_RX_DR	MASK_TX_DS	MASK_MAX_RT	EN_CRC	CE_REG	PWR_UP	PRIM_RX
1	0	0	0	1	0	0	1
RW	RW	RW	W	RW	RW	RW	RW

3.4.2 EN_AA (BANK0--01)寄存器

EN_AA寄存器是使能pipe 的自动应答功能，低 3Bit 对应 3 个pipe。“1”是使能对应管道的自动应答，“0”是禁止对应 pipe 的自动应答。

EN_AA (RW) Address: 01H

表 3-7：EN_AA寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
reserved					ENAA_P2	ENAA_P1	ENAA_P0
0					1	1	1
RW					RW	RW	RW

3.4.3 EN_RXADDR (BANK0--02) 寄存器

EN_RXADDR寄存器的 Bit3 是白化功能开关，“1” 是打开白化功能，“0” 关闭白化功能。Bit[2:0]是对应 pipe 的使能。

EN_RXADDR (RW) Address:02H

表 3-8：EN_RXADDR寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved				scramble_en	ERX_P2	ERX_P1	ERX_P0
0				1	0	1	1
RW				RW	RW	RW	RW

3.4.4 PMU_CTL (BANK0--03)寄存器

PMU_CTL寄存器的 Bit[7:2] 初始化的时候设置成 0xA8，应用时设置成 0xAC，然后Bit[1:0]是工作模式，“00”是工作模式，“01”是深睡眠模式，“10”是浅睡眠模式。

PMU_CTL (RW) Address: 03H

表 3-9：PMU_CTL寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Rtc32k_rdy_enb_reg	Rtc32k_rdy_enb_mn	Digldo_enb_dly_reg	Digldo_enb_mn	Digldo_enb_reg	rtc32k_en	RF_PWRDWN[1:0]	
1	0	1	0	1	0	01	
RW	RW	RW	RW	RW	RW	RW	

3.4.5 SETUP_RETR (BANK0--04)寄存器

SETUP_RETR寄存器的 Bit[7:4]是重传的间隔时间，0=256μs，T=(n+1)*256μs，Bit[3:0]是重传的次数，是在开了自动重传的时候有效。

SETUP_RETR (RW) Address: 04H

表 3-10：SETUP_RETR寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ARD				ARC			
4’b0				4’b11			
RW				RW			

3.4.6 RF_CH (BANK0--05)寄存器

RF_CH寄存器是设置通信信道的，写入信道值从 0～83，对应的是 2400 到 2483。

RF_CH (RW) Address: 05H

表 3-11：RF_CH寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reg_Rf_ch							
0x32							
RW							

3.4.7 RF_SETUP (BANK0--06)寄存器

RF_SETUP寄存器的 Bit7 是载波位，”1”是发载波，“0”是发数据，要CE 为高才会有载波出来。Bit6+Bit[2:0]是功率档位设置寄存器。Bit5 是校准完成标志位，校准完成之后这个位要清零。Bit4 是使能校准位，“1”是使能校准。Bit3 是速率控制位， “1”是 2M 速率，“0”是 1M 速率。

RF_SETUP (RW) Address: 06H

表 3-12：RF_SETUP寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CONT_WAVE	PA_PWR[3]	CAL_DONE	CAL_EN	RF_DR_HIGH	Pa_power		
0	1	0	1	0	3’b000		
RW	RW	RW	RW	RW	RW		

3.4.8 STATUS (BANK0--07)寄存器

正常读取STATUS寄存器的时候，应该是 0x0E 这个值。Bit7 是表示当前 BANK，“1” 是 BANK1，”0”是 BANK0，一定要确保我们是在 BANK0 下做通信操作。Bit6 是接收到数据时为“1”。Bit5 是发送完成时为“1”。 Bit4 是最大传输次数达到时为“1”。 Bit3 是表示当前 CE 的高低电平状态。Bit[2:1]是表示哪个 pipe 有数据，都没有数据时是“11”。Bit0 是 TX FIFO 是否满。

STATUS (RW) Address: 07H

表 3-13：STATUS寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BANK	RX_DR	TX_DS	SYNC_DS	CE	RX_P_NO		TX_FULL
0	0	0	0	0	2’b11		0
R	RW	RW	RW	R	R		R

