

# UM2002B 用户手册

版本：V1.2



广芯微电子（广州）股份有限公司

<http://www.unicmicro.com/>

## 条款协议

本档的所有部分，其著作权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本档的全部或部分组件。本档没有任何形式的担保、立场表达或其他暗示，若有任何因本档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本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。

## 目录

1	系统概述 .....	1
1.1	应用场景 .....	1
2	功能框图 .....	2
3	封装及引脚定义 .....	3
3.1	封装引脚分布 .....	3
3.2	引脚功能描述 .....	3
4	寄存器定义 .....	5
4.1	Reg00 Address:0x00 Default:0x1D .....	5
4.2	Reg01 Address:0x01 Default:0x78 .....	5
4.3	Reg02 Address:0x02 Default:0x00 .....	5
4.4	Reg03 Address:0x03 Default:0x00 .....	5
4.5	Reg04 Address:0x04 Default:0x00 .....	6
4.6	Reg05 Address:0x05 Default:0x03 .....	6
4.7	Reg06 Address:0x06 Default:0x33 .....	6
4.8	Reg07 Address:0x07 Default:0x33 .....	6
4.9	Reg08 Address:0x08 Default:0x10 .....	6
4.10	Reg09 Address:0x09 Default:0x00 .....	6
4.11	Reg0A Address:0x0A Default:0x00 .....	7
4.12	Reg0B Address:0x0B Default:0x00 .....	7
4.13	Reg0D Address:0x0D Default:0x30 .....	7
4.14	Reg0E Address:0x0E Default:0x32 .....	8
4.15	Reg10 Address:0x10 Default:0x40 .....	8
4.16	Reg11 Address:0x11 Default:0x00 .....	9
4.17	Reg20 Address:0x20 Default:0x50 .....	10
4.18	Reg28 Address:0x28 Default:0x19 .....	10
4.19	Reg2B Address:0x2B Default:0xFF .....	11
4.20	Reg2C Address:0x2C Default:0xFF .....	11
4.21	Reg2D Address:0x2D Default:0x10 .....	11
4.22	Reg37 Address:0x37 Default:0x06 .....	11
4.23	Reg38 Address:0x38 Default:0x91 .....	12
4.24	Reg4B Address:0x4B Default:0x45 .....	12
4.25	Reg4C Address:0x4C Default:0xF4 .....	12
4.26	Reg4D Address:0x4D Default:0x10 .....	12
4.27	Reg4E Address:0x4E Default:0x45 .....	13
4.28	Reg4F Address:0x4F Default:0x67 .....	13

4.29	Reg50	Address:0x50	Default:0x00	.....	14
4.30	Reg51	Address:0x51	Default:0x00	.....	14
4.31	Reg52	Address:0x52	Default:0x00	.....	14
4.32	Reg56	Address:0x56	Default:0xB7	.....	14
4.33	Reg57	Address:0x57	Default:0x00	.....	14
4.34	Reg5D	Address:0x5D	Default:0x00	.....	15
4.35	Reg60	Address:0x60	Default:0x80	.....	15
4.36	Reg61	Address:0x61	Default:0x00	.....	15
4.37	Reg64	Address:0x64	Default:0x00	.....	15
4.38	Reg66	Address:0x66	Default:0x37	.....	16
4.39	Reg67	Address:0x67	Default:0x80	.....	16
4.40	Reg68	Address:0x68	Default:0x00	.....	16
4.41	Reg69	Address:0x69	Default:0x00	.....	16
4.42	Reg6A	Address:0x6A	Default:0x00	.....	16
4.43	Reg6B	Address:0x6B	Default:0x00	.....	17
4.44	Reg6C	Address:0x6C	Default:0x00	.....	17
4.45	Reg6D	Address:0x6D	Default:0x00	.....	17
4.46	Reg6E	Address:0x6E	Default:0x00	.....	17
4.47	Reg6F	Address:0x6F	Default:0x00	.....	17
4.48	Reg70	Address:0x70	Default:0xFF	.....	17
4.49	Reg71	Address:0x71	Default:0x09	.....	18
4.50	Reg72	Address:0x72	Default:0x01	.....	18
4.51	Reg74	Address:0x74	Default:0x03	.....	18
4.52	Reg75	Address:0x75	Default:0xA7	.....	19
4.53	Reg76	Address:0x76	Default:0xA7	.....	19
4.54	Reg77	Address:0x77	Default:0x98	.....	19
4.55	Reg78	Address:0x78	Default:0xF3	.....	19
4.56	Reg79	Address:0x79	Default:0x98	.....	20
4.57	Reg7A	Address:0x7A	Default:0xF3	.....	20
4.58	Reg7B	Address:0x7B	Default:0x98	.....	20
4.59	Reg7C	Address:0x7C	Default:0xF3	.....	20
4.60	Reg7D	Address:0x7D	Default:0x0E	.....	20
4.61	Reg7E	Address:0x7E	Default:0x00	.....	20
4.62	Reg7F	Address:0x7F	Default:0x00	.....	21
4.63	Reg80	Address:0x80	Default:0x1D	.....	21
4.64	Reg81	Address:0x81	Default:0x78	.....	21
4.65	Reg82	Address:0x82	Default:0x00	.....	22
4.66	Reg83	Address:0x83	Default:0x00	.....	22

4.67	Reg84 Address:0x84 Default:0x00 .....	22
4.68	Reg85 Address:0x85 Default:0x03 .....	22
4.69	Reg86 Address:0x86 Default:0x33 .....	22
4.70	Reg87 Address:0x87 Default:0x33 .....	22
4.71	Reg88 Address:0x88 Default:0x10 .....	23
4.72	Reg89 Address:0x89 Default:0x00 .....	23
4.73	Reg8A Address:0x8A Default:0x00.....	23
4.74	Reg8D Address:0x8D Default:0x30 .....	23
4.75	Reg94 Address:0x94 Default:0x08 .....	24
4.76	Reg95 Address:0x95 Default:0x20 .....	24
4.77	Reg96 Address:0x96 Default:0x26 .....	24
4.78	Reg97 Address:0x97 Default:0x1F .....	24
4.79	Reg98 Address:0x98 Default:0x68 .....	25
4.80	Reg99 Address:0x99 Default:0x04 .....	25
4.81	Reg9A Address:0x9A Default:0xAE .....	25
4.82	Reg9B Address:0x9B Default:0x89.....	25
4.83	Reg9C Address:0x9C Default:0x84.....	25
4.84	RegA0 Address:0xA0 Default:0x50.....	25
4.85	RegA3 Address:0xA3 Default:0x42.....	26
4.86	RegE0 Address:0xE0 Default:0x80 .....	26
4.87	RegE1 Address:0xE1 Default:0x00.....	26
5	功能描述 .....	27
5.1	接收机.....	27
5.2	频率综合器 .....	27
5.3	RSSI .....	28
5.3.1	RSSI相关寄存器 .....	29
5.3.2	自动计算RSSI.....	29
5.3.3	手动调节增益.....	30
5.3.4	RSSI自动过滤功能 .....	31
5.4	系统复位 .....	31
5.4.1	上电复位POR.....	32
5.4.2	外部复位nRST.....	33
5.4.3	软件复位 .....	33
6	芯片运行.....	34
6.1	状态机控制图 .....	34
6.2	工作模式 .....	34
6.3	状态机说明 .....	35

6.4	GPIO和中断 .....	35
6.4.1	GPIO的配置 .....	36
6.4.2	中断的配置和映射.....	37
6.5	FIFO缓冲区.....	38
6.5.1	FIFO相关的寄存器 .....	38
6.5.2	FIFO的工作模式.....	38
6.5.3	FIFO的中断时序.....	39
6.5.4	FIFO的应用场景.....	39
7	数据处理机制.....	40
7.1	直通模式 (Direct) .....	40
7.1.1	Direct相关寄存器.....	40
7.2	数据包 (Packet) .....	40
7.2.1	Mode相关寄存器 .....	41
7.2.2	MODE 0 .....	41
7.2.3	MODE 1 .....	41
7.2.4	MODE 2 .....	42
7.2.5	MODE 3 .....	42
7.3	Preamble配置.....	43
7.3.1	Preamble配置相关寄存器 .....	43
7.4	Sync Word配置 .....	44
7.4.1	Sync Word相关的寄存器.....	44
7.5	Length配置.....	45
7.5.1	Length相关的寄存器 .....	45
7.5.2	MODE 0的length配置 .....	46
7.5.3	MODE 1的length配置 .....	46
7.5.4	MODE 2的length配置 .....	46
7.5.5	MODE 3的length配置 .....	47
7.6	Address配置 (仅Mode 3) .....	52
7.6.1	Address的相关寄存器 .....	52
7.7	seqnum配置 (仅Mode 3) .....	53
7.7.1	seqnum相关的寄存器.....	53
7.8	Payload Data配置 .....	53
7.8.1	Payload Data相关的寄存器.....	53
7.8.2	Payload_bit_order的示例 .....	54
7.8.3	数据白化示例 .....	55
7.9	CRC配置 .....	56
7.9.1	CRC相关的寄存器.....	56
7.9.2	CRC配置说明 .....	57

7.9.3	常用CRC标准和多项式 .....	58
7.9.4	配置CRC8和CRC16示例 .....	59
8	接口说明 .....	60
8.1	SPI接口 .....	60
9	应用参考 .....	62
9.1	应用参考电路图 .....	62
9.2	参考参数 .....	63
10	封装尺寸 .....	64
10.1	QFN20 (4*4mm) .....	64
11	版本维护 .....	65

## 表目录

表 3-1: 引脚功能说明 .....	3
表 5-1: RSSI相关寄存器列表 .....	29
表 5-2: Reg2B增益控制表 .....	31
表 6-1: 工作模式 .....	34
表 6-2: GPIO配置相关寄存器列表 .....	36
表 6-3: GPIO输出功能描述 .....	37
表 6-4: FIFO相关寄存器列表 .....	38
表 7-1: Direct相关寄存器列表 .....	40
表 7-2: Mode相关寄存器列表 .....	41
表 7-3: Preamble配置相关寄存器列表 .....	43
表 7-4: Sync配置相关寄存器列表 .....	44
表 7-5: Length相关的寄存器列表 .....	45
表 7-6: Address配置相关寄存器列表 .....	52
表 7-7: M3_ADDR_SIZE选择 .....	52
表 7-8: seqnum相关寄存器列表 .....	53
表 7-9: Payload Data相关寄存器列表 .....	53
表 7-10: CRC相关的寄存器列表 .....	56
表 7-11: 常用CRC标准和多项式列表 .....	58
表 8-1: SPI时序参数 .....	61
表 9-1: 参考参数表 .....	63



## 图目录

图 2-1: 功能框图 .....	2
图 3-1: QFN20封装引脚分布图 .....	3
图 5-1: RSSI线性图 .....	30
图 6-1: 状态机控制图 .....	34
图 7-1: Mode 0帧格式 .....	41
图 7-2: Mode 1帧格式 .....	42
图 7-3: Mode 2帧格式 .....	42
图 7-4: Mode 3帧格式 .....	43
图 7-5: Mode 0 .....	46
图 7-6: Mode 1 .....	46
图 7-7: Mode 2 .....	46
图 7-8: m3_length_en=0时MODE 3帧格式 .....	47
图 7-9: 可变长度包格式, Address不存在 .....	48
图 7-10: Address存在, Address在Length Byte之前 .....	49
图 7-11: Address存在, Address在Length Byte之后 .....	50
图 7-12: PAYLOAD_BIT_ORDER操作 .....	55
图 7-13: CRC编码范围 .....	57
图 7-14: CRC_INV .....	58
图 7-15: CRC_BIT_ORDER .....	58
图 8-1: SPI读写寄存器时序图 .....	60
图 8-2: SPI读写FIFO时序图 .....	61
图 9-1: 应用参考电路图 .....	62
图 10-1: QFN20封装图 .....	64

# 1 系统概述

UM2002B 是一款工作于 200 ~ 960MHz 范围内的低功耗、高性能、单片集成的(G)FSK/OOK 无线接收机芯片。内部集成完整的射频接收机、频率综合器、解调器，只需配备简单、低成本的外围器件就可以获得良好的接收性能。

芯片支持灵活可设的数据包格式，支持数据直通模式，支持跳频操作，同时集成了 FEC 功能。外部 MCU 可通过 SPI 对芯片进行控制，并访问内部的 256 bytes 的 RX FIFO。

