RC816

------2.4GHz 高集成度无线收发芯片

功能描述：

- 频率范围2402~2530MHz，提供128个channel
- 无线速率： 1Mbps
- 支持SPI或I2C接口
- 自动应答及自动重传
- 快速频道切换，支持跳频算法
- 支持RSSI功能
- 支持软件复位
- 简单低成本外围元器件
- 低工作电压：1.9V~3.6V
- SSOP16封装

应用领域：

- 工业传感器及无线工控设备
- 无线游戏设备
- 遥感勘测
- 安防系统
- 智能运动设备
- 智能电视遥控器
- 无线标签
- 无线门禁
- 安防系统
- 遥控装置
- 无线玩具

目录

1. 概述

RC816是一款工作在2.4GHz世界通用ISM频段的单片集成无线收发芯片。该芯片集成了射频接收器、射频发射器、频率综合器、GFSK调制器、GFSK解调器等功能模块。通过SPI或I2C接口可以对输出功率、频道选择以及协议进行灵活配置，并且内置CRC、FEC、自动应答和自动重传机制，可以大大简化系统设计并优化性能。外围电路简单，只需搭配极少的被动器件就能达到优良的收发性能。

芯片的优点:

低功耗

- 当工作在发射模式下（发射功率为0dBm）时电流消耗为14mA；
- 当工作在接收模式时电流消耗为16mA；
- 当工作在休眠模式时电流消耗低于1uA。

低成本

- 低成本系统解决方案；
- 外围元器件低于5个；
- 内部集成上电复位和软件复位功能，外围控制简单；
- 采用双层PCB，可以采用印制板天线方式；

高性能

- 工作频率 2402MHz~2530MHz；
- 最高数据码率1Mbps；
- 最大输出功率6dBm，灵敏度可达-89dBm
- 正常配置下（发射功率6dBm），空旷地带通信距离大于80米

2.极限值

表2.1 参数极限值

参数	符号	最小值	最大值	单位
工作温度	Top	-40	100	℃
存储温度	Tstor	-55	125	℃
工作电压	VDD	-0.3	3.7	V
输入射频信号强度	Pin_max		+10	dBm
ESD(人体模型)	ESD_HBM		2	KV

* 注意：强行超过一项或多项极限值使用会导致器件永久性损坏。

* 小心：芯片为静电敏感器件，操作时请遵守防护规则。

3.主要电特性

表3.1 芯片主要电特性

特　性	条件(除另有规定外，VCC = 3.3V， TA=25℃)	参数值			单位
		最小	典型	最大	
ICC	休眠模式		0.8		uA
	待机模式		600		uA
	发射模式 (0dBm)		14		mA
	接收模式		16		mA
系统指标					
f_{OP}	工作频率	2402		2530	MHz
f_{XTAL}	晶振频率		12		MHz
PLL_stable	PLL 稳定时间		150		us
	码率		1		Mbps
FCH_{1M}	频道间隔		1		MHz
发射模式指标					
PRF	最大输出功率		6		dBm
PRF	典型输出功率		0		dBm
$PRFC$	输出功率范围	-18		6	dBm
$PBW1$	载波调制的 20dB 带宽 (1Mbps)		1	1.1	MHz

接收模式指标					
RXSENS2	接收灵敏度（0.1%BER）		-89		dBm
抗干扰特性					
C/I_{CO}	同道干扰		9		dBc
C/I_{1ST}	第 1 相邻道干扰		5		dBc
C/I_{2ND}	第 2 相邻道干扰		-12		dBc
C/I_{3RD}	第 3 相邻道干扰		-24		dBc
操作条件					
VDD	供电电压	1.9	3	3.6	V
VSS	芯片地		0		V
V_{OH}	高电平输出电压	VDD-0.3		VDD	V
V_{OL}	低电平输出电压	VSS		VSS+0.3	V
V_{IH}	高电平输入电压	2.0	3	3.6	V
V_{IL}	低电平输入电压	VSS		VSS+0.3	V
Cin	输入电容			10	pF
	工作温度	-40	27	+100	℃
	储存温度	-40	27	+125	℃

Notes:　12MHz 晶振的负载电容为22pF

4.系统结构方框图

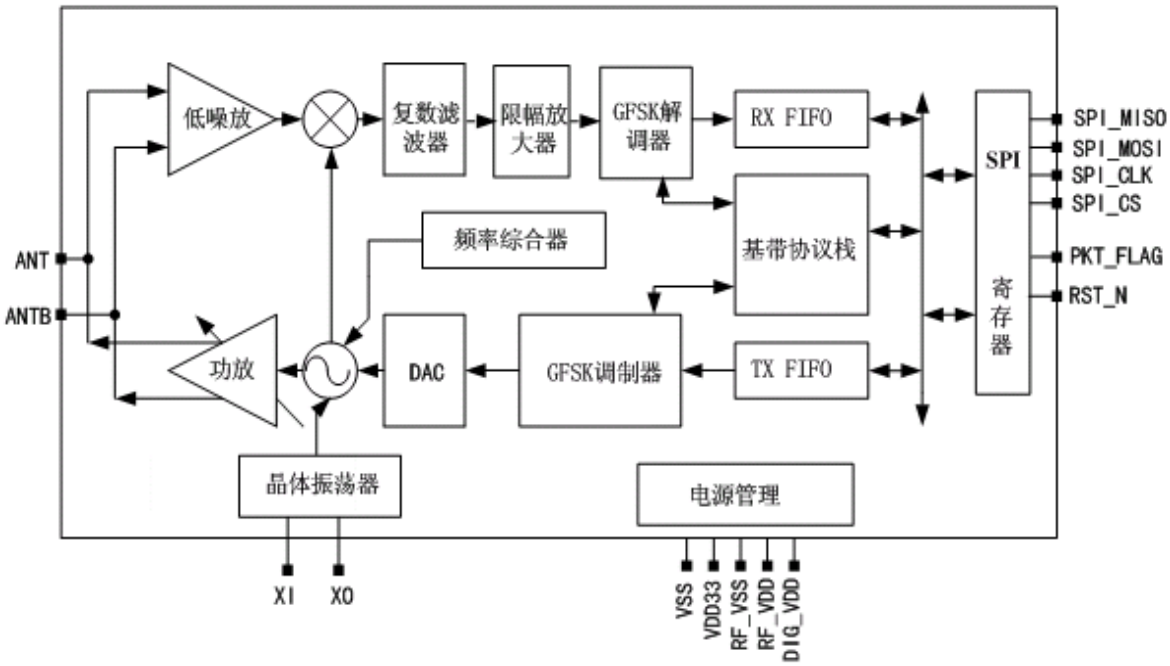


图4.1 系统结构方框图

5.引脚定义

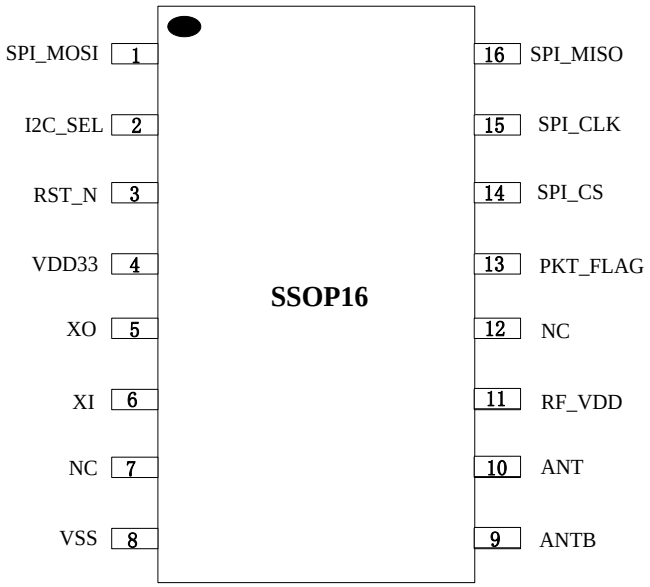


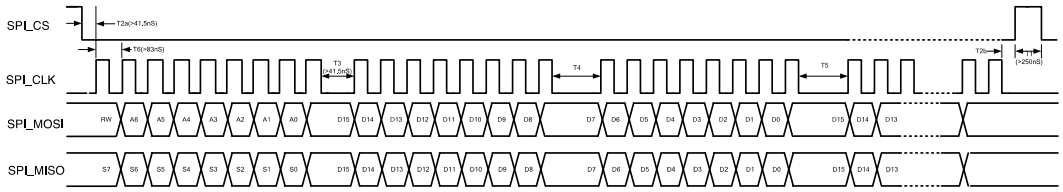
图5.1　SSOP16 引脚图

表5.1　SSOP16 引脚功能说明

序号	Pin Name	Type	Description
1	SPI_MOSI	DI	SPI 数据输入脚
2	I2C_SEL	DI	接口模式选择 0：SPI 1：I2C
3	RST_N	DI	芯片复位脚，低电平有效；复位后寄存器数值丢失，全部变为默认值。
4	VDD33	PWR	3.3V 电源出入
5	XO	AO	晶体振荡器输出
6	XI	AI	晶体振荡器输入
7	NC	NC	NC
8	VSS	GND	地
9	ANTB	RF I	射频输入 1
10	ANT	RF I	射频输入 2
11	RF_VDD	PWR	内部 1.8V 电源输出
12	NC	NC	NC
13	PKT_FLAG	DO	发射接收状态标志位
14	SPI_CS	DI	使能信号，低有效，拉低可使芯片退出 sleep mode
15	SPI_CLK	DI	SPI/I2C 时钟输入脚
16	SPI_MISO	D I/O	SPI 数据输出脚/I2C 数据脚

6.SPI 接口

6.1 SPI 时序格式



Notes:

1. SPI 为下降沿采样数据，上升沿改变数据；MCU 在控制 SPI_MOSI 时一定要在 SPI_CLK 上升沿之后再改变数据。
2. SPI 读写位： 写=0，读=1。
3. 访问接收 FIFO 寄存器 0X28 时，可以按字节读。访问多个 FIFO 数据时可以用一个 SPI_CS 周期。
4. 访问除 FIFO 外的其他寄存器时，一次要读写 16bits。
5. 访问除 FIFO 个的其他多个寄存器时，可以用一个 SPI_CS 周期。此时，地址只要写一次，然后是 16bit 数据，当写完一个寄存器值后，芯片会自动增加寄存器地址。