3.4.9 RX_ADDR_P0 (BANK0--0A)寄存器

RX_ADDR_P0寄存器是设置接收时的地址的， 一共 5 个字节。 RX_ADDR_P1 和 RX_ADDR_P2 的高 4 字节和 RX_ADDR_P0 完全一样，只有低字节不一样，低字节是 C2 和C3。

3.4.10 TX_ADDR (BANK0--10)寄存器

TX_ADDR寄存器是设置发送时的地址，一共 5 个字节。

3.4.11 RX_PW_P0 (BANK0--11)寄存器

RX_PW_P0寄存器是设置 pipe0 收发数据包长度的，如果是设置成定长的数据包收发的话，

那么芯片只发送所设置的固定长度的数据包内容和接收固定长度的数据包内容，其他不符合长度的数据包是丢弃的。普通 2.4G 模式下最多 32 字节数据包长度。RX_PW_P1 和RX_PW_P2 是一样的道理。

3.4.12 FIFO_STATUS (BANK0--17)寄存器

FIFO_STATUS寄存器的 Bit6 表示是否重用数据包内容，如果你的数据包内容是重复使用的，不想每次都通过SPI 写一次，则可以使能这个位，然后通过 REUSE_TX_PL命令可以重用数据包。Bit5 是 TX FIFO 满标志，Bit4 是TX FIFO 空标志，Bit1 是 RX FIFO 满标志，Bit0 是 RX FIFO 空标志。

FIFO_STATUS (RW) Address: 17H

表 3-14：FIFO_STATUS寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved	TX_REUSE_PL	TX_FULL	TX_EMPTY	Reserved		RX_FULL	RX_EMPTY
0	0	0	1	0		0	1
RW	R	R	R	RW		R	R

3.4.13 DYNPD (BANK0--1C)寄存器

DYNPD寄存器的Bit3 是 3/4 线 SPI 选择位，Bit[2:0]是对应的 pipe 是否使能动态数据包长度。

DYNPD (RW) Address: 1CH

表 3-15：DYNPD寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reg_output	Reg_output_en	Bypass_io	Xn_en	Spi4_en	DPL_P2	DPL_P1	DPL_P0
0	0	1	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

位编号	位符号	说明
2	DPL_P2	Enable dynamic payload length data pipe 2. (Requires EN_DPL and ENAA_P2)
1	DPL_P1	Enable dynamic payload length data pipe 1. (Requires EN_DPL and ENAA_P1)
0	DPL_P0	Enable dynamic payload length data pipe 0. (Requires EN_DPL and ENAA_P0)

3.4.14 FEATURE (BANK0--1D)寄存器

FEATURE寄存器的 Bit5 是软件复位控制位。Bit4 是高斯滤波器开关位，做 2.4G 模式使用时 Bit4 要置“1”。Bit2 是使能动态数据包长度，Bit1 是使能 ACK 包，需要提前先把 ACK 包写到接收端的TX FIFO 里面去，配合 W_ACK_PAYLOAD 这个命令写数据到 TX FIFO，要动长 ACK 包的话配合 Bit2 使用，Bit0 是不带应答的。

FEATURE (RW) Address: 1DH

表 3-16：FEATURE寄存器

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Long_packet_en	Ble_en	Soft_rst	BP_GAU	Vco_amp_tx_mux	EN_DPL	EN_ACK_PAY	EN_DYN_ACK
0	0	0	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

位编号	位符号	说明
2	EN_DPL	Enable Dynamic Payload Length
1	EN_ACK_PAY	Enable Payload with ACK
0	EN_DYN_ACK	Enable the W_TX_PAYLOAD_NOACK command

4 使用注意事项

4.1 天线端防止烙铁漏电烧坏处理

为了防止在生产的时候，由于烙铁没有做防漏电处理，导致芯片的天线脚被烧坏，需要在芯片脚和天线焊接点中间串一个 5pf 左右的电容。

4.2 晶体负载电容

晶体两端的负载电容可以不用焊接。

4.3 SPI 接口电平

SPI 接口不支持 5V 电平，只能支持 3.3V 的电平。

4.4 通信地址要求

通信地址的设置，不能是所有的字节都一样，也不能出现连续的6 个”0”或者连续的 6 个”1”。也不能全部是 01010101….这样的。