## 1.1 应用场景

- 工业传感
- 安防系统
- 无线标签，无线门禁
- 遥控装置，无线玩具
- 智能交通，智慧城市，智能家居
- 智能门锁，无线监控等智能传感器终端应用

## 2 功能框图

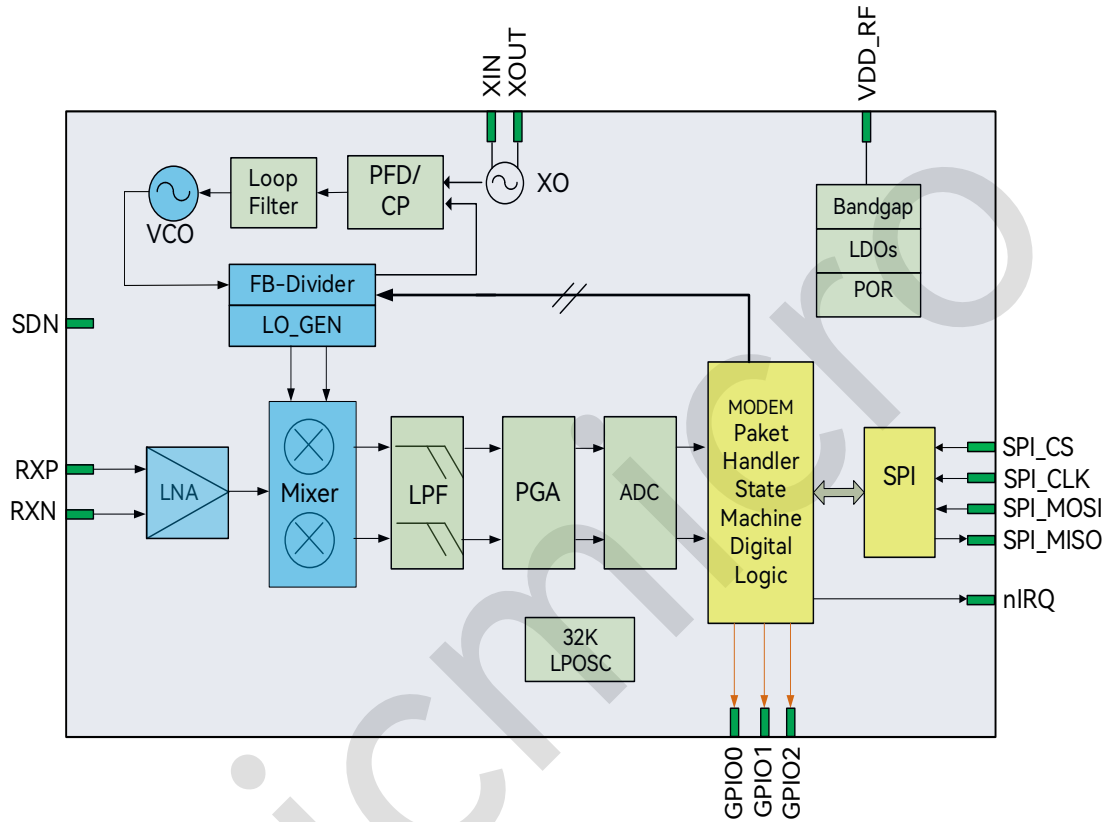


图 2-1: 功能框图

## 3 封装及引脚定义

### 3.1 封装引脚分布

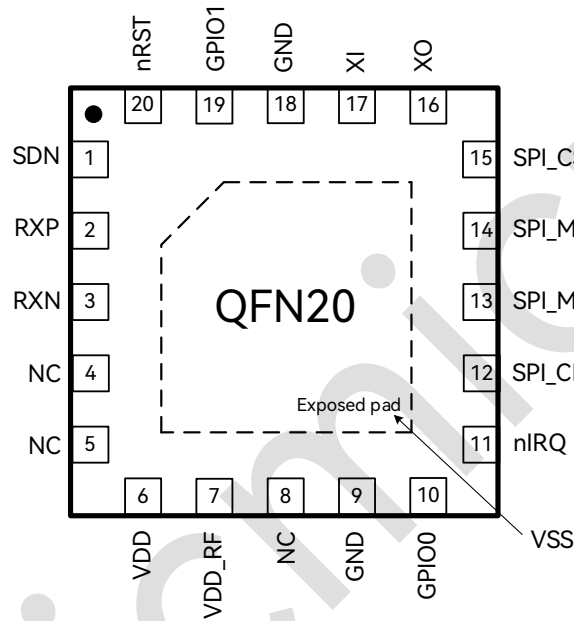


图 3-1: QFN20 封装引脚分布图

### 3.2 引脚功能描述

表 3-1: 引脚功能说明

引脚编号	引脚名称	IO Type	功能描述
0	VSS	G	芯片地(LF PAD)公共地
1	SDN	DI	芯片关断使能, SDN 高电平时芯片处于关断模式
2	RXP	RFI	射频正端输入
3	RXN	RFI	射频负端输入
4	NC	NC	空脚, 未连接任何内部电路
5	NC	NC	空脚, 未连接任何内部电路
6	VDD	P	1.8V~3.6V 电源输入
7	VDD_RF	P	LDO 供电电源

引脚编号	引脚名称	IO Type	功能描述
8	NC	NC	空脚，未连接任何内部电路
9	GND	G	芯片地
10	GPIO0	DIO	可配置 GPIO0
11	nIRQ	DIO	可配置 GPIO，默认为中断输出
12	SPI_CLK	DI	SPI 时钟
13	SPI_MISO	DO	SPI 数据输出
14	SPI_MOSI	DIO	SPI 数据输入（或三线制输入输出）
15	SPI_CS	DI	片选信号
16	XO	AO	晶振输出
17	XI	AI	晶振输入
18	GND	G	芯片地
19	GPIO1	DIO	可配置 GPIO1
20	nRST	DIO	可配置 GPIO2，默认为外部复位引脚

说明：RF - 射频信号；A - 模拟信号；D - 数字信号；I - Input；O - Output；G - Ground；P - Power。

## 4 寄存器定义

### 4.1 Reg00 Address:0x00 Default:0x1D

Bit	Name	Type	Description	Default
7	reserved	R/W	-	1'b 0
6:0	rf_f0[30:24]	R/W	信道频率 f0[30:24]。 信道频率 rf_freq 设置公式： $rf\_freq=rf\_f0+ch\_num*f\_step$ rf_f0 由{Reg00,Reg01,Reg02,Reg03}配置， 单位为 MHz，低 20bit 为小数； ch_num 由 Reg04 配置； 步进频率 f_step 由{Reg05,Reg06,Reg07}配置， 单位为 MHz，低 20bit 为小数。	7'h1D

### 4.2 Reg01 Address:0x01 Default:0x78

Bit	Name	Type	Description	Default
7:0	rf_f0[23:16]	R/W	信道频率 f0[23:16]	8'h78

### 4.3 Reg02 Address:0x02 Default:0x00

Bit	Name	Type	Description	Default
7:0	rf_f0[15:8]	R/W	信道频率 f0[15:8]	8'h00

### 4.4 Reg03 Address:0x03 Default:0x00

Bit	Name	Type	Description	Default
7:0	rf_f0[7:0]	R/W	信道频率 f0[7:0]	8'h00

#### 4.5 Reg04 Address:0x04 Default:0x00

Bit	Name	Type	Description	Default
7:0	ch_num	R/W	信道号 ch_num, 参考 Reg00 描述	8'h00

#### 4.6 Reg05 Address:0x05 Default:0x03

Bit	Name	Type	Description	Default
7:0	f_step[23:16]	R/W	信道步进频率 f_step[23:16], 单位为 MHz, 低 20bit 为小数, 参考 Reg00 描述	8'h03

#### 4.7 Reg06 Address:0x06 Default:0x33

Bit	Name	Type	Description	Default
7:0	f_step[15:8]	R/W	信道步进频率 f_step[15:8], 参考 Reg00 描述	8'h33

#### 4.8 Reg07 Address:0x07 Default:0x33

Bit	Name	Type	Description	Default
7:0	f_step[7:0]	R/W	信道步进频率 f_step[7:0], 参考 Reg00 描述	8'h33

#### 4.9 Reg08 Address:0x08 Default:0x10

Bit	Name	Type	Description	Default
7	reserved	R/W	-	1'b0
6:0	ref_freq[30:24]	R/W	参考频率 (晶振频率) 设置, ref_freq 由 {Reg08[6:0],Reg09,Reg0A,Reg0B}, 单位为 MHz, 低 24bit 为小数部分。	7'h10

#### 4.10 Reg09 Address:0x09 Default:0x00

Bit	Name	Type	Description	Default
7:0	ref_freq[23:16]	R/W	参考 Reg08 描述	8'h00

### 4.11 Reg0A Address:0x0A Default:0x00

Bit	Name	Type	Description	Default
7:0	ref_freq[15:8]	R/W	参考 Reg08 描述	8'h00

### 4.12 Reg0B Address:0x0B Default:0x00

Bit	Name	Type	Description	Default
7:0	ref_freq[7:0]	R/W	参考 Reg08 描述	8'h00

### 4.13 Reg0D Address:0x0D Default:0x30

Bit	Name	Type	Description	Default
7	payload_bit_order	R/W	Payload bit 顺序: 0: LSB 低位在前 1: MSB 高位在前	1'b0
6	manchester_inv	R/W	曼彻斯特编码: 0: 上升沿为编码 1, 下降沿为编码 0 1: 下降沿为编码 1, 上升沿为编码 0	1'b0
5	syncword_en	R/W	Syncword 同步字使能: 0: disable 1: enable	1'b1
4	preamble_en	R/W	设置 Preamble 前导码使能: 0: disable 1: enable	1'b1
3:2	pkt_enc_type	R/W	数据包编码: 00: NRZ 01: 曼彻斯特编码 10: 无效 11: 交织编码	2'b00
1:0	fec_type	R/W	数据包 FEC 类型: 00: NO FEC 01: 1/3 FEC 10: 2/3 FEC 11: 1/2 FEC	2'b00



## 4.14 Reg0E Address:0x0E Default:0x32

Bit	Name	Type	Description	Default
7	length_byte_swap	R/W	包长度控制域为两个字节时, 高低字节顺序: 0: 低字节在前 1: 高字节在前	1'b0
6	length_sel	R/W	包长度控制域字节配置: 0: 1 字节 1: 2 字节	1'b0
5	crc_en	R/W	CRC 使能: 0: disable 1: enable	1'b1
4	scramble_en	R/W	数据白化 (扰码) 使能: 0: disable 1: enable	1'b1
3	fifo_share_en	R/W	FIFO 大小设置: 0: RX FIFO 128 字节 1: RX FIFO 256 字节	1'b0
2	reserved	R/W	-	1'b0
1:0	packet_mode	R/W	数据包控制模式: 00: 模式 0, 循环接收数据并写入 RX FIFO 中, 需要 MCU 写命令退出接收 01: 模式 1, 由寄存器控制数据包长度 10: 模式 2, RX FIFO 中的第 1 个或 2 个字节作为包长度 11: 模式 3, 含有 length, address, seqnum 等多种数据域的模式	2'b10

## 4.15 Reg10 Address:0x10 Default:0x40

Bit	Name	Type	Description	Default
7:4	reserved	R/W	-	4'h4
3	rss_val_sel	R/W	读取 RSSI 时, 选择读取的数据内容: 0: 经过计算的 RSSI 值 1: 原始信号幅度值	1'b0

2	preamble_int_en	R/W	接收 Preamble (前导码) 匹配中断使能: 0: disable 1: enable	1'b0
1	syncword_int_en	R/W	接收 Syncword (同步字) 匹配中断使能: 0: disable 1: enable	1'b0
0	int_pulse_sel	R/W	中断信号类型设置: 0: 电平中断 1: 脉冲中断 (1 $\mu$ s)	1'b0

#### 4.16 Reg11 Address:0x11 Default:0x00

Bit	Name	Type	Description	Default
7	int_polarity	R/W	中断信号极性选择: 0: high active 1: low active	1'b0
6	int_flag_clr	R/W	写'1'清除中断标志	1'b0
5	miso_tri_opt	R/W	spi 在非工作状态时, spi_miso 状态: 0: 高阻 1: 输出	1'b0
4:3	reserved	R/W	-	2'b00
2:0	brclk_sel	R/W	GPIO brclk 输出时钟选择: 000: rx_clk 接收 bit rate 时钟 001: xtal_clk 晶振时钟 010: xtal_clk/8 011: xtal_clk/16 100: 频综时钟 101: NC 110: 内部 RC32K 时钟 111: ADC_clk	3'b000

## 4.17 Reg20 Address:0x20 Default:0x50

Bit	Name	Type	Description	Default
7:6	frequency_band	R/W	工作频段设置: 00: 800MHz~1GHz 频段 01: 400MHz~500MHz 频段 10: 267MHz~350MHz 频段 11: 200MHz~250MHz 频段	2'b01
5:3	reserved	R/W	-	3'b010
2	clk_en	R/W	使能解调时钟数据恢复模块: 0: enable 1: disable	1'b0
1:0	reserved	-	-	2'b00

## 4.18 Reg28 Address:0x28 Default:0x19

Bit	Name	Type	Description	Default
7	rss_i_Reg_clr	R/W	写'1'清除被锁定的 RSSI	1'b0
6	rss_i_sel	R/W	收到同步字后锁定 RSSI 值: 0: disable 1: enable	1'b0
5	rx_rssi_thr_en	R/W	接收时判断 RSSI, 如果收到前导码的 RSSI 低于设置的 RSSI 门限值 (Reg2c), 会复位解调器重新接收: 0: disable 1: enable	1'b0
4	agc_en	R/W	AGC 使能: 0: disable 1: enable	1'b1
3:0	reserved	R/W	-	4'h9

## 4.19 Reg2B Address:0x2B Default:0xFF

Bit	Name	Type	Description	Default
7:6	reserved	R/W	-	2'b11
5:4	rxfe_gn	R/W	rxfe 增益控制	2'b11
3:2	filter_gn	R/W	Filter 增益控制	2'b11
1:0	pga_gn	R/W	PGA 增益控制	2'b11

## 4.20 Reg2C Address:0x2C Default:0xFF

Bit	Name	Type	Description	Default
7:0	rx_rssi_thr	R/W	接收 RSSI 门限值, 参考 Reg28[5]	8'hFF