6.2 SPI 时序要求

Name	Min	Typ.	Max	Description
T1	250ns			两次 SPI 访问的间隔时间
T2a, T2b	41.5ns			SPI_CS 和 SPI_CLK 的间隔
T3	Note 1			地址和数据间隔时间
T4	Note 1			高位字节和低位字节的时间间隔
T5	Note 2			两个寄存器数据的时间间隔
T6	83ns			SPI_CLK 时钟周期

Notes:

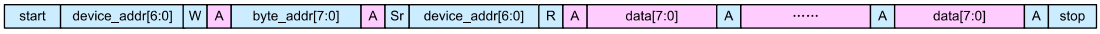
1. 在访问寄存器 0x28 中的 FIFO 数据时，芯片需要 450ns 去找到正确的读 FIFO 读取的指针地址。
2. 当读寄存器 0x28 中的 FIFO 数据时，至少需要等 450ns。
   读其他寄存器时，T5min=41.5ns

7. I2C 接口

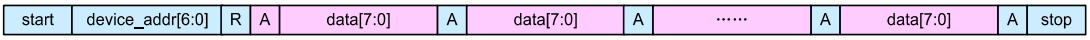
7.1 I2C 命令格式

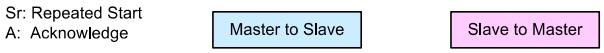
Master 写一个或多个字节到 FIFO 或 Register

Master 写一个字节到指定的寄存器，然后从 FIFO 中读出 1 个或多个字节数据



Master 可以连续读出 FIFO 数据





7.2 I2C 特性

I2C 设备从模式可选特性	是否支持
Standard mode – 100kbps	Yes
Fast mode – 400 kbps	Yes
Fast mode plus – 1000 kbps	Yes
High speed mode – 3200 kbps	No
Clock stretching	No
10-bit slave address	No
General call address	No
Software reset	No
Device id	No

7.3 I2C 器件地址

在 I2C 模式下，芯片器件地址如下：

A6	A5	A4	A3	A2	A1	A0	R/W
0	1	由 SPI_MOSI 决定	1	0	0	0	Read=1 Write=0

8. 状态机控制图

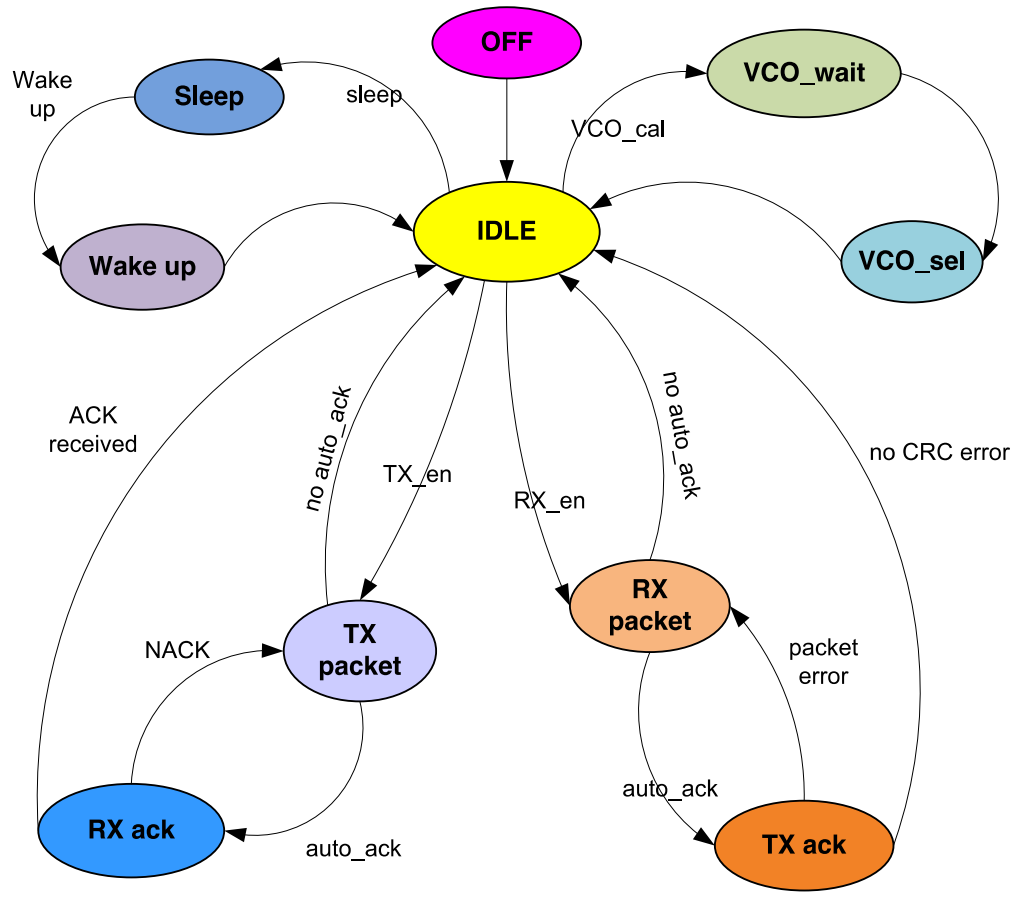


图 8.1 状态机控制图

9. 部分寄存器描述

RF Synthesizer TX/RX control Register 0x00 default : 0x0030

Bit	Name	R/W	Description	Default
15	TX_EN	R/W	使能芯片进入发送状态，1 有效	0
14	RX_EN	R/W	使能芯片进入接收状态，1 有效 注：TX_EN 和 RX_EN 不能同时为 1，同时为 0 时芯片处于 IDLE 状态	0
13-12	reserved	R/W		00
11-7	SWALLOW[4:0]	R/W	当 RF_PLL_DIIRECT=1 时，	00000

			载波频率由寄存器直接设置： RF={reg0[6:0],reg0[13:9]}	
6-0	RF_PLL_CH_NO[6:0]	R/W	当 RF_PLL_DIIRECT=0 时，载波频率由信道号设置： RF=2402+RF_PLL_CH_NO[6:0]	110000 f=2450MHz

Miscellaneous configuration Register 0x01 default: 0x2077

Bit	Name	R/W	Description	Default
15	Sleep_mode	R/W	使能芯片进入 sleep 模式，1 有效（SPI_CS 需同时保持高电平）	0
14		R/W		0
13	CRC_EN	R/W	0: CRC 关闭 1: CRC 开启	1
12				
11				
10	RF_PLL_DIRECT	R/W	当 RF_PLL_DIIRECT=1 RF={reg0[6:0],reg0[13:9]} 否则 RF=2402+RF_PLL_CH_NO[6:0]	0
9	Pkt_hint_pority	R/W	1: PKT/FIFO flag 低有效 0: PKT/FIFO flag 高有效	0
8	Miso_tri_opt	R/W	0: SPI_CS=1 时 SPI_MISO 为高阻态输出 1: SPI_CS=1 时 SPI_MISO 为低电平输出	0
7	Reset_system	W/R	1：使能芯片进行软件复位 注：复位前必须先把 reg0x1e[0] 写为 1	0
6-4				
3	Auto-ack	R/W	1:接收到数据后发送 ACK 0: 接收到数据后不发送 ACK，直接进入 IDLE	0
2	Pack_lenth_en	R/W	1: payload 的第一个字节作为包长度	1

1	Fw_term_tx	R/W	0: 由 MCU 控制何时终止 TX 状态 1: 当 FIFO 写指针等于读指针时，芯片自动结束 TX 状态	1
0	SCRAMBLE_EN	R/W	0: scramble off 1: scramble on	1

PA Power Control Register 0x02 default: 0x3080

Bit	Name	R/W	Description	Default
15-12	reserved	R/W		0011
11-8	reserved	R/W	rev	0000
7-4	reserved	R/W		1000
3-0	PA_PW_SET[4:0]	R/W	PA 输出功率控制： 1111: min 1000: med 0000: max	0000

Operation configuration Register 0x03 default: 0x5800

Bit	Name	R/W	Description	Default
15-13	Preamble_len[2:0]	R/W	000: 1 byte, 10101010 001: 2 byte, 10101010 10101010 111: 8 byte, 10101010.....	010
12-11	Syncword_len[1:0]	R/W	00: 16 bits,{reg31[15:0]} 01: 32 bits, {reg31[15:0], reg34[15:0]} 10: 48 bits, {reg31[15:0], reg33[15:0], reg34[15:0]} 11: 64 bits, {reg31[15:0], reg32[15:0], reg33[15:0], reg34[15:0]}	11
10-8	Trailer_len[2:0]	R/W	000: 4 bits, 1010 001: 6 bits, 101010 ..... 111: 18 bits,101010.....101010	000
7-6	Data packet type[1:0]	R/W	00: NRZ law data 01:Manchester data type 10: 8/10 line code 11: interleave data type	00

5-4	FEC type[1:0]	R/W	00: No FEC 01: FEC 13 10: FEC 23 11: reserve, same as 00	00
3				
2-0	brclk_sel[2:0]	R/W	000: brclk keep 0 001: xtal_core out 010: crystal divided by 6, 2M out 011: crystal divided by 12, 1M out 100: APLL_clk out 101: clk_tx_out	000

Operating configuration Register 0x07 default: 0x7311

Bit	Name	R/W	Description	Default
15	EN_VCO_CAL_IDLE	R/W	1: 在 IDLE 状态人工使能 VCO 校准过程	0
14	TXRX_VCO_CAL_EN	R/W	1: TX/RX 状态下，VCO 自动进行校准	1
13-10	TXRX_vco_tim[3:0]	R/W	TXRX 状态下 VCO 自动校准等待时间： 0000: 12us 0001: 14us 1111: 42us	1100
9-0				

Timing configuration Register 0x0b default: 0x837F

Bit	Name	R/W	Description	Default
15-12	TX_CW_DLY[3:0]	R/W	PA 开启后，发送 CW 的时间： 0000: 4us 0001: 6us 1111: 34us	1000

11-8	Re_transmit_time	R/W	ACK 功能开启时，最大重发次数： 2H: 2 times 3H: 3 times	0011
7-6		R/W		01
5-0	Rx_ack_time[5:0]	R/W	发射机等待 ACK 返回的时间： T=16us x Rx_ack_time[5:0] （超过这个时间如果还没收到返回的 ACK 信号，会自动重发）	111111

Timing configuration Register 0x0c default: 0x3E11

Bit	Name	R/W	Description	Default
15-10	VCO_ON_DLY[5:0]	R/W	开启 TX/RX 状态后，等待 VCO 稳定时间： 000000: 4us 000001: 8us 111111: 256us	001111
9-8	TX_PA_OFF_DLY[1:0]	R/W	PA 关闭延时： 00: 4us 01: 6us 10: 8us 11: 10us	10
7-4	TX_PA_ON_DLY[3:0]	R/W	PA 开启延时： x000: 4us x001: 8us x1111: 32us	0001
3-0	TX_SW_ON_DELAY[3:0]	R/W	TX_SW 开启延时： 0000: 8us 0001: 12us 11111: 68us	0001

RSSI value Register 0x11 Read only

Bit	Name	R	Description	Default
15-8	RAW_RSSI[7:0]	R	8 bit RSS 值（实时更新，保留最大值）	
7-0	rssi_lat[7:0]	R	RSSI latch	

Block status Register 0x12 Read only

Bit	Name	R	Description	Default	Optium
15-14					
13	FIFO_FLAG	R	1: FIFO 为空或满的标志位		
12	reserved	R			
11					
10	VCO_CAL_ERROR	R	1：VCO 校准失败		
9-0					

Main status Register 0x16 Read only

Bit	Name	R	Description	Default	Optium
15	CRC_error	R	1：CRC 错误 0: CRC 正确		
14	FEC23_error	R	1：FEC23 错误 0: FEC23 正确		
13-8	Framer_st	R	Framer 状态： 31H: ST_SLEEP 33H: ST_WAKE_UP.....		
7	Syncword_rev	R	1: 接收到 syncword，只在接收状态下有效		
6	PKT_FLAG	R	Packet flag 标志位		
5-3	Tx_st[2:0]	R	Transmit packet control state 000: TX_IDLE 001:TX_ID 010: TX_CRC 011: TX_DATA 110: TX_DONE		
2-0	Rx_st[2:0]	R	Receive packet control state 000: RX_IDLE 001:RX_ID		

			010: RX_DATA 011: RX_TRAILER 100: RX_DONE 110: RX_CRC		

AMS TEST Control Register 0x1c default: 0x4008

Bit	Name	R/W	Description	Default	Optim
15	RSSI_PDN	R/W	1: 关闭 RSSI 功能 0：开启 RSSI 功能	0	
14-0		R/W		1	

Operating Control Register 0x1e default: 0x7FF4

Bit	Name	R/W	Description	Default	Optim
15-14		R/W		7FF4H	
0	Reg_Reset_EN	R/W	1: 开启软件复位功能，1 有效。使能后再写 reg0x01[7]为 1 就可完成对芯片的复位。	0	

SYN_WORD_1 Register 0x1f default: 0x0101

Bit	Name	R/W	Description	Default
15-0	SYNC_WORD[15:0]	R/W	LSB bits of SYN_WORD is first	0101H

SYN_WORD_2 Register 0x20 default: 0x0202

Bit	Name	R/W	Description	Default
15-0	SYNC_WORD[31:16]	R/W	LSB bits of SYN_WORD is first	0202H

SYN_WORD_3 Register 0x21 default: 0x0303

Bit	Name	R/W	Description	Default
15-0	SYNC_WORD[47:32]	R/W	LSB bits of SYN_WORD is first	0303H

SYN_WORD_4 Register 0x22 default: 0x0404

Bit	Name	R/W	Description	Default
15-0	SYNC_WORD[63:48]	R/W	LSB bits of SYN_WORD is first	0404H

Register 0x23 default: 0x8001

Bit	Name	R/W	Description	Default

15	FIFO_share_en	R/W	1: RX/TX FIFO 共享，总长 64bytes 0：RX/TX FIFO 不共享，各自 32bytes	1
14:0				

FIFO threshold_reg Register 0x24 default: 0x4401

Bit	Name	R/W	Description	Default
15-11	FIFO_empty_threshold	R/W	认为 FIFO 为空的阈值	01000
10-6	FIFO_full_threshold	R/W	认为 FIFO 为满的阈值	10000
5-0	Synword_threshold	R/W	认为 syncword 正确的阈值, XX 表示错 XX-1 bit 也认为是正确（设 1 表示全部正确）	000001

RX_FIFO_RD_PTR Register 0x25 default: 0x0000

Bit	Name	R/W	Description	Default
15	RXFIFO_CLR_W_PTR	R/W	1: 清空 RX FIFO 写指针	0
14	RX_FIFO_WR_PTR*[6]*	R/W	RX FIFO 写指针最高位，当 fifo_share_en=1 有效	0
13-8	RX_FIFO_WR_PTR*[5:0]*	R/W	RX FIFO 写指针低 6 位	0
7	RXFIFO_CLR_R_PTR	R/W	1: 清空 RX FIFO 读指针	0
6	RX_FIFO_RD_PTR*[6]*	R/W	RX FIFO 读指针最高位，当 fifo_share_en=1 有效	0
5-0	RX_FIFO_RD_PTR*[5:0]*	R/W	RX FIFO 读指针低 6 位	0H

TX_FIFO_RD_PTR Register 0x26 default: 0x0000

Bit	Name	R/W	Description	Default
15	TXFIFO_CLR_W_PTR	R/W	1: 清空 TX FIFO 写指针	0
14	TX_FIFO_WR_PTR*[6]*	R/W	TX FIFO 写指针最高位，当 fifo_share_en=1 有效	0
13-8	TX_FIFO_WR_PTR*[5:0]*	R/W	TX FIFO 写指针低 6 位	0
7	TXFIFO_CLR_R_PTR	R/W	1: 清空 RX FIFO 读指针	0
6	TX_FIFO_RD_PTR*[6]*	R/W	TX FIFO 读指针最高位，当	0

			fifo_share_en=1 有效	
5-0	TX_FIFO_RD_PTR*[5:0]*	R/W	TX FIFO 读指针低 6 位	0H

TX_FIFO_REG Register 0x27 default: 0x0000

Bit	Name	R/W	Description	Default
15-0	TX_FIFO_REG	R/W	MCU 写 FIFO 数据的接口	0000H

RX_FIFO_REG Register 0x28 default: 0x0000

Bit	Name	R/W	Description	Default
15-0	RX_FIFO_REG	R/W	MCU 读 FIFO 数据的接口	0000H

10. 寄存器默认值和优化值

芯片共有 42 个寄存器，复位后，所有寄存器均为默认值，如表 10.1 所示。正常工作时，只需要对少数几个寄存器的值进行优化即可，如表 10.2 所示。

表 10.1 寄存器默认值

Address	reset value	Address	reset value
0x00	0x0030	0x15	Read only
0x01	0x2077	0x16	Read only
0x02	0x3080	0x17	0x0000
0x03	0X5800	0x18	0x6FE1
0x04	0x4A00	0x19	0x1300
0x05	0x7126	0x1a	0x07F7
0x06	0x1988	0x1b	0x1800
0x07	0x7311	0x1c	0x4008
0x08	0x1659	0x1d	0x0000
0x09	0x007B	0x1e	0x7FF4
0x0a	0x2433	0x1f	0x0101
0x0b	0x837F	0x20	0x0202
0x0c	0x3E11	0x21	0x0303
0x0d	0x6000	0x22	0x0404
0x0e	0x4c00	0x23	0x8001
0x0f	0x6609	0x24	0x4401
0x10	0x5F8F	0x25	0x0000
0x11	Read only	0x26	0x0000
0x12	Read only	0x27	0x0000
0x13	Read only	0x28	0x0000
0x14	Read only	0x29	0x0000

表 10.2　寄存器优化值

Address	Opt value
0x0a	0x2403
0x02	0x4060
0x03	0x5810
0x05	0x7fa6
0x0f	0x661d
0x0d	0x6003
0x1a	0x00f7