## 4.21 Reg2D Address:0x2D Default:0x10

Bit	Name	Type	Description	Default
7:4	reserved	R/W	-	4'h01
3	ook_en	R/W	调制解调模式选择: 0: FSK 1: OOK	1'b0
2:0	reserved	R/W	-	3'b000

## 4.22 Reg37 Address:0x37 Default:0x06

Bit	Name	Type	Description	Default
7:0	csrst_len	R/W	在补偿功能使能时, 找到信号的频谱后连续收到 Reg37*4 个前导后, 则认为找到有效的 Preamble	8'h06

### 4.23 Reg38 Address:0x38 Default:0x91

Bit	Name	Type	Description	Default
7:5	reserved	R/W	-	3'b100
4	demod_comp_en	R/W	解调频偏补偿: 0: disable 1: enable	1'b1
3:0	reserved	R/W	-	4'h1

### 4.24 Reg4B Address:0x4B Default:0x45

Bit	Name	Type	Description	Default
7:6	reserved	R/W	-	2'b01
5	rc32k_cal_en	R/W	RC32K 校准功能使能: 0: disable 1: enable	1'b0
4:0	reserved	R/W	-	5'h05

### 4.25 Reg4C Address:0x4C Default:0xF4

Bit	Name	Type	Description	Default
7:0	rc32k_std[7:0]	R/W	RC32K 校准值	8'hF4

### 4.26 Reg4D Address:0x4D Default:0x10

Bit	Name	Type	Description	Default
7	nirq_dir	R/W	nIRQ_flag IO 口方向: 0: output 1: input	1'b0
6	gpio_dir[0]	R/W	gpio[0] 方向: 0: output 1: input	1'b0
5	gpio_dir[1]	R/W	gpio[1] 方向: 0: output 1: input	1'b0

Bit	Name	Type	Description	Default
4	gpio_dir[2]	R/W	gpio[2] 方向, 默认复位输入功能: 0: output 1: input	1'b1
3:0	nirq_sel	R/W	nIRQ 引脚输出功能选择: 0x0: pkt_flag(包括数据包完成中断 pkt_int, 接收前导码中断 preamble_int, 接收同步字中断 syncword_int) 0x1: pkt_int 0x2: preamble_int 0x3: syncword_int 0x4: fifo_flag 0x5: brclk (参考 Reg11[2:0]描述) 0x6: rxdata 0x7: NC 0x8: NC 0x9: NC 0xa: rxdata 0xb: NC 0xc: NC 0xd: 高电平 others: 低电平	4'h0

#### 4.27 Reg4E Address:0x4E Default:0x45

Bit	Name	Type	Description	Default
7:4	gpio_0_sel	R/W	gpio[0]输出功能选择, 参考 Reg4D[3:0]描述	4'h4
3:0	gpio_1_sel	R/W	gpio[1]输出功能选择, 参考 Reg4D[3:0]描述	4'h5

#### 4.28 Reg4F Address:0x4F Default:0x67

Bit	Name	Type	Description	Default
7:4	gpio_2_sel	R/W	gpio[2]输出功能选择, 参考 Reg4D[3:0]描述	4'h6
3:0	reserved	R/W	-	4'h7

## 4.29 Reg50 Address:0x50 Default:0x00

Bit	Name	Type	Description	Default
7:0	rx_fifo_wr_ptr	R/W	RX FIFO 写指针, 写 0x80 清除指针	8'h00

## 4.30 Reg51 Address:0x51 Default:0x00

Bit	Name	Type	Description	Default
7:0	rx_fifo_rd_ptr	R/W	RX FIFO 读指针, 写 0x80 清除指针	8'h00

## 4.31 Reg52 Address:0x52 Default:0x00

Bit	Name	Type	Description	Default
7:0	rx_fifo_data	R/W	RX FIFO	8'h00

## 4.32 Reg56 Address:0x56 Default:0xB7

Bit	Name	Type	Description	Default
7:0	delta_rssi	R/W	计算 RSSI 时的预偏移量	8'hB7

## 4.33 Reg57 Address:0x57 Default:0x00

Bit	Name	Type	Description	Default
7:3	reserved	R/W	-	5'h00
2	spi_3w_sel	R/W	三线 SPI 选择: 0: 4 线标准 SPI 1: 3 线 SPI, 数据线只使用 SPI_MOSI	1'b0
1	crc_err_st	R/W	接收 CRC 错误时, 数据包的处理方法: 0: 正常完成该包的接收, 退出接收状态发出数据包中断和错误 CRC 标志 1: 丢弃该包数据并重新进入接收状态	1'b0
0	reserved	R/W	-	1'b0

### 4.34 Reg5D Address:0x5D Default:0x00

Bit	Name	Type	Description	Default
7	nrst_sel	R/W	nRST 引脚功能设置: 0: nRST 1: GPIO2	1'b0
6	pull_up_en	R/W	nRST 引脚 Pull-UP 使能: 0: enable 1: disable	1'b0
5:0	reseved	R/W	-	5'h00

### 4.35 Reg60 Address:0x60 Default:0x80

Bit	Name	Type	Description	Default
7:0	command	R/W	0x80: IDLE command 0x20: RX command 0x05: StandBy command 0x06: Reset command, 需要先使能 RegA3[7] 0x10: FS command 打开频综	8'h80

### 4.36 Reg61 Address:0x61 Default:0x00

Bit	Name	Type	Description	Default
7	syncword_rec	R	接收到正确的 syncword	1'b0
6	preamble_rec	R	接收到有效的 preamble	1'b0
5	crc_error	R	高电平时表示 CRC 校验错误	1'b0
4	pkt_flag	R	pkt_flag	1'b0
3	fifo_flag	R	fifo_flag	1'b0
2:0	reserved	R	-	3'b000

### 4.37 Reg64 Address:0x64 Default:0x00

Bit	Name	Type	Description	Default
7:0	rssr	R	读出的值为实际 RSSI 的绝对值	8'h00



### 4.38 Reg66 Address:0x66 Default:0x37

Bit	Name	Type	Description	Default
7	rc32k_cal_done	R	RC32k 校准完成	1'b0
6:0	rc32k_ftrim	R	RC32K 校准完成后的 trimming 值	7'h37

### 4.39 Reg67 Address:0x67 Default:0x80

Bit	Name	Type	Description	Default
7	st_idle	R	IDLE 状态	1'b1
6	Reseved	R	-	1'b0
5	st_rx	R	接收状态	1'b0
4	st_fson	R	频综开启状态	1'b0
3:0	reseved	R	-	4'h0

### 4.40 Reg68 Address:0x68 Default:0x00

Bit	Name	Type	Description	Default
7:2	reseved	R	-	6'h0
1	m3_seqnum_ok	R	数据包模式 3 时, 接收 seqnum 匹配正确	1'b0
0	m3_addr_comp_ok	R	数据包模式 3 时, 接收 address 匹配正确	1'b0

### 4.41 Reg69 Address:0x69 Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_fcs2_value	R	数据包模式 3 时, 接收 FCS2 数据	8'h00

### 4.42 Reg6A Address:0x6A Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_payload_len[7:0]	R	数据包模式 3 时, 接收 length 数据	8'h00

#### 4.43 Reg6B Address:0x6B Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_payload_len[15:8]	R	数据包模式 3 时, 接收 length 数据	8'h00

#### 4.44 Reg6C Address:0x6C Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_addr[7:0]	R	数据包模式 3 时, 接收 address	8'h00

#### 4.45 Reg6D Address:0x6D Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_addr[15:8]	R	数据包模式 3 时, 接收 address	8'h00

#### 4.46 Reg6E Address:0x6E Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_addr[23:16]	R	数据包模式 3 时, 接收 address	8'h00

#### 4.47 Reg6F Address:0x6F Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_rx_addr[31:24]	R	数据包模式 3 时, 接收地址域数据	8'h00

#### 4.48 Reg70 Address:0x70 Default:0xFF

Bit	Name	Type	Description	Default
7:0	scramble_init_val[7:0]	R/W	scramble 初始化值	8'hFF

#### 4.49 Reg71 Address:0x71 Default:0x09

Bit	Name	Type	Description	Default
7	reserved	R/W	-	1'b0
6	srcamble_len	R/W	伪随机数长度选择: 0: PN9 1: PN7	1'b0
5	scramble_msb	R/W	选择首先输出的移位寄存器: 0: LSB 1: MSB	1'b0
4	scramble_type	R/W	伪随机数产生器类型: 0: 抽出模式 1: 插入模式	1'b0
3:1	scramble_poly	R/W	反馈抽头选择	3'b100
0	scramble_init_val[8]	R/W	scramble 初始化值最高位	1'b1

#### 4.50 Reg72 Address:0x72 Default:0x01

Bit	Name	Type	Description	Default
7:5	reserved	R/W	-	3'b000
4:0	sync_thres	R/W	同步字检测允许的误差位数	5'h01

#### 4.51 Reg74 Address:0x74 Default:0x03

Bit	Name	Type	Description	Default
7:6	reserved	R/W	-	2'b00
5	cont_rx_en	R/W	控制接收状态机: 0: 接收完成数据包后退出接收状态进入 IDLE 状态 1: 接收完成数据包后重新再次进入接收状态	1'b0
4	sync_man_en	R/W	同步字曼彻斯特编码使能: 0: disable 1: enable	1'b0
3	sync_bit_order	R/W	同步字 bit 顺序: 0: 字节低位 1: 字节高位 MSB	1'b0

Bit	Name	Type	Description	Default
2:0	sync_len	R/W	同步字长度: 000: {sync_id_1} 001: {sync_id_1, sync_id_2} 010: {sync_id_1, sync_id_2, sync_id_3} 011: {sync_id_1, sync_id_2, sync_id_3, sync_id_4} 100: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5} 101: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6} 110: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6, sync_id_7} 111: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6, sync_id_7, sync_id_8}	3'b011

#### 4.52 Reg75 Address:0x75 Default:0xA7

Bit	Name	Type	Description	Default
7:0	sync_id_1	R/W	同步字字节 1	8'hA7

#### 4.53 Reg76 Address:0x76 Default:0xA7

Bit	Name	Type	Description	Default
7:0	sync_id_2	R/W	同步字字节 2	8'hA7

#### 4.54 Reg77 Address:0x77 Default:0x98

Bit	Name	Type	Description	Default
7:0	sync_id_3	R/W	同步字字节 3	8'h98

#### 4.55 Reg78 Address:0x78 Default:0xF3

Bit	Name	Type	Description	Default
7:0	sync_id_4	R/W	同步字字节 4	8'hF3

**4.56 Reg79 Address:0x79 Default:0x98**

Bit	Name	Type	Description	Default
7:0	sync_id_5	R/W	同步字节 5	8'h98

**4.57 Reg7A Address:0x7A Default:0xF3**

Bit	Name	Type	Description	Default
7:0	sync_id_6	R/W	同步字节 6	8'hF3

**4.58 Reg7B Address:0x7B Default:0x98**

Bit	Name	Type	Description	Default
7:0	sync_id_7	R/W	同步字节 7	8'h98

**4.59 Reg7C Address:0x7C Default:0xF3**

Bit	Name	Type	Description	Default
7:0	sync_id_8	R/W	同步字节 8	8'h F3

**4.60 Reg7D Address:0x7D Default:0x0E**

Bit	Name	Type	Description	Default
7:0	payload_len[7:0]	R/W	payload 长度，在数据包模式 1 和数据包模式 3 有效	8'h0E

**4.61 Reg7E Address:0x7E Default:0x00**

Bit	Name	Type	Description	Default
7:0	payload_len[15:8]	R/W	payload 长度，在数据包模式 1 和数据包模式 3 有效	8'h00

## 4.62 Reg7F Address:0x7F Default:0x00

Bit	Name	Type	Description	Default
7:1	reserved	R/W	-	7'h0
0	page_sel	R/W	寄存器页地址: 0: Reg00~Reg7E 1: Reg80~RegFE	1'b0

## 4.63 Reg80 Address:0x80 Default:0x1D

Bit	Name	Type	Description	Default
7:6	reserved	R/W	-	2'b00
5	m3_length_en	R/W	数据包模式 3 时, 数据长度域控制: 0: disable 1: enable	1'b 0
4	m3_crc_sel	R/W	数据包模式 3 时, CRC 的校验范围: 0: All payload 1: 仅 FIFO 中的数据	1'b 1
3:0	reserved	R/W	-	4'hD

## 4.64 Reg81 Address:0x81 Default:0x78

Bit	Name	Type	Description	Default
7:6	reserved	R/W	-	2'b 01
5:4	m3_addr_size	R/W	数据包模式 3 时, 地址长度: 00: 1 字节 01: 2 字节 10: 3 字节 11: 4 字节	2'b 11
3:2	reserved	R/W	-	2'b 10
1	m3_addr_pos_sel	R/W	数据包模式 3 时, 地址位置选择: 0: address 放在 length 之后 1: address 放在 length 之前	1'b 0
0	m3_addr_en	R/W	数据包模式 3 时, 地址域控制: 0: disable 1: enable	1'b 0

**4.65 Reg82 Address:0x82 Default:0x00**

Bit	Name	Type	Description	Default
7:0	m3_addr_val[7:0]	R/W	数据包模式 3 时, 地址值	8'h00

**4.66 Reg83 Address:0x83 Default:0x00**

Bit	Name	Type	Description	Default
7:0	m3_addr_val[15:8]	R/W	数据包模式 3 时, 地址值	8'h00

**4.67 Reg84 Address:0x84 Default:0x00**

Bit	Name	Type	Description	Default
7:0	m3_addr_val[23:16]	R/W	数据包模式 3 时, 地址值	8'h00