11.典型应用电路

RC816 典型应用电路（SPI 模式）

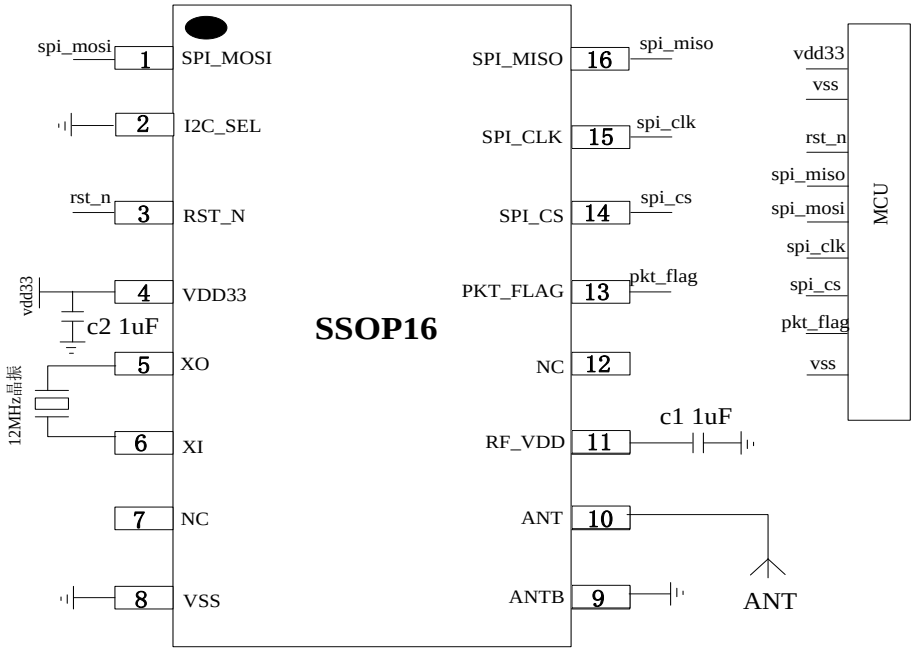


图 11.2 RC816（SPI）典型应用电路

RC816 典型应用电路（I2C 模式）

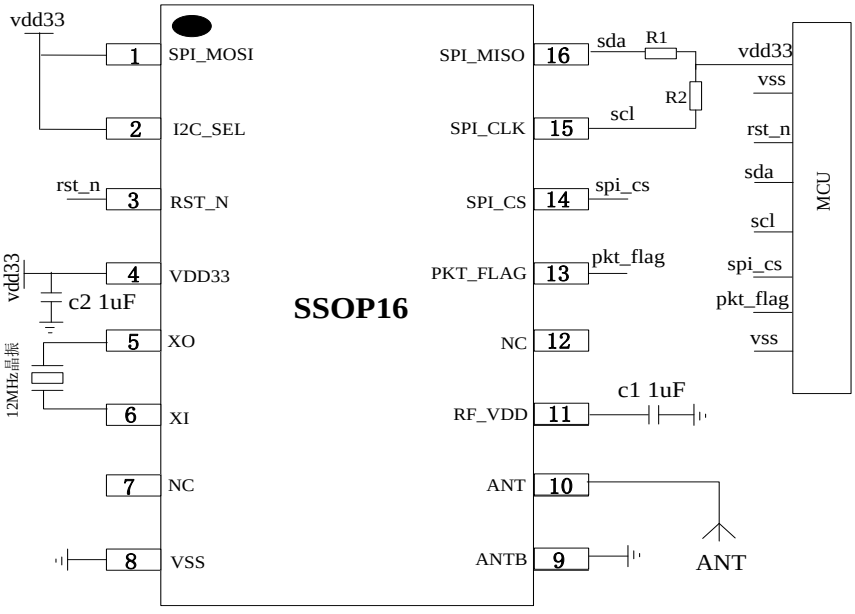


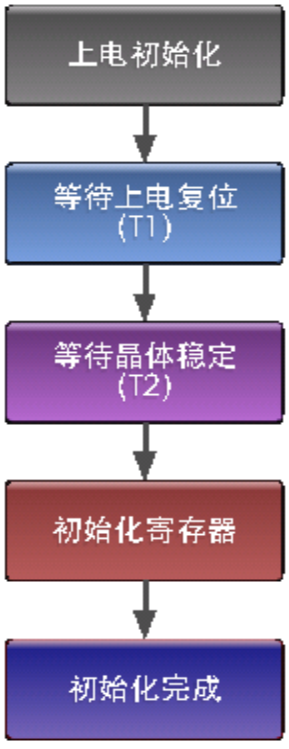
图 11.3 RC816（I2C 地址 A4=1）典型应用电路

注：I2C 模式下，芯片内部在 SPI_MISO 和 SPI_CLK 脚分别集成了 4.7K 上拉电阻，外部上拉电阻可根据实际应用情况来加或不加。

12. 注意事项

12.1 上电和寄存器初始化数据

1. 芯片内部集成上电复位功能（POR）（如果只用内部复位，RST_N 脚只需要拉高或悬空），T1 时间为上电复位时间，约 0.5ms
2. T2 是晶振稳定时间，约为 1.5ms，然后由 MCU 初始化寄存器
3. 寄存器初始化完成后，芯片可以开始发射或接收数据
4. 除了内部自动复位，芯片还有其它两种复位方式：
   a、RST_N 引脚复位：RST_N 为低时对芯片进行复位
   b、软件复位：先写 0x1e[0]寄存器为‘1’，使能软件复位功能，再对 0x01[7]寄存器写‘1’就可以完成复位操作。采用内部 POR 和软件复位功能，可以省去 RST_N 引脚以节省 MCU 资源，此时 RST_N 悬空或上拉至 VDD33 即可。



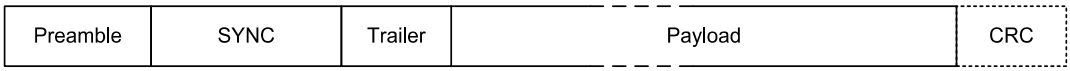
初始化流程（采用内部 POR）

12.2 进入 sleep mode 和唤醒

当 SPI_CS拉高，并且MCU 写寄存器 0x01[15] 为“1”后, 芯片进入 sleep mode, 此时消耗电流<1uA, sleep mode下寄存器的值能够保持。当SPI_CS拉低后,芯片自动唤醒进入IDLE状态。MCU 要拉低SPI_CS一段时间(等待晶体稳定，约1.5ms), 再写 SPI 数据。

12.3 数据包格式

空中数据包格式



- Preamble: 1~8 bytes, 可设置.
- SYNC: 16/32/48/64 bits, 可设置.
- Trailer: 4~18 bits, 可设置.
- Payload: TX/RX data. There are 4 data types:
  - Raw data
  - 8bit/10bitlinecode
  - Manchester
  - Interleave with FEC option
- CRC: 16-bit CRC is optional.

12.4 FIFO 长度

默认是 64 个字节，接收和发送共用，也可设置成分开使用各 32 个字节
把 0x23[15]设置为 0 即各使用 32 字节

12.5 清空 FIFO 指针

接收 FIFO 和发送 FIFO 的读写指针可通过对 0x25，0x26 相应寄存器写‘1’来清除。
当接收到数据包,读指针将指示 FIFO 中有多少 bytes 数据。
当接收器收到 SYNC 时，接收 FIFO 写指针会自动清 0。
当发射器发送完数据后，发射 FIFO 读指针会自动清 0。

12.6 数据包 payload 长度

两种方式去确定 TX/RX 包长度。当寄存器 0x01[2]=1 时，内部状态机会根据 payload 第一个 byte 数据来检测包长度。如要发 8 个字节，第一个字节应写 8，总字长为 9 个 字节。当寄存器 0x01[2]=0，第一个 byte 数据没什么特殊意义。数据包长度将由 TX FIFO 何时为

空或者何时清空 TX_EN 来决定，见下表。

数据包长度

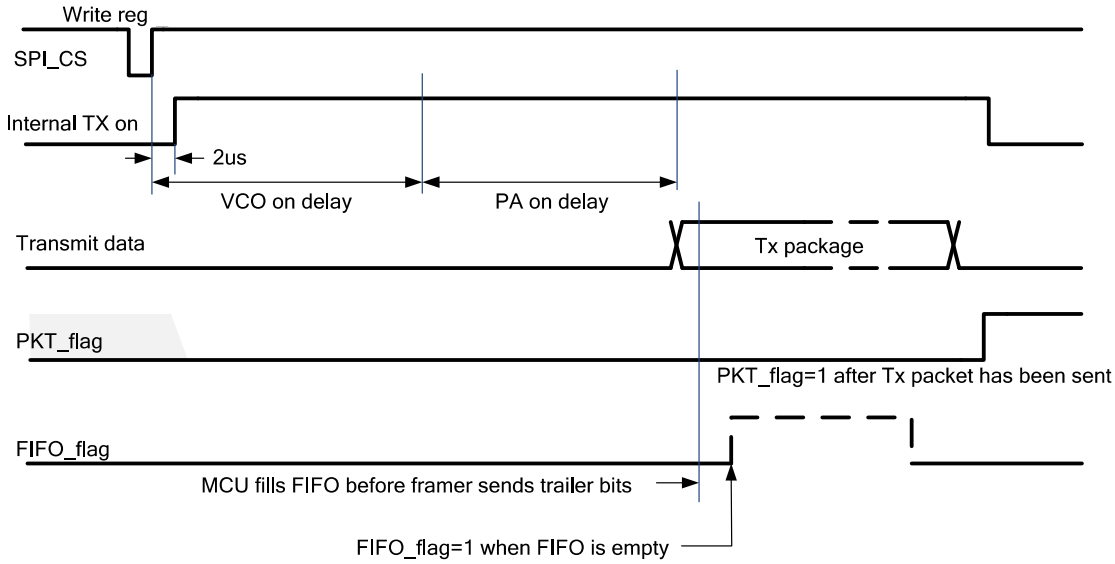
0x01[2] pack_length_en	0x01[1] fw_term_tx	
0	0	当TX_EN=0 时, 终止发射。 当RX_EN=0 时, 终止接收。
	1	当FIFO 为空时, 自动终止发射。 当RX_EN=0 时, 终止接收。
1	x	Payload 第一个字节表示包长度, 0 到 255bytes。当发完所有的数据后, 发射自动终止。

12.7 状态机决定包长度

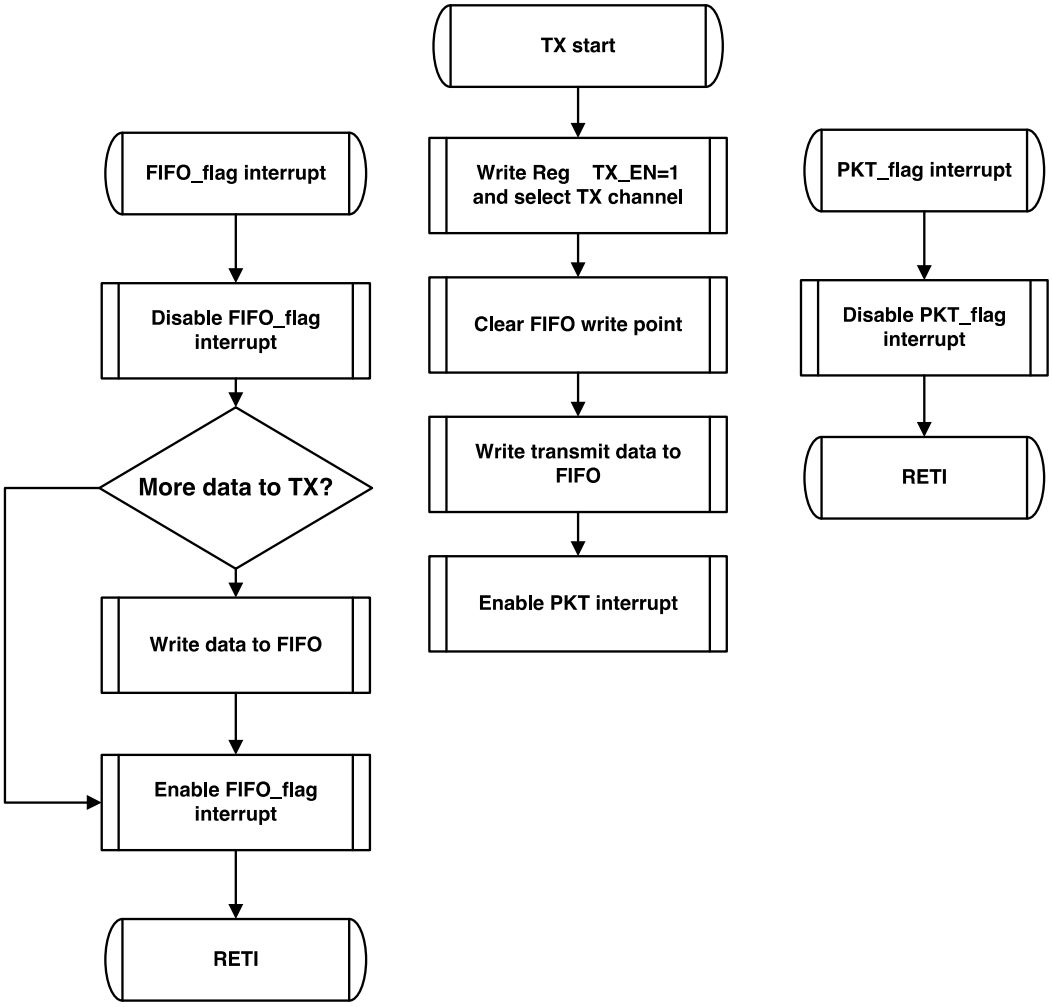
当寄存器0x01[2]=1 时，payload 的第一个 byte 表示包长度，最大长度是 255 bytes。

12.7.1 发射时序

下面显示 TX 时序.：当 MCU 将寄存器 0x00[15]写为‘1’后，同时写寄存器 0x00[6:0]设定好发射信道，芯片将自动根据 payload 来产生包。MCU 需要在发射 trailer 前写入发射数据。 如果包长度超过 FIFO 长度，MCU 需要多次写 FIFO 数据。FIFO flag (reg0x12[13]) 表示 FIFO 是不是为空。



TX 时序图



MCU 发送数据包流程图
FIFO 和 PKT flags 作为 MCU 的中断信号

在上面的流程图中，先使能发送状态机，再写发送数据到发送 FIFO 中，这样可以提高 MCU 效率，但要保证数据包在发送 TRAILER 之前写入所有数据；如果要写入的时间需要很长，则应先写好所有数据再启动发送状态。

//////////////// 发射流程例子：

1， 初始化寄存器，即写入需要优化的寄存器值

```
write reg[0x0a] = 0x2403;
write reg[0x02] = 0x4060;
write reg[0x03] = 0x5810;
write reg[0x05] = 0x7fa6;
write reg[0x0f] = 0x661d;
write reg[0x0d] = 0x6003;
write reg[0x1a] = 0x00f7;
```

2, 清空发送 FIFO

```
write reg[0x26] = 0x8080;
```

3, 写数据到 FIFO（假如发送数据：0x05 0x01 0x02 0x03 0x04 0x05, 第一个字节表示长度）

```
write reg[0x27] = 0x0501;
write reg[0x27] = 0x0203;
write reg[0x27] = 0x0405;
```

4, 启动发送使能，同时设置频道

write reg[0x00] = 0x80XX;//低 7 位为频道号

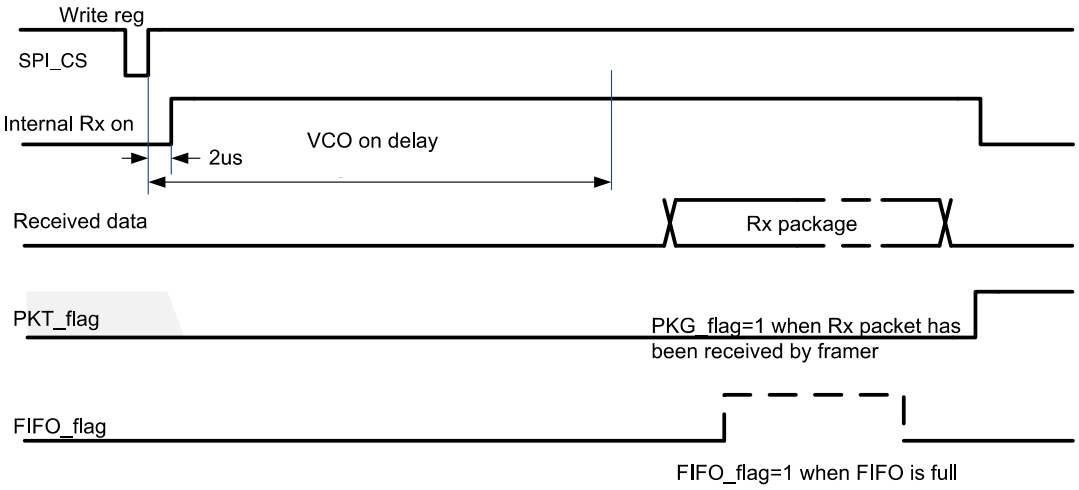
5，等待 pkt_flag 为高，表示发送完成

12.7.2 接收时序

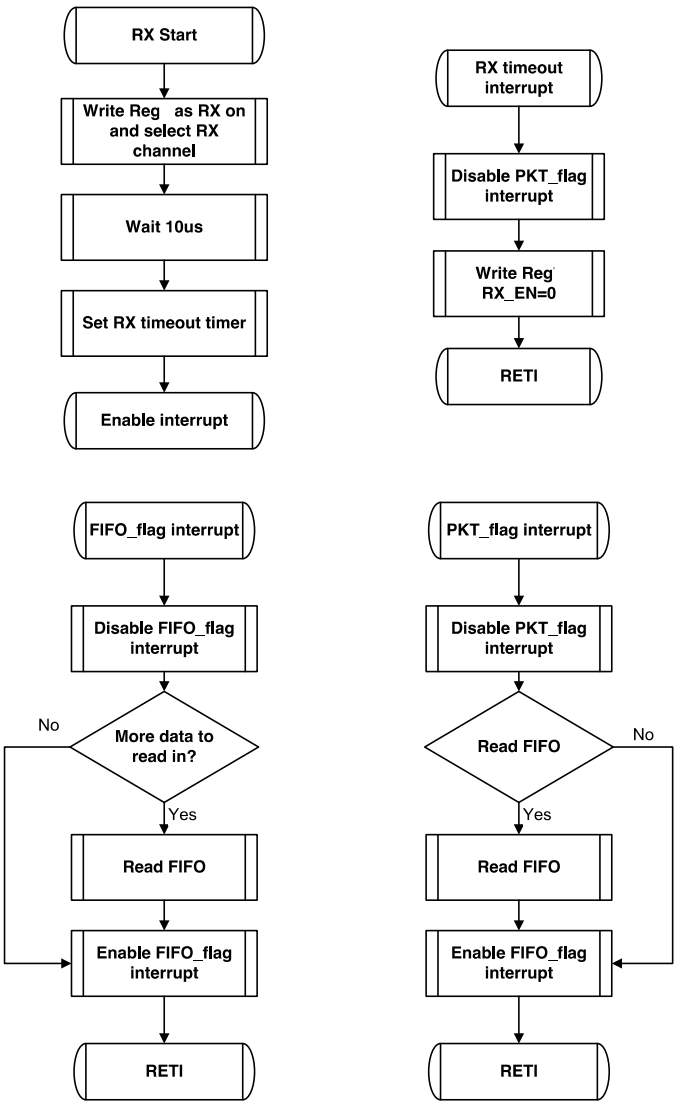
下面显示 RX 接收时序。当 MCU 将寄存器 0x00[14]写为 1 并且选择好接收器信道，芯片将打开 RX 并等待正确的 syncword。当收到正确的 syncword，芯片将自动开始处理数据包。当数据包处理完毕，状态机将进入 IDLE。

当接收到的数据包长度长于 63 bytes，FIFO flag 将起作用，意味着 MCU 必须从 FIFO 中读取数据。

在弱信号、多径和远距离时，不一定能收到正确的 syncword。为了避免出现死机情况，MCU 需要做一个定时器。在大多数应用中，数据包是在一定时间窗口内可以收到的，如果没收到系统要有定时器恢复到正常模式。



接收时序图



接收流程图

///////// 接收流程例子：

1，初始化寄存器，即写入需要优化的寄存器值

write reg[0x0a] = 0x2403;
write reg[0x02] = 0x4060;
write reg[0x03] = 0x5810;
write reg[0x05] = 0x7fa6;
write reg[0x0f] = 0x661d;
write reg[0x0d] = 0x6003;
write reg[0x1a] = 0x00f7;

2, 清空接收 FIFO

write reg[0x25] = 0x8080;

4, 启动接收使能，同时设置频道

write reg[0x00] = 0x40XX; //低 7 位为频道号

5，等待 pkt_flag 为高，表示接收到一帧数据，开始从接收 FIFO 读数据,读出的第一个字节为数据长度

read reg[0x28]
read reg[0x28], 。。。。。。。

12.7.3 自动应答和自动重传

该功能可配置的寄存器有：

写寄存器 0x01[3] = 1，使能 auto-ack 功能。

写寄存器 0x0b[11:8] ，设置重发次数，默认值是 3，即第一次发送失败后，再重发 2 次。

写寄存器 0x0b[5:0], 设置等待 ACK 时间，默认是 0x3f, 每个 step 是 16us, 总共大约为 1ms，表示等待 1ms 时间如果没收到 ACK 包且重发次数没有到最大值就会重新发送数据包。

使能和不使能 AUTO-ACK 功能对应的 PKT_FLAG 的拉起时间也不一样：

发射方收到 ACK 包后或未收到 ACK 包但重发次数已最大，这时会拉起 PKT_FLAG，退出发射状态，MCU 在 PKT_FLAG 拉高后，可读寄存器 0x16[7]位，如果为 1 即收到 ACK 包。