**4.68 Reg85 Address:0x85 Default:0x03**

Bit	Name	Type	Description	Default
7:0	m3_addr_val[31:24]	R/W	数据包模式 3 时, 地址值	8'h03

**4.69 Reg86 Address:0x86 Default:0x33**

Bit	Name	Type	Description	Default
7:0	m3_addr_bit_mask[7:0]	R/W	数据包模式 3 时, 地址过滤, mask 为 '1' 时 对应地址位不作比较	8'h33

**4.70 Reg87 Address:0x87 Default:0x33**

Bit	Name	Type	Description	Default
7:0	m3_addr_bit_mask[15:8]	R/W	数据包模式 3 时, 地址过滤, mask 为 '1' 时 对应地址位不作比较	8'h33

#### 4.71 Reg88 Address:0x88 Default:0x10

Bit	Name	Type	Description	Default
7:0	m3_addr_bit_mask[23:16]	R/W	数据包模式 3 时，地址过滤，mask 为‘1’时对应地址位不作比较	8'h10

#### 4.72 Reg89 Address:0x89 Default:0x00

Bit	Name	Type	Description	Default
7:0	m3_addr_bit_mask[31:24]	R/W	数据包模式 3 时，地址过滤，mask 为‘1’时对应地址位不作比较	8'h00

#### 4.73 Reg8A Address:0x8A Default:0x00

Bit	Name	Type	Description	Default
7:5	reserved	R/W	-	3'b 000
4	m3_seqnum_en	R/W	数据包模式 3 时，seqnum 数据使能： 0: disable 1: enable	1'b 0
3	m3_seqnum_bit_size	R/W	数据包模式 3 时，seqnum 数据长度： 0: 8bit 1: 16bit	1'b 0
2:0	reserved	R/W	-	3'b 000

#### 4.74 Reg8D Address:0x8D Default:0x30

Bit	Name	Type	Description	Default
7:1	reserved	R/W	-	7'h 30
0	m3_fcs2_en	R/W	数据包模式 3 时，FCS2 数据域使能： 0: disable 1: enable	1'b 0



### 4.75 Reg94 Address:0x94 Default:0x08

Bit	Name	Type	Description	Default
7:5	reserved	R/W	-	3'b000
4	crc_bit_order	R/W	CRC bit 顺序: 0: LSB 1: MSB	1'b0
3	crc_inv	R/W	CRC bit 取反: 0: disable 1: enable	1'b1
2	crc_man_en	R/W	CRC 曼彻斯特编码使能: 0: disable 1: enable	1'b0
1:0	crc_len	R/W	CRC 长度: 00: 8 位 01: 16 位 10: 24 位 11: 32 位	2'b00

### 4.76 Reg95 Address:0x95 Default:0x20

Bit	Name	Type	Description	Default
7:0	crc_poly[7:0]	R/W	CRC 多项式配置	8'h20

### 4.77 Reg96 Address:0x96 Default:0x26

Bit	Name	Type	Description	Default
7:0	crc_poly[15:8]	R/W	CRC 多项式配置	8'h26

### 4.78 Reg97 Address:0x97 Default:0x1F

Bit	Name	Type	Description	Default
7:0	crc_poly[23:16]	R/W	CRC 多项式配置	8'h1F

**4.79 Reg98 Address:0x98 Default:0x68**

Bit	Name	Type	Description	Default
7:0	crc_poly[31:24]	R/W	CRC 多项式配置	8'h68

**4.80 Reg99 Address:0x99 Default:0x04**

Bit	Name	Type	Description	Default
7:0	crc_init_val[7:0]	R/W	CRC 移位寄存器初始化值	8'h04

**4.81 Reg9A Address:0x9A Default:0xAE**

Bit	Name	Type	Description	Default
7:0	crc_init_val[15:8]	R/W	CRC 移位寄存器初始化值	8'hAE

**4.82 Reg9B Address:0x9B Default:0x89**

Bit	Name	Type	Description	Default
7:0	crc_init_val[23:16]	R/W	CRC 移位寄存器初始化值	8'h89

**4.83 Reg9C Address:0x9C Default:0x84**

Bit	Name	Type	Description	Default
7:0	crc_init_val[31:24]	R/W	CRC 移位寄存器初始化值	8'h84

**4.84 RegA0 Address:0xA0 Default:0x50**

Bit	Name	Type	Description	Default
7	crc_infifo_en	R/W	接收的 CRC 值是否需要写入 RX FIFO: 0: disable 1: enable	1'b0
6	reserved	R/W	-	1'b0
5:0	fifo_full_thr	R/W	RX FIFO 满门限, RX FIFO 数据高于门限值时 fifo_flag 置'1'	6'h50

**4.85 RegA3 Address:0xA3 Default:0x42**

Bit	Name	Type	Description	Default
7	srst_cmd_enable	R/W	Reset Command 使能: 0: disable 1: enable	1'b0
6:0	reserved	R/W	-	7'h42

**4.86 RegE0 Address:0xE0 Default:0x80**

Bit	Name	Type	Description	Default
7:0	rx_seqnum_val[7:0]	R	数据包模式 3 时, 接收 seqnum 数据	8'h80

**4.87 RegE1 Address:0xE1 Default:0x00**

Bit	Name	Type	Description	Default
7:0	rx_seqnum_val[15:8]	R	数据包模式 3 时, 接收 seqnum 数据	8'h00

## 5 功能描述

该芯片是一款高集成度的 sub-GHz 无线接收机。支持 OOK, 2-(G)FSK 解调方式, 支持 Direct 和 Packet 数据处理模式, 支持标准四线 SPI 和自定义三线 SPI 接口。

### 5.1 接收机

该芯片内部集成了低功耗、高性能的低中频架构接收机。天线接收到的射频信号经过低噪声放大器放大之后, 由正交混频器下变频至中频。I/Q 两路中频信号进一步通过低通滤波器 (LPF)、可编程放大器 (PGA) 放大到合适的幅度, 然后由 ADC 转换到数字域。

接收到的信号在数字域完成镜像抑制、中频滤波、基带滤波和 OOK/(G)FSK 解调。解调后的数据在直通模式下可以通过 GPIO 管脚直接输出, 也可以在包模式下通过 SPI 从内部 FIFO 中读取。

芯片的接收通路集成了 AGC 功能, Mixer、LPF、PGA 的增益受 AGC 环路调节。AGC 的环路控制在数字域完成, 通过设定合理的 AGC 控制参数, 芯片接收机的灵敏度、选择性和线性度可以达到最佳的性能。

### 5.2 频率综合器

该芯片内部集成了高精度的 Sigma-Delta 小数分频频率综合器, 在 200 ~ 960MHz 范围内精准的频率。工作频率可以通过两种方式进行设置:

1. 寄存器直接设置: 在这种方式下, 把所需的载波频率值直接写入相应的寄存器即可。
2. 查表方式。在这种方式下, 可以按照以下公式进行频率设置:

$$F_c = F_0 + N * F_{step}$$

其中,  $F_0$ 、 $N$  和  $F_{step}$  都可以通过寄存器进行设置。

- $F_0$  由{Reg00[6:0],Reg01,Reg02,Reg03}共 31bits 设置, 高 11 位表示整数, 低 20 位表示小数, 单位为 MHz。
- N 为整数由 Reg04 设置。
- $F_{step}$  由{Reg05,Reg06,Reg07}设置, 低 20bits 为小数, 单位为 MHz。

如设置  $F_c=433.92\text{MHz}$ , 可以设置  $F_0 = 433.92$ ,  $N = 0$ ,  $F_{step} = 0$ 。即写入以下寄存器值:

Reg00=0x1B

Reg01=0x1E

Reg02=0xB8

Reg03=0x51

Reg04=0x00

Reg05=0x00

Reg06=0x00

Reg07=0x00

### 5.3 RSSI

芯片内部集成的输入信号强度指示 (RSSI) 功能可以对天线端接收到的信号强度进行评估。RSSI 检测必须在 RX 状态下进行, 检测到的是信道内的信号强度。

UM2002B 内置信号检测模块, 用于接收数据包时测量 ADC 的数据信号强度 RSSI\_RAW, 在 AGC(自动增益控制)使能时, 使用 RSSI\_RAW 值对接收链路增益自动调节, 同时可自动计算出 RF 输入端的 RSSI(信号强度指示)值。也可以使用 RSSI\_RAW 值手动调节 (AGC 关闭) 接收链路增益, 并根据增益值自行计算 RSSI。RSSI\_RAW 值或自动计算的 RSSI 值可以从寄存器 Reg64 读取。

### 5.3.1 RSSI 相关寄存器

RSSI 相关寄存器如下表所示:

表 5-1: RSSI 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x10	3	R/W	rss_val_sel	读取 RSSI 时, 选择读取的数据内容: 0: 经过计算的 RSSI 值 1: 原始信号幅度值
0x28	6	R/W	rss_sel	收到同步字后锁定 RSSI 值: 0: disable 1: enable
	5	R/W	rx_rssi_thr_en	接收时判断 RSSI, 如果收到前导码的 RSSI 低于设置的 RSSI 门限值 (Reg2c), 会复位解调器重新接收 0: disable 1: enable
	4	R/W	agc_en	AGC 使能: 0: disable 1: enable
0x2B	5:0	R/W	rx_gain	每个 step 为 6dBm, AGC 使能时该寄存器只读。
0x2C	7:0	R/W	rx_rssi_thr	接收 RSSI 门限值, 参考 Reg28[5]
0x56	7:0	R/W	delta_rssi	计算 RSSI 时的预偏移量
0x64	7:0	R	rssi	读出的值为实际 RSSI 的绝对值

### 5.3.2 自动计算 RSSI

寄存器优化值设置:

- Reg10[3] = 0
- Reg28[4] = 1
- Reg56 = 0xB7

在进入接收状态后, 可以读取 Reg64 寄存器的值 rssi\_value[7:0], 每个 bit 对应 0.5dBm, 且是真实信号 RSSI 的绝对值。

例如读取 Reg64 = 0x65, 则  $RSSI = -(rssi\_value/2) = -50.5dBm$ 。

如下图  $F_{RF}=433.92\text{MHz}$ ,  $DR=10\text{kbps}$ ,  $F_{DEV}=22\text{K}$  实测的 RSSI 值对应的输入功率。从图上看  $[-120\text{dBm}, -35\text{ dBm}]$  的线性区间内, 输入功率值与读出的 RSSI 值有很好的对应关系, 如果在某些频段或速率在线性区间内读出的值与输入功率值有固定差值, 可以写寄存器 Reg56 去微调偏移量。

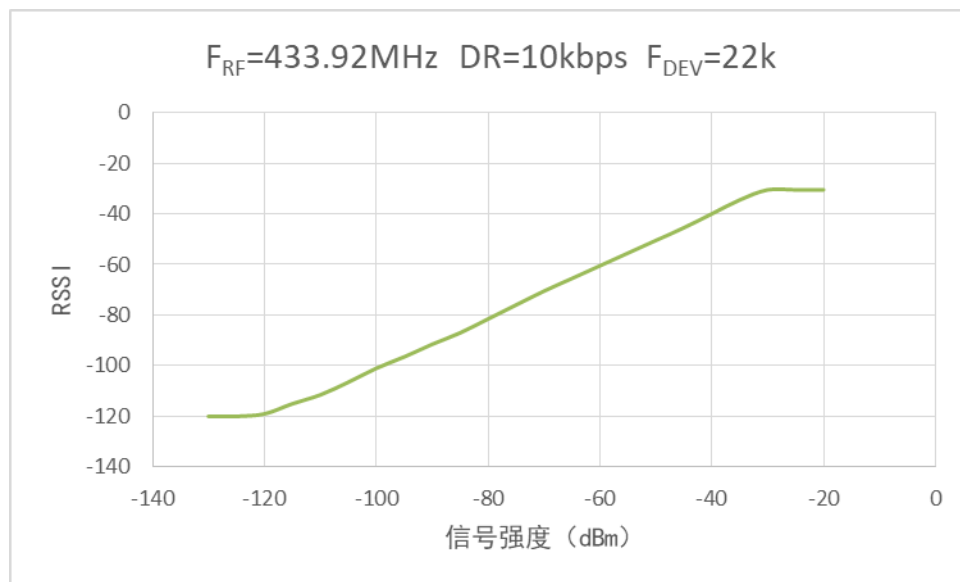


图 5-1: RSSI 线性图

UM2002B 在整个接收数据包期间, RSSI 值实时更新到寄存器 Reg64, 当退出接收状态 Reg64 值也恢复到 0; 如果设置寄存器 Reg28[6]=1, 则收到同步字后寄存器 Reg64 值锁定, RSSI 值不再更新, 直到下次重新进入接收时 RSSI 才会被更新。

### 5.3.3 手动调节增益

在正常 GFSK 模式时, 都使用自动增益调节。在 OOK 模式时, 可以手动调节增益并计算 RSSI 值。寄存器优化值设置如下:

- Reg10[3] = 1
- Reg28[6] = 0
- Reg28[4] = 0
- Reg2B[5:0] = 0x3F

进入接收状态后, 读取 Reg64 的值, 根据 Reg64 的值上下调节 Reg2B 的增益控制。Reg2B 对应增益表如下:

表 5-2: Reg2B 增益控制表

Reg2B[5:0]	Gain(dBm)
111111	54
111110	48
111101	42
111100	36
111000	30
110100	24
110000	18
100000	12
010000	6
000000	0