接收方在收到数据包后再发出 ACK 包，然后退出接收状态，拉起 PKT_FLAG 信号。

ACK 包的时间约为 150us, 这由 preamble 和 syncword 的长度决定。

12.7.4 CRC16 校验

该功能默认为开启。

写寄存器 0x01[13]开启或关闭 CRC 功能。

CRC 开启但 AUTO-ACK 功能没有开启时，芯片接收状态不受 CRC 状态影响，但寄存器会有 CRC error 状态位。PKT_FLAG 拉起后，MCU 可读 0x16[15]位判断 CRC 是否正确，来决定是否需要读该帧数据。

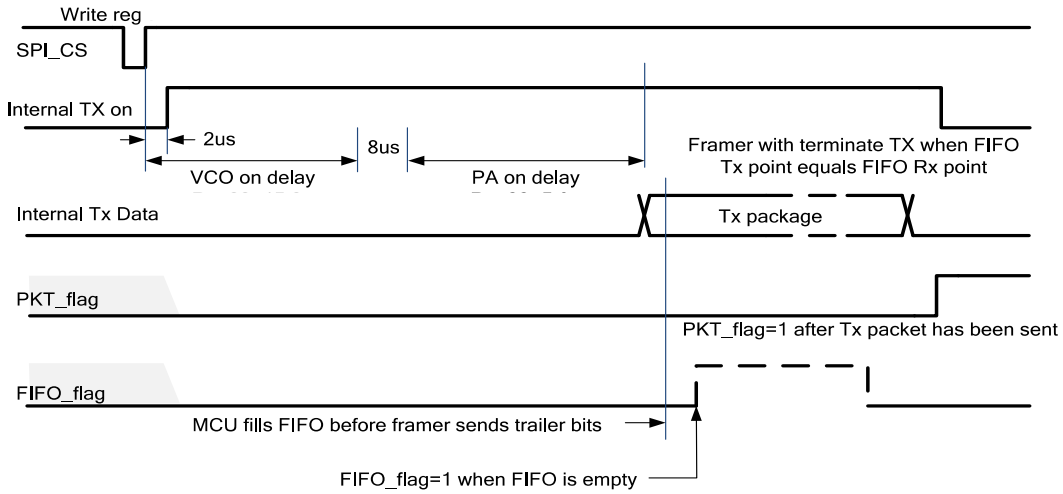
CRC 开启且 AUTO-ACK 功能也开启，这时芯片会自动判断 CRC 是否正确，如果接收方 PKT_FLAG 拉起，说明收到一帧数据包且 CRC 正确。

12.8 应用层决定包长度

当寄存器 0x01[2]=0 时，payload 第一个字节没有特殊意义，包长度由寄存器 0x01[1]的状态决定。

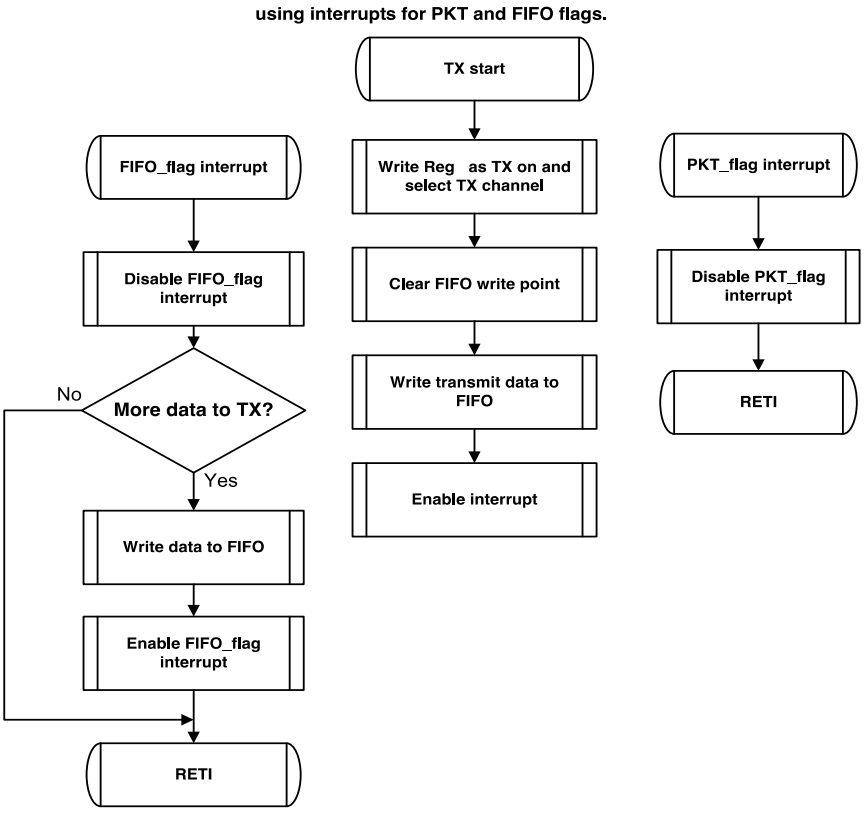
12.8.1 fw_term_tx=1

当寄存器 0x01[1]=1，在发射数据时，系统将比较发送 FIFO 的写指针和读指针，如果 MCU 停止向发送 FIFO 写数据，芯片将会最终探测到 FIFO 何时为空，然后自动退出发射状态。时序图如下：



注意：当寄存器 0x01[1]=1, 不要让 FIFO 过空或过满，FIFO full/empty 阀值可以通过寄存器

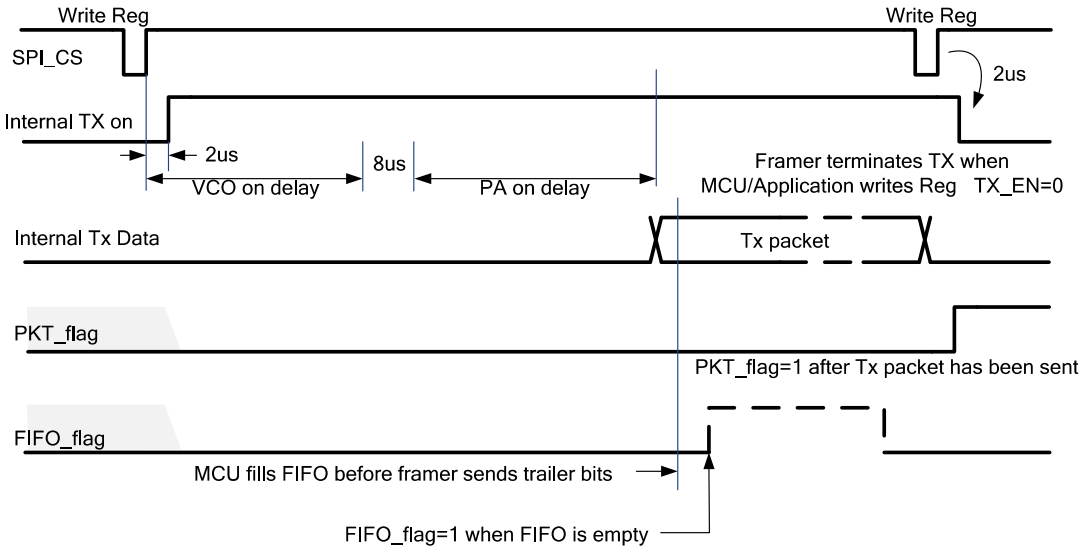
FIFO THRESHOLD 设置，最优值由 SPI 速度和 MCU 读写 FIFO 速度决定。



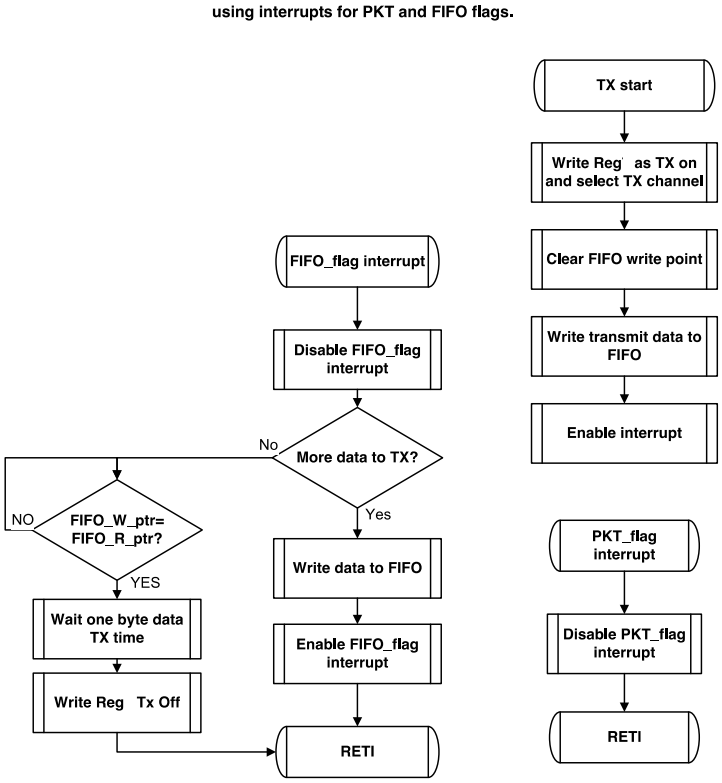
寄存器 0x01[2:1]=2’b01 时发射的流程图

12.8.2 fw_term_tx=0(TX)

当寄存器 0x01[2:1]=2’b00 时，系统只有在写寄存器 0x00[15] tx_en=0 时停止发射。



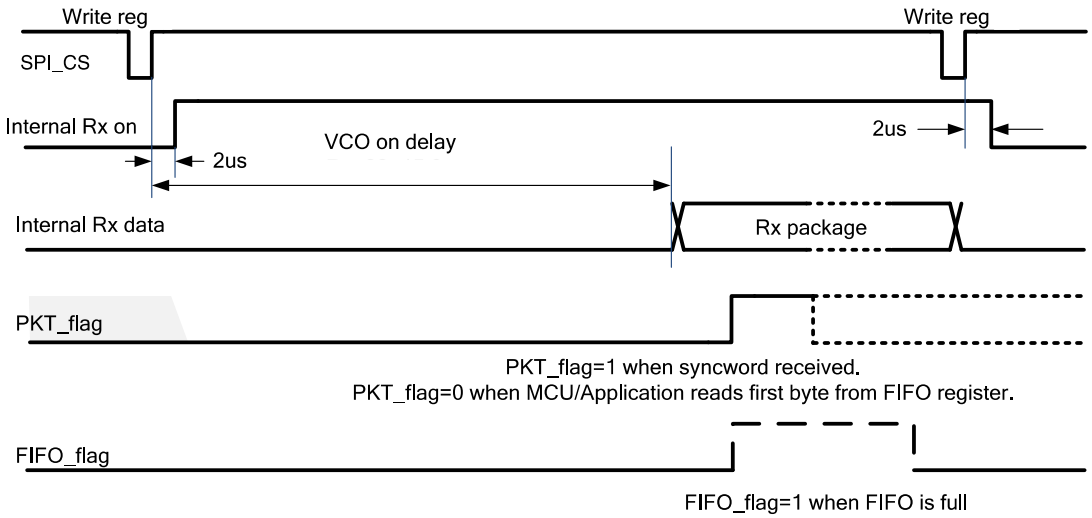
当寄存器 0x01[2:1]=2’b00 时，TX 时序图



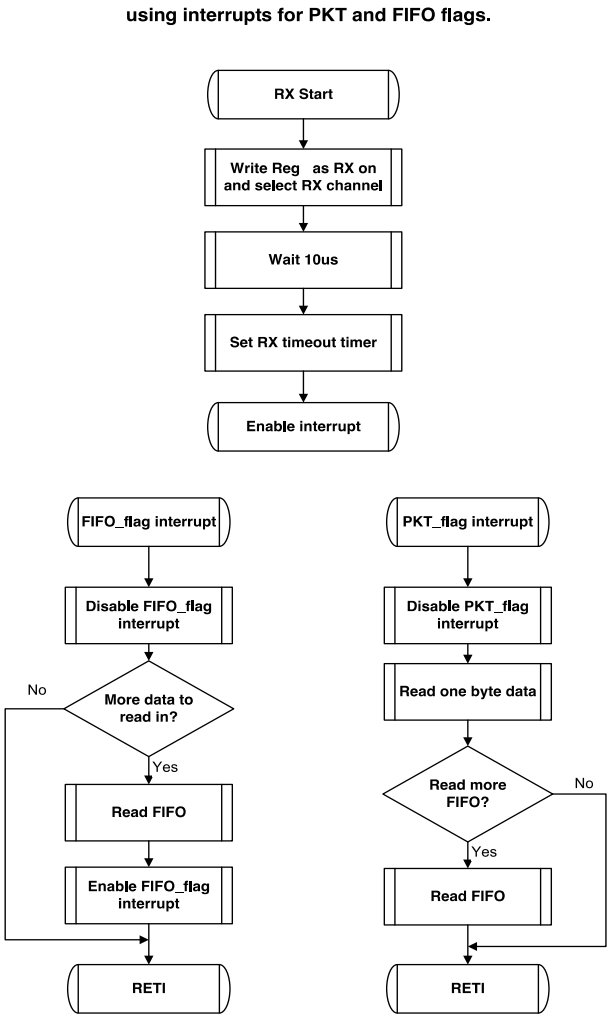
当寄存器 0x01[2:1]=2'b00 时，TX 流程图

12.8.3 fw_term_tx=0 (RX)

当 Reg0x01[2]=0 时，芯片将会在 Reg0x00[14]写为 1（RX_EN=1） 时开始接收包，此时芯片将会自动将 RX 设定到固定的频道接收。在等待一定时间使内部时钟和 RX 电路稳定后，芯片开始在收到的信号中寻找 syncword，一旦找到，将拉高 PKT_FLAG，并向 FIFO 里写收到的数据。PKT_FLAG 将一直为高直到 MCU 将 FIFO 中的数据读完。当 MCU 把数据读完后，PKT_FLAG 将拉低直到下个 TX/RX 周期。
当 Reg0x01[2:1]= 'b00 or 'b01 时，必须由 MCU 将 Reg0x00[14]写为 0 才能退出 RX 状态。



当 Reg0x01[2:1]='b00 or 'b01 时，RX 时序图



当 Reg0x01[2:1]=’b00 or ‘b01 时，RX 流程图

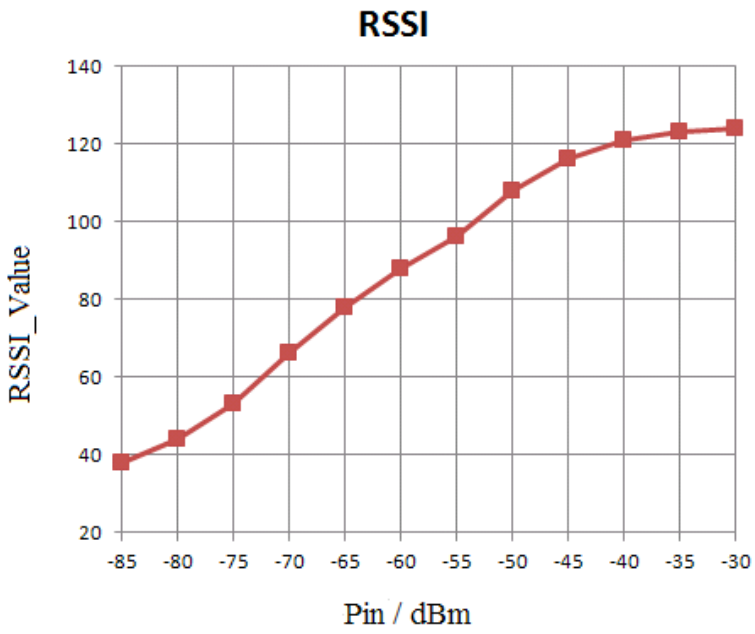
12.8.4 PA 输出功率设置

PA 的输出功率可以通过寄存器 PA_PW_SET[3:0]，即 reg0x02[3:0]来设置，reg0x02[3:0]是一个无符号数，数值越大，PA 输出功率越小。最大输出功率可到 6dBm 左右，对应 reg0x02[3:0]=0000，最小输出功率约为-18dBm 左右，对应 reg0x02[3:0]=1111。

12.8.5 RSSI

RC816 支持 RSSI 检测功能，在 RX 状态下，可通过 reg0x11[15:8]直接读取实时的 RSSI 值，其检测方法如下：
(1)、将芯片设置在 RX 状态；
(2)、先将 reg0x1c[15]写为 1，关闭 RSSI 以消除之前留存的 RSSI 值。然后再将 reg0x1c[15]写为 0，开启 RSSI 检测功能，等待 60us 之后就可以通过 reg0x11[15:8]读取当下的 RSSI 值。

下图是芯片 RSSI 检测功能的实测性能：



13. FAQ

1、如何排查 SPI/I2C 接口不工作的问题

a、先检查芯片电源是否正常，VDD33 脚是否有电压（3.3V）输入，RF_VDD 脚是否有 1.8V 左右的电压输出；
b、检查晶振是否起振，芯片上电复位后默认是 IDLE 状态，晶振开启，耗电约为 0.6mA，此时用万用表量 XI 和 XO 脚的电压应该在 0.9V 左右，用示波器探头点在 XOUT 脚上应该能看到 12MHz 正弦波（如果示波器探头电容过大、阻抗过低可能会导致晶振不起振，建议测量时将探头打在 X10 的档位）；
c、程序跑起来后，查看数字接口上是否有信号；
d、如果上述条件都正常，接口时序也没问题，应该就能正常读写寄存器。

2、芯片主要有哪几种状态

芯片主要有 4 种状态：
1、 IDLE 状态：芯片上电（复位后）默认为 IDLE 状态，此时晶振开启，耗电约为 0.6mA；
2、 RX 状态：写寄存器 reg0x00=0x40XX（XX 代表信道号）即可进入 RX 状态；
3、 TX 状态：写寄存器 reg0x00=0x80XX（XX 代表信道号）即可进入 TX 状态；
4、 Sleep 状态：将 reg0x01[15]写为“1”然后拉高 SPI_CS 即可进入 sleep 状态，此时芯片耗电约为 1uA，在 sleep 状态下，寄存器的值仍然可以保持。

3、如何验证 TX 功能正常以及如何测试 PA 输出功率

a、芯片上电复位，并按照说明书要求将优化值写入相应的寄存器（见表 10.2）；
b、写寄存器 reg0x01=0x2070，并写寄存器 reg0x00=0x8030 启动 TX，用频谱仪测，ANT 脚上应该有 2.45GHz 的信号输出（也可以设置其它信道号进行测试）。
c、改变寄存器 reg0x02 的值可以调节 PA 输出功率的大小；

4、典型收发流程

(1)、发送过程：

a、初始化寄存器，即写入需要优化的寄存器值
write reg[0x0a] = 0x2403;
write reg[0x02] = 0x4060;

write reg[0x03] = 0x5810;
write reg[0x05] = 0x7fa6;
write reg[0x0f] = 0x661d;
write reg[0x0d] = 0x6003;
write reg[0x1a] = 0x00f7;

b、清空发送 FIFO
write reg[0x26] = 0x8080

c、写数据到 FIFO，如发送的数据为：0x05 0x01 0x02 0x03 0x04 0x05（第一个字节表示长度，后面 5 个字节是数据，总长度为 6 个字节），则：
write reg[0x27] = 0x0501;
write reg[0x27] = 0x0203;
write reg[0x27] = 0x0405;

d、启动发送使能，同时设置频道
write reg[0x00] = 0x80XX; // 低 7 位为频道号

e、等待 pkt_flag 为高，表示发送完成

(2)、接收过程:

a、初始化寄存器，即写入需要优化的寄存器值
write reg[0x0a] = 0x2403;
write reg[0x02] = 0x4060;
write reg[0x03] = 0x5810;
write reg[0x05] = 0x7fa6;
write reg[0x0f] = 0x661d;
write reg[0x0d] = 0x6003;
write reg[0x1a] = 0x00f7;

b、清空接收 FIFO
write reg[0x25] = 0x8080;

c、启动接收使能，同时设置频道号
write reg[0x00] = 0x40XX; //低 7 位为频道号

d、等待 pkt_flag 为高，表示接收到一帧数据，此时可以通过读取 reg0x16[15]来判断 CRC 是否正确（reg0x16[15]=0 表示 CRC 正确，如果 CRC 出错，该位会置为“1” ，但再次进入接收时 reg0x16[15]会自动清零），如果 CRC 正确，说明接收到了正确的数据，则 MCU 可以从接收 FIFO 中读取所需的数据，读出的第一个字节为数据长度。
read reg[0x28]
read reg[0x28]
。。。。。。。