RSSI 计算公式:  $RSSI = (Reg64/2) - Gain + delta\_rssi$

### 5.3.4 RSSI 自动过滤功能

RSSI 自动过滤功能, 即设置 RSSI 门限值, 在解调器接收到前导信号时, 会判断 RSSI 值, 如接收到的信号强度低于设置的门限, 则解调器自动复位, 丢弃该信号重新进入接收状态。

如设置信号强度低于-96dBm 时, 丢弃该包数据, 则进入接收之前设置:

- Reg28[5] = 1
- Reg2C[7:0] = 0xC0

## 5.4 系统复位

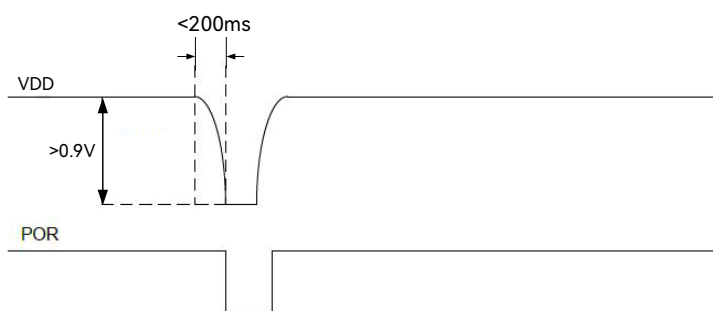
该芯片内部集成了上电复位 (POR)、外部复位和软件复位。上电默认状态下, GPIO2 默认为外部复位信号 (nRST) 输入, 只要 nRST 拉低就可以完成对芯片的复位。此外, 还可以通过 SPI 写命令的方式对芯片进行复位操作。芯片一旦复位, 内部所有配置都恢复到上电默认状态, MCU 需要对芯片重新进入初始化操作。



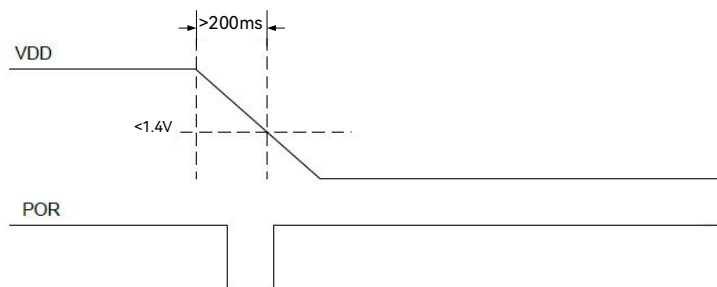
## 5.4.1 上电复位 POR

上电复位电路检测电源电压变化，当电压压降出现较大变化时，上电复位电路会产生复位信号来复位芯片系统，当 POR 复位信号出现过后，芯片内部所有参数恢复到原始状态，需要 MCU 重新对芯片进行初始化配置。芯片触发 POR 产生复位信号条件有两种：

1. 短暂电源突变导致 POR 产生复位信号，触发条件是当 VDD 在 200ms 以内骤降大于 0.9V 时，POR 会产生复位信号，如下图：



2. 缓慢的电源电压下降导致 POR 产生复位信号，触发条件是 VDD 在大于 200ms 下降到 1.4V 以下时，POR 会产生复位信号，如下图：



## 5.4.2 外部复位 nRST

芯片支持外部复位，上电默认状态 GPIO2 配置为复位（nRST）功能，通过拉低 nRST 可复位芯片。相关寄存器如下：

寄存器地址	位数	R/W	比特名	功能说明
0x4D	4	R/W	gpio_dir[2]	gpio[2] 方向，默认复位输入功能： 0: output 1: input
0x5D	7	R/W	nrst_sel	nRST 引脚功能设置： 0: nRST 1: GPIO2
	6	R/W	pull_up_en	nRST 引脚 Pull-UP 使能： 0: enable 1: disable

## 5.4.3 软件复位

软件复位是通过写寄存器复位芯片，需先使能复位命令，再写入复位命令才能有效复位芯片，相关寄存器配置如下：

寄存器地址	位数	R/W	比特名	功能说明
0x60	7:0	R/W	command	0x80: IDLE command 0x20: RX command 0x05: StandBy command 0x06: Reset command, 需要先使能 RegA3[7] 0x10: FS command 打开频综
0xA3	7	R/W	srst_cmd_enable	Reset Command 使能： 0: disable 1: enable

## 6 芯片运行

### 6.1 状态机控制图

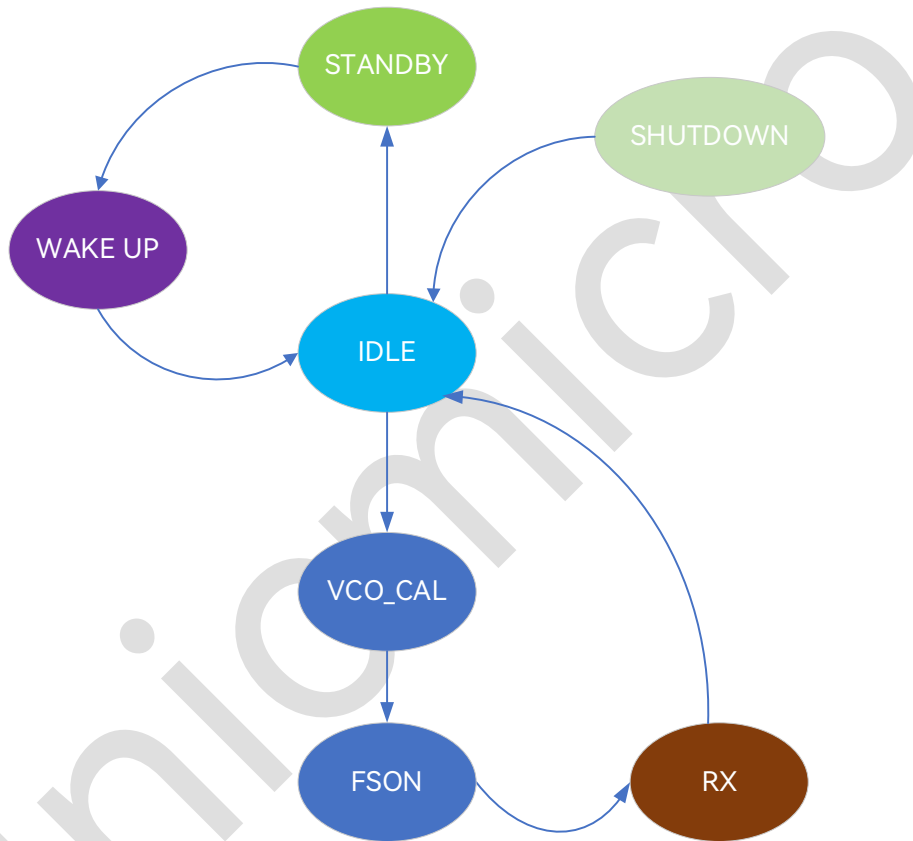


图 6-1: 状态机控制图

### 6.2 工作模式

表 6-1: 工作模式

State/mode	Description	Command
SHUTDOWN	芯片处于关断状态	SDN 拉高
IDLE	空闲状态，复位后进入此状态	上电复位或发送 IDLE 命令
STANDBY	数字模块工作，其他模块关闭，寄存器保持	SPI 写 STANDBY 命令
FSON	频综打开状态，可快速进入接收状态	SPI 写 FSON 命令
RX	接收数据状态	SPI 写 RX 命令

## 6.3 状态机说明

### 1. 关断状态 (SHUTDOWN)

当 SDN 管脚拉高之后, 芯片进入关闭状态, 芯片内部所有的电路模块都关闭, 消耗电流小于 10nA。

### 2. IDLE 状态

上电复位或发送 IDLE 命令后, 芯片进入 IDLE 状态。在此状态下, 晶体振荡器开启, 等待 SPI 接口命令再执行其它的动作。

### 3. STANDBY 状态

SPI 写命令进入 STANDBY 状态后, 芯片的数字供电开启, 所有数字电路正常供电, 其它模块都关闭。寄存器值可以保持。将 SPI\_CS 拉低后, 芯片从 STANDBY 状态退出, 进入 IDLE 状态。

### 4. 频综打开状态 (FSON)

频综打开后保持在这个状态, 在此状态下, 当收到 RX 命令会马上进入 RX 状态。

### 5. 接收状态 (RX)

收到 RX 命令后, 芯片先打开 PLL, 然后进行 VCO 校准并依次启动接收电路 (LNA、Mixer、PGA 和 ADC) 和数字解调器。当芯片收到数据包后, 会给出 nIRQ 中断指示信号然后退回到 IDLE 状态或者 STANDBY 状态 (定时唤醒模式下)。

## 6.4 GPIO 和中断

UM2002B 有 4 个 GPIO, 分别是 nIRQ 和 GPIO[2:0], 每个 GPIO 都可以配置成不同的输入或者输出。

芯片能产生两种中断信号, pkt\_flag 和 fifo\_flag, 该两个中断信号标志位都可读。pkt\_flag 分为 3 个功能: 前导匹配、同步字匹配和接收包完成。fifo\_flag 在接收模式时表示 FIFO full。

## 6.4.1 GPIO 的配置

nIRQ 和 GPIO[2:0]为双向口，GPIO[2]默认为输入口，作为 nRST 使用，拉低可复位整个数字部分。GPIO 配置相关寄存器如下表所示：

表 6-2: GPIO 配置相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x4D	7	R/W	nirq_dir	nIRQ IO 口方向: 0: output 1: input
	6	R/W	gpio_dir[0]	gpio[0] 方向: 0: output 1: input
	5	R/W	gpio_dir[1]	gpio[1] 方向: 0: output 1: input
	4	R/W	gpio_dir[2]	gpio[2] 方向, 默认复位输入功能: 0: output 1: input
	3:0	R/W	nirq_sel	nIRQ 引脚输出功能选择: 0x0: pkt_flag(包括数据包完成中断 pkt_int, 接收前导码中断 preamble_int, 接收同步字中断 syncword_int) 0x1: pkt_int 0x2: preamble_int 0x3: syncword_int 0x4: fifo_flag 0x5: brclk (参考 Reg11[2:0]描述) 0x6: rxdata 0x7: NC 0x8: NC 0x9: NC 0xa: rxdata 0xb: NC 0xc: NC 0xd: 高电平 Others: 低电平

寄存器地址	位数	R/W	比特名	功能说明
0x4E	7:4	R/W	gpio_0_sel	gpio[0]输出功能选择, 参考 Reg4D[3:0]描述
	3:0	R/W	gpio_1_sel	gpio[1]输出功能选择, 参考 Reg4D[3:0]描述
0x4F	7:4	R/W	gpio_2_sel	gpio[2]输出功能选择, 参考 Reg4D[3:0]描述
0x11	2:0	R/W	brclk_sel	GPIO brclk 输出时钟选择: 000: rx_clk 接收 bit rate 时钟 001: xtal_clk 晶振时钟 010: xtal_clk/8 011: xtal_clk/16 100: 频综时钟 101: NC 110: 内部 RC32K 时钟 111: ADC_clk

## 6.4.2 中断的配置和映射

GPIO 输出功能描述如下表所示:

表 6-3: GPIO 输出功能描述

Value	Name	功能说明
0x0	pkt_flag	pkt_flag (包括数据包完成中断 pkt_int, 接收前导码中断 preamble_int, 接收同步字中断 syncword_int)
0x1	pkt_int	包完成中断, 接收完成
0x2	preamble_int	接收 preamble 有效中断
0x3	syncword_int	接收 syncword 有效中断
0x4	fifo_flag	RX FIFO 快满中断
0x5	brclk	测试时钟输出, 由 Reg11[2:0]设置。 Reg11[2:0]: 000: rx_clk 接收 bit rate 时钟 001: xtal_clk 晶振时钟 010: xtal_clk/8 011: xtal_clk/16 100: 频综时钟 101: NC 110: 内部 RC32K 时钟 111: ADC_clk
0x6	rxdata	解调器输出的串行数据
0x7	NC	-

Value	Name	功能说明
0x8	NC	-
0x9	NC	-
0xa	rxdata	-
0xb	NC	-
0xc	NC	-
0xd	Hi_l	高电平
Others	Lo_l	低电平

## 6.5 FIFO 缓冲区

芯片内置 256 字节的 FIFO，可以通过寄存器 Reg0E[3]设置 FIFO 的使用大小。在读一帧接收 FIFO 数据时，应对接收 FIFO 读指针进行清零。接收写指针芯片内部会自动清零。

### 6.5.1 FIFO 相关的寄存器

FIFO 相关的寄存器如下表所示：

表 6-4: FIFO 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x50	7:0	R/W	rx_fifo_wr_ptr	RX FIFO 写指针，写 0x80 清除指针
0x51	7:0	R/W	rx_fifo_rd_ptr	RX FIFO 读指针，写 0x80 清除指针
0x52	7:0	R/W	rx_fifo_data	RX FIFO
0x0E	3	R/W	fifo_share_en	FIFO 大小设置： 0: RX FIFO 128 字节 1: RX FIFO 256 字节
0xA0	5:0	R/W	fifo_full_thr	RX FIFO 满门限，RX FIFO 数据高于门限值时 fifo_flag 置‘1

### 6.5.2 FIFO 的工作模式

芯片的 RX FIFO 默认使用 128-byte。用户也可以将 fifo\_share\_en 设为 1，那么 RX FIFO 使用一个 256-byte 的 FIFO。

### 6.5.3 FIFO 的中断时序

对于 RX, 设置 `fifo_full_thr` 为将满阈值, 只要接收的剩余空间低于此门限值, `fifo_flag` 产生中断。

### 6.5.4 FIFO 的应用场景

#### 1. 在 RX 下接收数据

初始化寄存器优化值、数据率以及数据包的配置, 并写 `Reg60 = 0x20` 进入接收, 等 MCU 收到 `nIRQ` 中断后:

- A. 读 `Reg61` 寄存器。
- B. 判断 `Reg61[7:4]` 是否为 `0xD`, 如果正确表明收到一包数据。
- C. 清 RX FIFO 读指针, `Reg51 = 0x80`。
- D. 读 RX FIFO 数据 `Read Reg(0x52)`。
- E. 读 RSSI (`Reg0x64`): 根据需要可读出该包的 RSSI 值。
- F. 写接收命令重新进入接收数据状态。

#### 2. 进入 RX 后, 边读 FIFO 数据边接收

该场景适用于每次接收超出 FIFO 长度的数据包, 在启动接收后, 需要监控 `fifo_flag` 信号, 该信号可以配置到引脚输出, 也可以通过读 `Reg61` 寄存器获取; 在 RX FIFO 快满时要及时读出数据, 保证接收到的数据不被覆盖。



## 7 数据处理机制

芯片提供灵活可配置的数据控制模式，主要分为如下两大类：

- Direct 直通模式：在 RX 模式下，接收的串行数据可直接从 GPIO 口输出。
- Packet 数据包模式：在数据包模式下所有数据都要经过 FIFO，且支持各种数据包模式的控制，数据包模式又可以分成四种帧格式。

### 7.1 直通模式 (Direct)

数据直通 (direct) 指外部 MCU 通过 GPIO 获取接收数据，在 RX 启动前使能 Direct Mode，且将包模式配置为 Mode 0，让芯片工作在循环状态，此时 RX 的启动或停止完全由 MCU 的命令控制。

#### 7.1.1 Direct 相关寄存器

表 7-1: Direct 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0E	2	R/W	direct_mode	直通模式使能： 0: disable 1: enable

### 7.2 数据包 (Packet)

数据包 (Packet) 中的 Payload 数据都是从 FIFO 读写，芯片内部实现了 256 字节的 FIFO。在数据包模式下有四种控制方式可选。在四种模式下，Preamble 和 Syncword 的配置都起作用，Preamble 最大可配置 256 个字节的长度，Syncword 可配置 0~8 个字节的长度。

数据包模式还可以对数据包进行 Manchester、FEC、Whiten 编解码以及 CRC 校验。CRC 可配置 8、16、24、32 位四种长度校验，且 CRC 多项式任意可配置。Syncword 和 Payload 可配置高位优先或低位优先。CRC 和 Syncword 数据的 Manchester 编码可单独开关。

## 7.2.1 Mode 相关寄存器

表 7-2: Mode 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0E	1:0	R/W	packet_mode	数据包控制模式: 00: 模式 0, 接收到的数据循环写入 RX FIFO 中, 需要 MCU 写命令退出接收状态 01: 模式 1, 由寄存器控制数据包长度 10: 模式 2, RX FIFO 中的第 1 个或 2 个字节作为包长度 11: 模式 3, 含有 length, address, seqnum 等多种数据域的模式

## 7.2.2 MODE 0

MODE 0 为 FIFO 循环模式。在此模式下启动 RX, 如果没有收到 MCU 的停止命令, RX 端会将接收到的数据不停地写入到 RX FIFO 中, 并且 RX 端会检测 Preamble 和 Syncword。帧格式有 Preamble、Syncword 和 Data 三个数据域, 长度不受芯片本身控制。

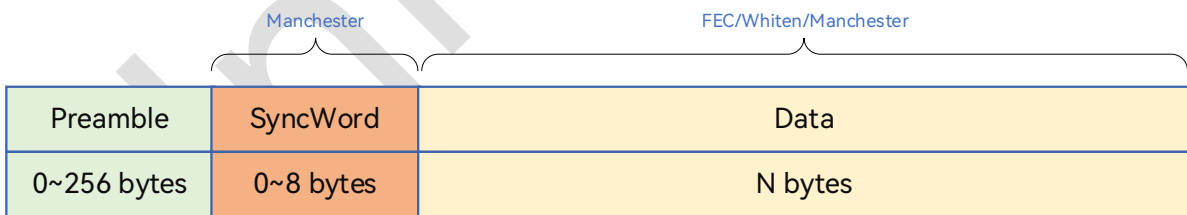


图 7-1: Mode 0 帧格式

## 7.2.3 MODE 1

MODE 1 帧格式包含 Preamble、Syncword、Data 和 CRC 域。RX 端接收 Data 的长度同样由 Payload\_len[15:0]寄存器控制。

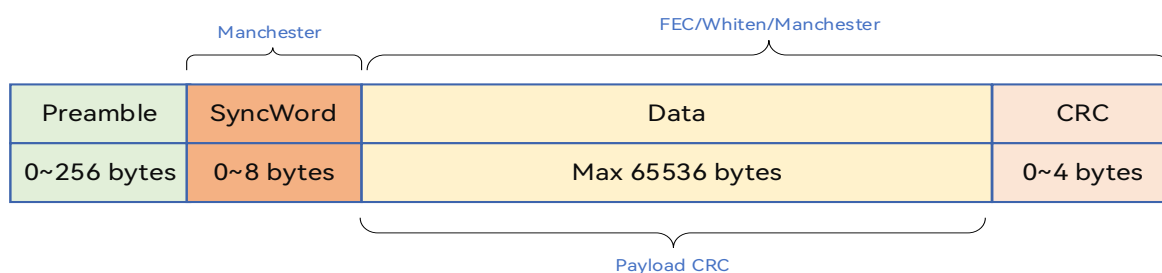


图 7-2: Mode 1 帧格式

## 7.2.4 MODE 2

MODE 2 帧格式包含 Preamble、Syncword、Length、Data 和 CRC 域。在 RX 端，Length 为接收到的数据的前 1 个字或前 2 个字节，Length 为两个字节时高低字节顺序可互换，根据接收到的 Length 域数据去控制 Data 域的长度。

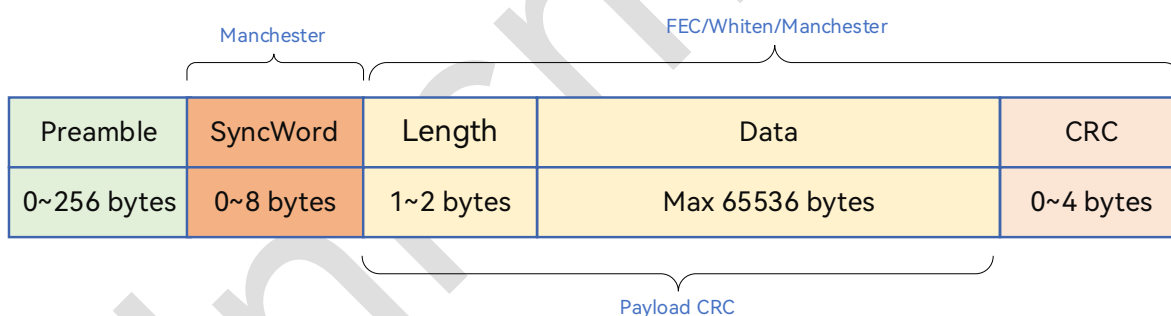


图 7-3: Mode 2 帧格式

## 7.2.5 MODE 3

MODE 3 帧格式包含 Preamble、Syncword、Length、Address、SeqNum、FCS2、Data 和 CRC 域。Length、Address、SeqNum、FCS2 作为 payload 域并且可以单独使能。Address 域可选择放置在 Length 之前。Length 数据来自 Payload\_len[15:0]寄存器，且 Payload 长度范围包括 Length 到 CRC 之前的所有数据域。SeqNum 为数据包计数器，可由寄存器设置也可设置为自动增加。如果 Length 域没有使能，则接收端根据寄存器 Payload\_len 来接收 Payload 数据。

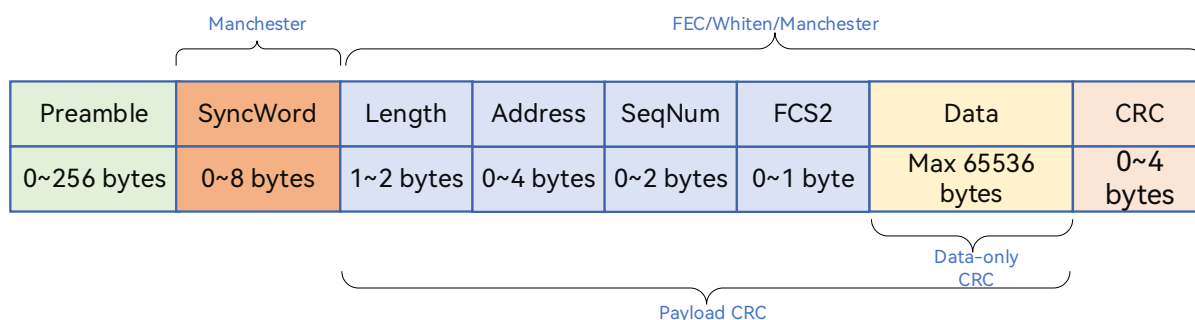


图 7-4: Mode 3 帧格式

## 7.3 Preamble 配置

对于 RX 来说, Preamble 检测成功会有 PREAM\_OK 中断产生。用户可在有需要的时候去检测此中断。

### 7.3.1 Preamble 配置相关寄存器

表 7-3: Preamble 配置相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0D	4	R/W	preamble_en	设置 preamble 前导码使能: 0: disable 1: enable 使能
0x10	2	R/W	preamble_int_en	接收 preamble (前导码) 匹配中断使能: 0: disable 1: enable
0x37	7:0	R/W	csrst_len	在补偿功能使能时, 找到信号的频谱后连续收到 Reg37*4 个前导后, 则认为找到有效的 preamble
0x38	4	R/W	demod_comp_en	解调频偏补偿: 0: disable 1: enable

## 7.4 Sync Word 配置

Sync Word 支持独立使能，支持曼彻斯特编码，支持 Bit 顺序互换。在接收时支持同步字中断。

### 7.4.1 Sync Word 相关的寄存器

表 7-4: Sync 配置相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0D	5	R/W	syncword_en	Syncword 同步字使能: 0: disable 1: enable
0x10	1	R/W	syncword_int_en	接收 Syncword (同步字) 匹配中断使能: 0: disable 1: enable
0x74	4	R/W	sync_man_en	同步字曼彻斯特编码使能: 0: disable 1: enable
	3	R/W	sync_bit_order	同步字 bit 顺序: 0: 字节低位 LSB 1: 字节高位 MSB
	2:0	R/W	sync_len	同步字长度: 000: {sync_id_1} 001: {sync_id_1, sync_id_2} 010: {sync_id_1, sync_id_2, sync_id_3} 011: {sync_id_1, sync_id_2, sync_id_3, sync_id_4} 100: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5} 101: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6} 110: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6, sync_id_7} 111: {sync_id_1, sync_id_2, sync_id_3, sync_id_4, sync_id_5, sync_id_6, sync_id_7, sync_id_8}

寄存器地址	位数	R/W	比特名	功能说明
0x75	7:0	R/W	sync_id_1	Sync Word 的值，根据不同的 sync_len 设置来填入不同的寄存器。
0x76	7:0	R/W	sync_id_2	
0x77	7:0	R/W	sync_id_3	
0x78	7:0	R/W	sync_id_4	
0x79	7:0	R/W	sync_id_5	
0x7A	7:0	R/W	sync_id_6	
0x7B	7:0	R/W	sync_id_7	
0x7C	7:0	R/W	sync_id_8	

## 7.5 Length 配置

在不同的包模式下，Length 域配置方式不一样。并且支持长度字节选择，高低位互换。

### 7.5.1 Length 相关的寄存器

表 7-5: Length 相关的寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0E	7	R/W	length_byte_swap	包长度控制域为两个字节时，高低字节顺序交换： 0: 低字节在前 1: 高字节在前
	6	R/W	length_sel	选择 1 个或 2 个字节作为包长度控制域： 0: 1 字节 1: 2 字节
0x6A	7:0	R	m3_rx_payload_len[7:0]	数据包模式 3 时，接收 length 数据
0x6B	7:0	R	m3_rx_payload_len[15:8]	数据包模式 3 时，接收 length 数据
0x80	5	R/W	m3_length_en	数据包模式 3 时，数据长度域控制： 0: disable 1: enable
0x7D	7:0	R/W	payload_len[7:0]	payload 长度，在数据包模式 1 和数据包模式 3 有效
0x7E	7:0	R/W	payload_len[15:8]	payload 长度，在数据包模式 1 和数据包模式 3 有效

寄存器地址	位数	R/W	比特名	功能说明
0x81	1	R/W	m3_addr_pos_sel	数据包模式 3 时，地址位置选择： 0: address 放在 length 之后 1: address 放在 length 之前