注：CRC 开启但 AUTO_ACK 没有开启的条件下（芯片复位后默认状态），接收过程中 pkt_flag 的拉高不受 CRC 影响，就是说 pkt_flag 拉高不能保证收到的数据包一定是正确的，在某些条件下，就算没有发送方发送数据过来，接收方接收到的噪声或干扰也可能刚好和同步码匹配从而拉高 pkt_flag。因此，pkt_flag 拉高但 CRC 不正确时，应该重新进入接收流程（清 FIFO、使能接收。。。。。。）。

5、PKT_FALG 的状态如何变化

启动 TX 后，PKT_FLAG 会自动拉低，等数据发送完成后 PKT_FLAG 拉高，进入 IDLE 之后 PKT_FLAG 依然保持为高直到再次启动 TX 或 RX。

在默认配置下，启动 RX 后，PKT_FLAG 会自动拉低，然后芯片会一直搜索 syncword，一旦接收到正确的 syncword，芯片会继续接收、处理后续的数据包，等数据包接收完成后，PKT_FLAG 变高并一直维持高电平状态直到再次启动 TX 或 RX。

6、SYNCWORD 该如何设

在接收状态下，芯片只有收到正确的 syncword 后才会继续接收、处理后续的数据包，因此 syncword 的设置对接收性能有一定影响。通过 reg0x03[12:11]可以设置 syncword 的长度，

通过 reg0x1f~reg0x22 可以设置 syncword 的具体数据，通过 reg0x24[5:0]可以设置 syncword 的容错 bit 数（如设为 2，则收到的 syncword 错 1 bit 也会收到后续的数据）。如果 syncword 设得很长，容错设得很低，则有可能因为 syncword 出错而漏接发送端发过来的数据，如果 syncword 设得很短，容错设得很高，则接收到的噪声或干扰容易和 syncword 匹配上而错接数据。因此，syncword 的设置要根据具体的应用环境来定，一般情况下，长度设为 48bit，容错 1bit，效果较好。

7、如何使用 auto-ack 和重发功能

写寄存器 0x01[3] = 1，使能 auto-ack 功能。写寄存器 0x0b[11:8]，设置重发次数，默认值是 3 次。开启该功能后，接收方在收到数据后会自动返回一个 ACK 信号给发送方，发送方发送完数据后会自动进入接收状态，等待接收方返回的 ACK 信号，如果在一定的时间内（该时间可以通过 0x0b[5:0]设置，默认值是 0x3f，约 1ms）没有收到 ACK 信号，则再次发送一遍数据直到接收到 ACK 信号或重发次数达到最大值。以上过程由芯片内部自动完成，无需 MCU 干预。

如果发送方开启了 auto-ack 和重发功能，但接收方没有开启，则发送方始终无法收到 ACK 信号，等待一定时间之后肯定会再次重发相同的数据直到重复次数达到最大值。该设置可作为硬件重发功能。

开启 auto-ack 和重发功能后，PKT_FLAG 拉高的时间和没开启该功能时的拉高时间不太一样：发送方在收到 ACK 信号或重发次数达到最大时才将 PKT_FLAG 拉高并退出发送状态，PKT_FLAG 变高后可以读寄存器 0x16[7]，如果为“1”则表示收到 ACK 信号。接收方在收到数据包并自动返回 ACK 信号后拉高 PKT_FLAG，退出接收状态。

在速率要求不高的情况下，建议打开 FEC 功能以提升性能。通过写寄存器 reg0x03[5:4]来开启 FEC13 或 FEC23，FEC13 比 FEC23 具有更好的性能，但效率也较低，收发同样的数据需要更长的时间。

8、sleep 状态如何进入，如何唤醒，唤醒需要多少时间

将 reg0x01[15]写为“1”，然后拉高 SPI_CS 即可进入 sleep 状态。SPI_CS 拉低一段时间就可从 sleep 状态唤醒进入 IDLE 状态，唤醒时间主要由晶振起振、稳定的时间决定，一般在 2ms 左右，待晶振起振稳定之后，就可以对芯片进行操作。

9、如何防止 FIFO 溢出

在接收或发送长数据包时，为了防止 FIFO 溢出，可通过 reg0x24 寄存器设置相应的阈值，超过该阈值后，芯片会给出指示，该指示可以通过 reg0x12[13] (FIFO_FLAG)来读取。

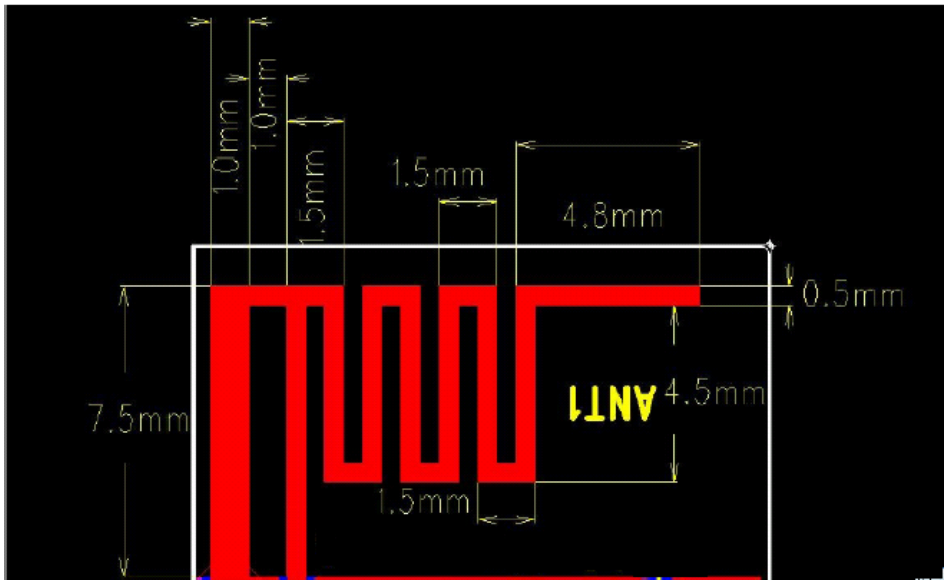
10、RF 电路的 PCB 板需要注意什么

首先要保证 RF 芯片的电源稳定、干净。电源上的稳压和去耦电容尽量靠近芯片的管脚，电源线和地线尽量粗短。可用大面积的铜层作为地线用，没有用到的地方可以和地线相连作为地线用。如果 RF 芯片 3.3V 供电电源的噪声或干扰比较大，建议经过 RC 或 LC 滤波之后再给 RF 芯片供电。

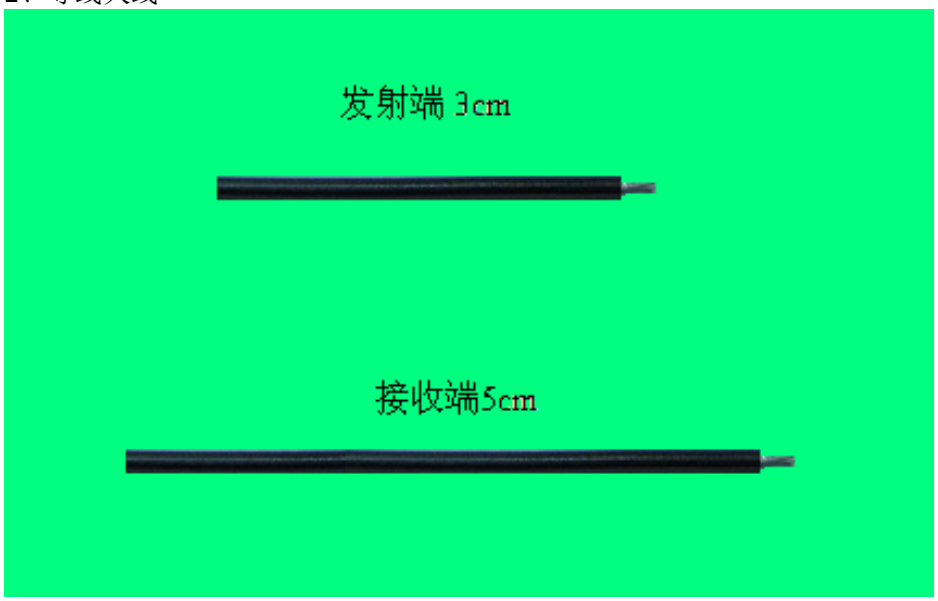
晶振尽量靠近 XI 和 XO 脚，连接晶振的线尽量短以减小噪声干扰和分布电容对晶振的影响。晶振外壳应良好接地以减小对外辐射提高抗干扰能力。

推荐几款天线如下：

1、板载天线



2、导线天线



注：导线天线的最佳长度和具体的 PCB 布局有关，可以通过实测找到最佳值。

11、提升芯片工作可靠性的说明

为了提升芯片工作可靠性以及产品良率，建议做以下改进：

1、0xoa 寄存器的优化值配置为 0x2403；

2、在每次启动收发之前，0x07 寄存器先为 0x3f11，再写为 0x7f11，确保每次接收或发送时，VCO 都工作在最佳的 band。

示例：

```
……..
write reg[0x07] = 0x3f11;
write reg[0x07] = 0x7f11;
write reg[0x00] = 0x80XX(TX)  或  write reg[0x00] = 0x40XX(RX)   [XX 是信道号]
……..
```

14. 版本更新历史

版本号	创建时间	创建内容	创建者	备注
V1.0	2015.4.20	初始文档 V1.0		
V1.1	2015.8.10	寄存器 0x0a 优化值更新为：0x2053		
V1.2	2016.1.15	1、寄存器 0x03 优化值更新为：0x5810 2、完善寄存器描述、增加 FAQ		
V1.3	2018.10.25	1、寄存器 0x0a 优化值更新为：0x2403 2、在每次启动收发之前，0x07 寄存器先为 0x3f11，再写为 0x7f11		具体见 FAQ--11