## 7.5.2 MODE 0 的 length 配置

在 MODE 0 中，RX 端会将接收到的数据不停地写入到 RX FIFO 中。所以在 MODE 0 中 payload length 是无限制的。

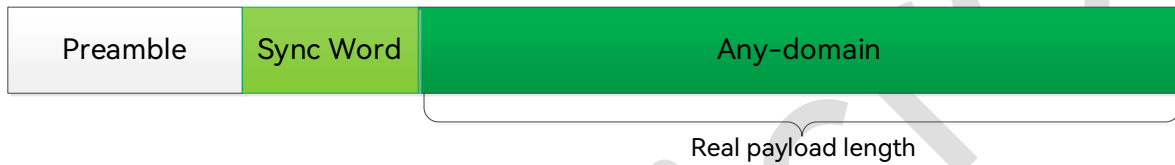


图 7-5: Mode 0

## 7.5.3 MODE 1 的 length 配置

在 MODE 1 中，payload\_len 来自寄存器{Reg7D,Reg7E}，Real payload length 的实际长度为配置的寄存器 Payload\_len 加一，例如 Real payload length 为 8 bytes，则 Payload\_len 配置为 7。

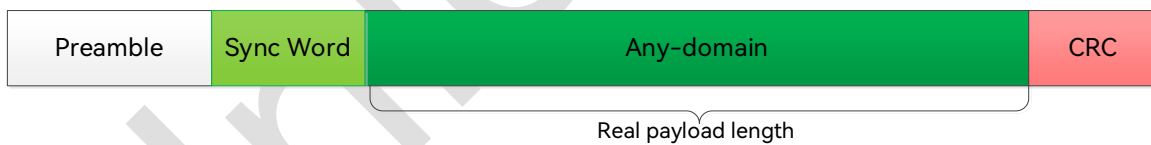


图 7-6: Mode 1

## 7.5.4 MODE 2 的 length 配置

在 MODE 2 中，payload length 由 FIFO 第一个或前两个字节配置，length 表示在 length 后面跟随的数据字节数量。

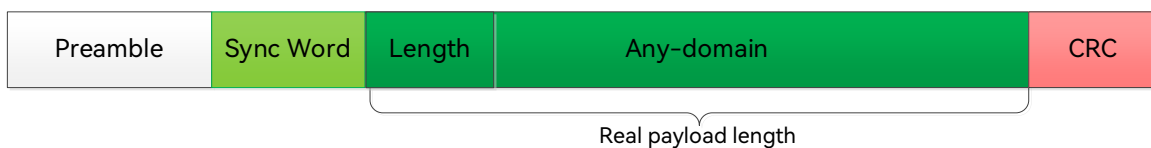


图 7-7: Mode 2

## 7.5.5 MODE 3 的 length 配置

在 MODE 3 中，根据 Length 使能或者不使能来分为两种格式：固定包格式和可变包格式，对于 RX 来说，如果 length 域不使能，则来自寄存器，如果 length 域使能，则来自数据包格式的 length。

### 7.5.5.1 固定包格式

在 MODE 3 中，Length 域不使能(Reg80[5]=0, m3\_length\_en=0)的情况下，对于 RX 来说，Real payload length 来自寄存器{Reg7D,Reg7E}，Real payload length 的实际长度为配置的寄存器 Payload\_len 加一。

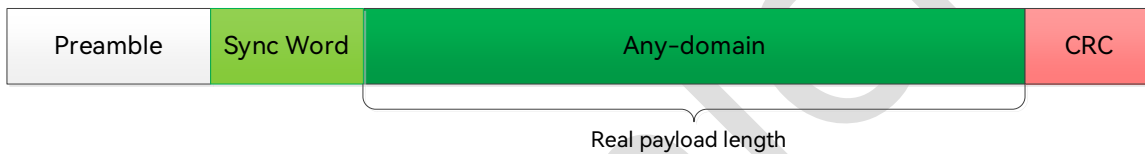


图 7-8: m3\_length\_en=0 时 MODE 3 帧格式

当 length 域不使能 (m3\_length\_en=0) 时，对于 RX 来说，Payload 结构 = Address + Seqnum + FCS2 + Data，其中 Address / Seqnum / FCS2 单独配置使能，示例如下：

● **举例1:**

Payload\_len<15:0> = 17

Addr\_en = 1

Addr\_size <1:0> = 2

Seqnum\_en = 1

Seqnum\_size = 1

Fcs2\_en = 1

此处，Payload 包含 Address + Seqnum + FCS2 + Data 共 4 部分，Payload 的总长度是 17 + 1 = 18，Address 的长度是 2 (Addr) + 1 = 3，Sequence Num 的长度是 1 + 1 = 2，FCS2 长度固定为 1，所以，Data 的长度是 18 - 3 - 2 - 1 = 12 个 byte。



- 举例2:

Payload\_len <15:0> = 17

Addr\_en = 1

Addr\_size<1:0> = 2

Seqnum\_EN = 0

Seqnum\_size = 1

Fcs2\_en = 0

此处, Address + Seqnum + FCS2 不存在, 那么 Data 的长度就是 18。

### 7.5.5.2 可变包格式

在 MODE 3 中, Length 域使能(Reg80[5]=1, m3\_length\_en=1)的情况下, 对于 RX 来说, payload length 来自数据包的 Length 域, 根据 Length 域的位置不同及 Address 域使能来决定 payload length 的计算。

- **Address 不使能 (m3\_addr\_en=0)**

当 Address 不使能的情况下, Payload 结构 = Length + Seqnum + FCS2 + Data, 其中 Seqnum、FCS2 可以独立配置使能, payload Length 包括 length+Seqnum\_FCS2+Data。

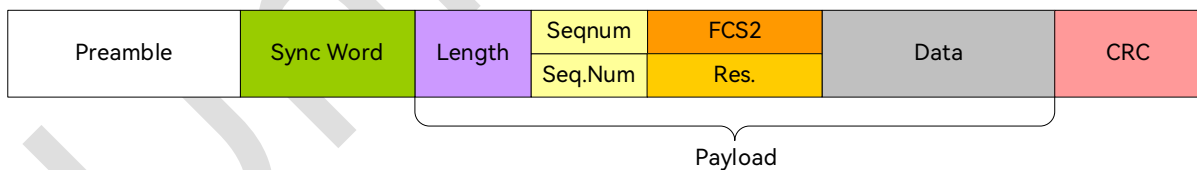


图 7-9: 可变长度包格式, Address 不存在

当 Length 域使能, Address 不使能 (m3\_length\_en=1, m3\_addr\_en=0), 整个 payload 的长度, 包括 length+Seqnum\_FCS2+Data, Length 的内容由 Payload\_len <15:0> 寄存器配置。如果 Length\_size = 0, 即 Payload\_len <7:0> 有效, 最大值为 255, Length Byte 本身是 1 个 byte 的长度。如果 Length\_size = 1, 即 Payload\_len <15:0> 有效, 最大值为 65535, Length Byte 本身是 2 个 byte 的长度, 对于 RX 来说, RX 需要解析 length 域, length+1 表示 Payload 长度, 示例如下:

举例:

Payload\_len <15:0> = 17

Length\_size = 0

Seqnum\_EN = 1

Seqnum\_size = 1

Fcs2\_en = 1

此处, Payload 包含 Length + Seqnum + FCS2 + Data 共 4 部分, Payload 的总长度是  $17 + 1 = 18$ , Length 的长度是 1, Sequence Num 的长度是  $1 + 1 = 2$ , FCS2 长度固定为 1, 所以, Data 的长度是  $18 - 1 - 2 - 1 = 14$  个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是  $18 - 1 = 17$  个 byte。

如果 length\_size = 1, 即 Length 的长度是 2, 所以, Data 的长度是  $18 - 2 - 2 - 1 = 13$  个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是  $18 - 2 = 16$  个 byte。

- **Address 存在, 且 Addr\_pos\_sel = 1 (Address 在 Length Byte 之前)**

ADDR 放在 LENGTH 之前, ADDR 单独罗列出来, Payload 结构 = Address + Length Byte + Seqnum + FCS2 + Data, Real payload length 的实际长度为配置的寄存器 Payload\_len 加一。

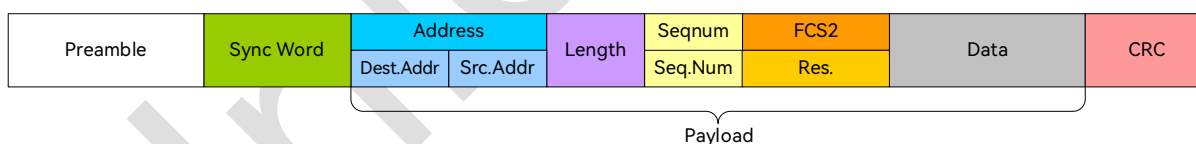


图 7-10: Address 存在, Address 在 Length Byte 之前

当 Address 存在, 且 Addr\_pos\_sel = 1 (Address 在 Length Byte 之前) 时, Payload 长度 =  $(addr\_size <1:0> + 1) + 1 + payload\_length <15:0>$ , 其中 Address 的长度根据  $addr\_size <1:0>$  确定,  $payload\_length <15:0>$  的内容就等于 Length Byte 的内容,  $payload\_length + 1$  指的是除 Address 外的 Payload 的长度。如果 length\_size = 0, 即  $payload\_length <7:0>$  有效, 最大值为 255, Length Byte 本身是 1 个 byte 的长度。如果 length\_size = 1, 即  $payload\_length <15:0>$  有效, 最大值为 65535, Length Byte 本身是 2 个 byte 的长度。对于 RX 来说, RX 需要解析 length 域,  $length + 1$  表示除 Address 外的 Payload 的长度, 示例如下:

举例:

Payload\_length<15:0> = 17

Addr\_en= 1

Addr\_size<1:0> = 2

Length\_size = 0

Seqnum\_EN = 1

Seqnum\_size = 1

FCS2\_EN = 1

此处, Payload 包含 Address + (Length + FCS1 + FCS2 + Data)共 5 部分, 其中, Address 的长度是 2 (Dest.Addr) + 1 = 3, Length 的长度是 1, Sequence Num 的长度是 1 + 1 = 2, FCS2 长度固定为 1, 那么 Payload 的总长度是 3 + (17 + 1) = 21, 所以, Data 的长度是 18 - 1 - 2 - 1 = 14 个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是 18 - 1 = 17 个 byte。

如果 length\_size = 1, 即 Length 的长度是 2, 所以, Data 的长度是 18 - 2 - 2 - 1 = 13 个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是 18 - 2 = 16 个 byte。

- **Address 存在, 且 Addr\_pos\_sel = 0 (Address 在 Length Byte 之后)**

ADDR 放在 LENGTH 之后, 同 MODE 1, Payload 结构 = Length Byte + Address + Seqnum + FCS2 + Data, Real payload length 的实际长度为配置的寄存器 Payload\_len 加一。

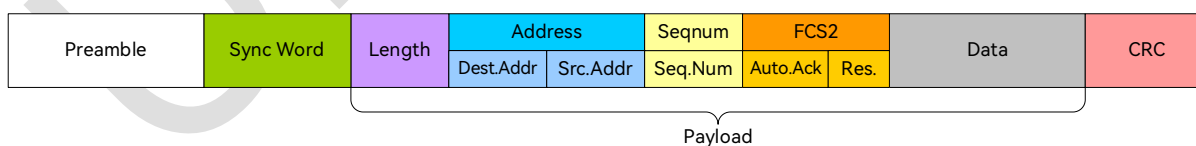


图 7-11: Address 存在, Address 在 Length Byte 之后

当 Address 存在, 且 Addr\_pos\_sel = 0 (Address 在 Length Byte 之后) 时, Payload 长度: 1 + payload\_length<15:0>, 其中 payload\_length<15:0>的内容就等于 Length Byte 的内容, 指的是 Payload 的长度。如果 length\_size = 0, 即 payload\_length<7:0>有效, 最大值为 255, Length Byte 本身是 1 个 byte 的长度。如果 length\_size = 1, 即 payload\_length<15:0>有效, 最大值为

65535, Length Byte 本身是 2 个 byte 的长度。对于 RX 来说, RX 需要解析 length 域, length+1 表示 Payload 的长度, 示例如下:

举例:

Payload\_length<15:0> = 17

Addr\_en = 1

Addr\_size<1:0> = 2

Length\_size = 0

Seq\_en = 1

Seqnum\_size = 1

FCS2\_EN = 1

此处, Payload 包含 Length + Address + Seqnum + FCS2 + Data 共 5 部分, Payload 的总长度是  $17 + 1 = 18$ , Length 的长度是 1, Address 的长度是  $2 (\text{Addr}) + 1 = 3$ , Sequence Num 的长度是  $1 + 1 = 2$ , FCS2 长度固定为 1, 所以, Data 的长度是  $18 - 1 - 3 - 2 - 1 = 11$  个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是  $18 - 1 - 3 = 14$  个 byte。

如果 length\_size = 1, 即 Length 的长度是 2, 所以, Data 的长度是  $18 - 2 - 3 - 2 - 1 = 10$  个 byte。

如果 Seqnum + FCS2 不存在, 那么 Data 的长度就是  $18 - 2 - 3 = 13$  个 byte。

## 7.6 Address 配置 (仅 Mode 3)

### 7.6.1 Address 的相关寄存器

表 7-6: Address 配置相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x81	5:4	R/W	m3_addr_size	数据包模式 3 时, 地址长度: 00: 1 字节 01: 2 字节 10: 3 字节 11: 4 字节
	0	R/W	m3_addr_en	数据包模式 3 时, 地址域控制: 0: disable 1: enable
0x82	7:0	R/W	m3_addr_val[7:0]	数据包模式 3 时, 地址值
0x83	7:0	R/W	m3_addr_val[15:8]	
0x84	7:0	R/W	m3_addr_val[23:16]	
0x85	7:0	R/W	m3_addr_val[31:24]	
0x86	7:0	R/W	m3_addr_bit_mask[7:0]	数据包模式 3 时, 地址过滤, mask 为 '1' 时对应地址位不作比较
0x87	7:0	R/W	m3_addr_bit_mask[15:8]	
0x88	7:0	R/W	m3_addr_bit_mask[23:16]	
0x89	7:0	R/W	m3_addr_bit_mask[31:24]	

表 7-7: M3\_ADDR\_SIZE 选择

ADDR_SIZE	ADDR_VALUE			
	<31:24>	<23:16>	<15:8>	<7:0>
0	√			
1	√	√		
2	√	√	√	
3	√	√	√	√

表中打勾的位置表示要填入的寄存器。

## 7.7 seqnum 配置 (仅 Mode 3)

在 MODE 3 中，在发射 Sequence Num 作为包结构内容一并发出，在接收中 Sequence Num 自动填入到映射的寄存器中，可以直接读取寄存器值。

### 7.7.1 seqnum 相关的寄存器

表 7-8: seqnum 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x8A	4	R/W	m3_seqnum_en	数据包模式 3 时，seqnum 数据使能： 0: disable 1: enable
	3	R/W	m3_seqnum_bit_size	数据包模式 3 时，seqnum 数据长度： 0: 8 bits 1: 16 bits

## 7.8 Payload Data 配置

Payload Data 支持 NRZ、曼彻斯特、数据白化编解码以及高低位互换。

### 7.8.1 Payload Data 相关的寄存器

表 7-9: Payload Data 相关寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x74	4	R/W	sync_man_en	同步字曼彻斯特编码使能： 0: disable 1: enable
0x0D	7	R/W	payload_bit_order	Payload bit 顺序： 0: LSB 低位在前 1: MSB 高位在前

寄存器地址	位数	R/W	比特名	功能说明
	3:2	R/W	pkt_enc_type	数据包编码: 00: NRZ 01: 曼彻斯特编码 10: 无效 11: 交织编码
0x94	2	R/W	crc_man_en	CRC 曼彻斯特编码使能: 0: disable 1: enable
0x0D	6	R/W	Manchester_inv	曼彻斯特编码: 0: 上升沿为编码 1, 下降沿为编码 0 1: 下降沿为编码 1, 上升沿为编码 0
0x0E	4	R/W	scramble_en	数据白化使能: 0: disable 1: enable
0x71	6	R/W	scramble_len	伪随机数长度选择: 0: PN9 1: PN7
	5	R/W	scramble_msb	选择首先输出的移位寄存器: 0: LSB 1: MSB
	4	R/W	scramble_type	伪随机数产生器类型: 0: 抽出模式 1: 插入模式
	3:1	R/W	scramble_poly	反馈抽头选择
	0	R/W	scramble_init_val[8]	scramble 初始化值最高位
0x70	7:0	R/W	scramble_init_val[7:0]	scramble 初始化值

## 7.8.2 Payload\_bit\_order 的示例

- Payload\_bit\_order = 0 表示发送时把 Payload 的每个 byte 自身从 LSB 到 MSB 的顺序发送或进行 Manchester/Whiten 编码。
- Payload\_bit\_order = 1 则表示把 Payload 的每个 byte 自身从 MSB 到 LSB 的顺序发送或进行 Manchester/Whiten 编码。

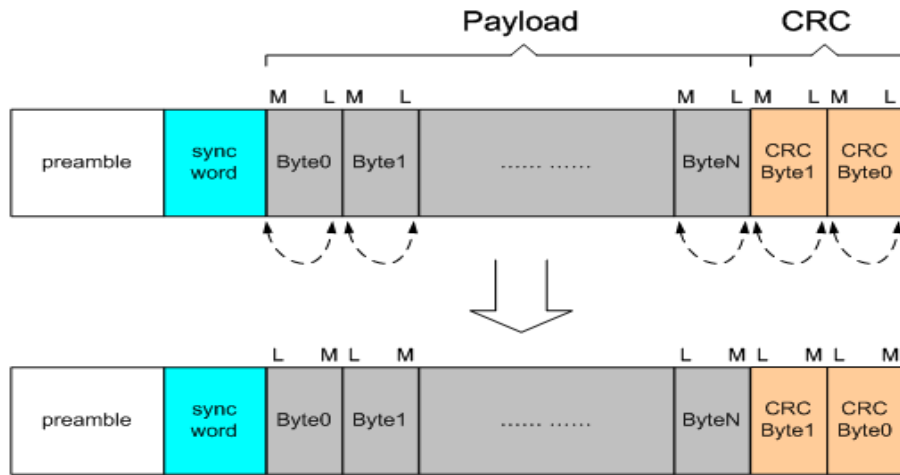
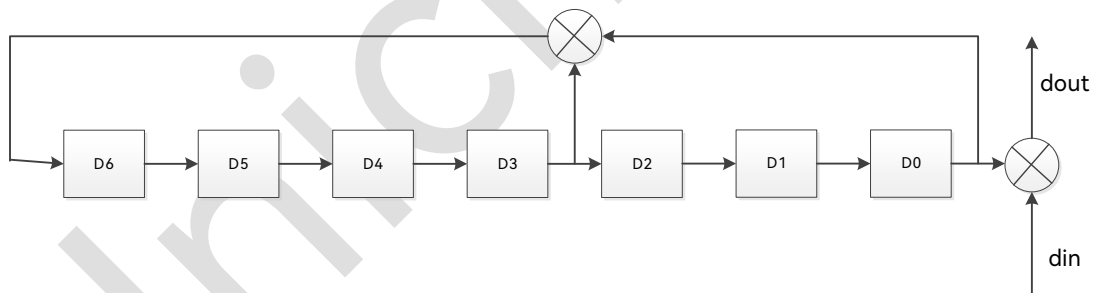


图 7-12: PAYLOAD\_BIT\_ORDER 操作

### 7.8.3 数据白化示例

数据白化随机数产生器有 PN7 或 PN9 可选择，结构上也有抽出或插入模式；数据输出可选最低位或最高位。

- PN7 的抽出模式

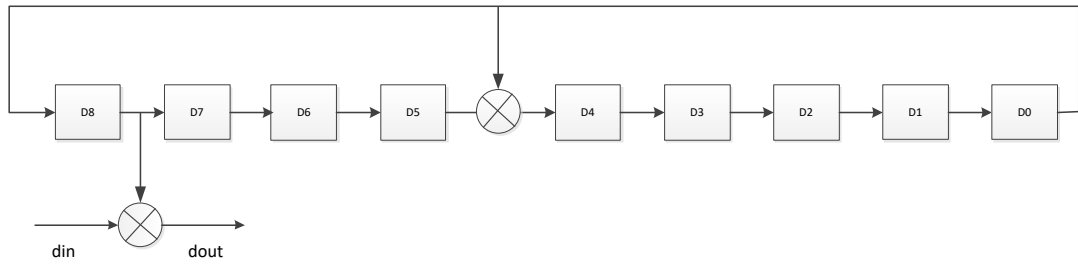


如上图，移位顺序不变，从高位移位到低位，对应配置为：

- Reg0E[4] = 1 打开白化使能。
- Reg71[6] = 1 选择 PN7。
- Reg71[5] = 0 选择 LSB 和数据做异或。
- Reg71[4] = 0 选择抽出模式。
- Reg71[3:1] = 'b010 选择在 D2 输入处抽出，即 Reg71[3:1]+1 的寄存器输出处，例如举例为 2，则在 D3 输出处抽出或者插入。



- PN9 的插入模式



如上图，移位顺序不变，从高位移位到低位，对应配置为：

- Reg0E[4] = 1 打开白化使能。
- Reg71[6] = 0 选择 PN9。
- Reg71[5] = 1 选择 MSB 和数据做异或。
- Reg71[4] = 1 选择插入模式。
- Reg71[3:1] = 'b100 选择在 D4 输入处插入，即 Reg71[3:1]+1 的寄存器输出处，例如举例为 4，则在 D5 输出处抽出或者插入。

## 7.9 CRC 配置

### 7.9.1 CRC 相关的寄存器

表 7-10: CRC 相关的寄存器列表

寄存器地址	位数	R/W	比特名	功能说明
0x0E	5	R/W	crc_en	CRC 使能: 0: disable 1: enable
0x80	4	R/W	m3_crc_sel	数据包模式 3 时, CRC 的校验范围: 0: All payload 1: 仅 FIFO 中的数据
0x94	4	R/W	crc_bit_order	CRC bit 顺序 0: LSB 1: MSB

寄存器地址	位数	R/W	比特名	功能说明
	3	R/W	crc_inv	CRC bit 取反: 0: disable 1: enable
	1:0	R/W	crc_len	CRC 长度: 00: 8 位 01: 16 位 10: 24 位 11: 32 位
0x95	7:0	R/W	crc_poly[7:0]	CRC 多项式配置
0x96	7:0	R/W	crc_poly[15:8]	
0x97	7:0	R/W	crc_poly[23:16]	
0x98	7:0	R/W	crc_poly[31:24]	
0x99	7:0	R/W	crc_init_val[7:0]	CRC 移位寄存器初始化值
0x9A	7:0	R/W	crc_init_val[15:8]	
0x9B	7:0	R/W	crc_init_val[23:16]	
0x9C	7:0	R/W	crc_init_val[31:24]	

## 7.9.2 CRC 配置说明

- **Mode3\_crc\_sel (Reg80[4])**

M3\_crc\_sel 用来指定在 Mode 3 时 CRC 的编解码校验对象。

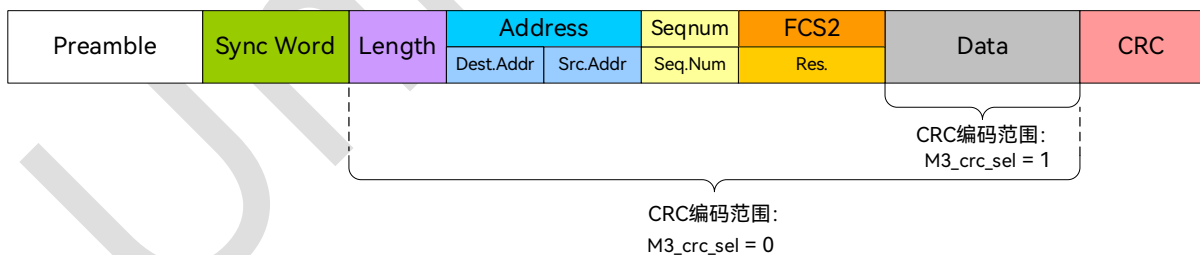


图 7-13: CRC 编码范围

- **crc\_inv (Reg94[3])**

此功能就是将 CRC 的每一位都进行逻辑取反，原来是 0 变为 1，原来是 1 变为 0。

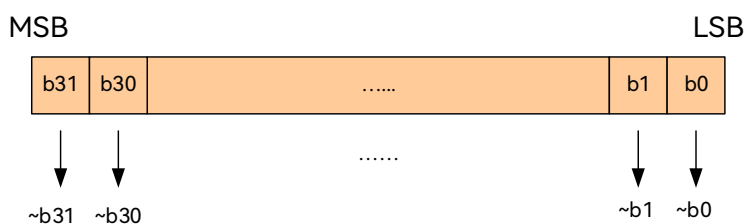


图 7-14: CRC\_INV

- **crc\_bit\_order (Reg94[4])**

此功能就是将 CRC 的 b31-b0 开始送出或者从 b0- b31 发送选项。



图 7-15: CRC\_BIT\_ORDER

### 7.9.3 常用 CRC 标准和多项式

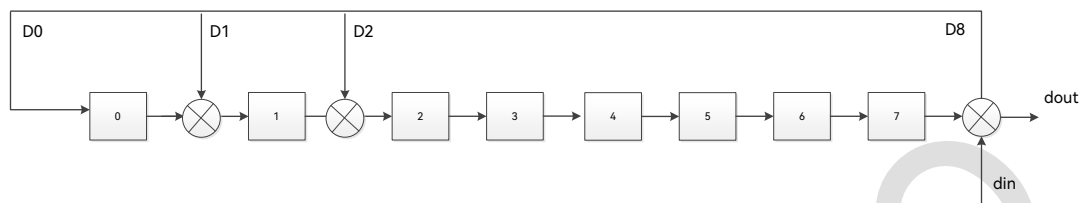
表 7-11: 常用 CRC 标准和多项式列表

名称	生成多项式	数值式	简记式	标准引用
CRC-4	$x^4+x+1$	0x1'3	0x3	ITU G.704
CRC-8	$x^8+x^5+x^4+1$	0x1'31	0x31	
CRC-8	$x^8+x^2+x+1$	0x1'07	0x07	
CRC-8	$x^8+x^6+x^4+x^3+x^2+x+1$	0x1'5E	0x5E	
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$	0x1'80F	0x80F	
CRC-32c	$X^{32}+x^{28}+x^{27}+\dots+x^8+x^6+1$	0x1'1EDC6F41	0x1EDC6F41	SCTP
CRC-16	$x^{16}+x^{15}+x^2+1$	0x1'8005	8005	IBM SDLC
CRC-CCITT	$x^{16}+x^{12}+x^5+1$	0x1'1021	0x1021	ISO HDLC, ITU X.25, V.34/V.41/V.42, PPP-FCS
CRC-32	$X^{32}+x^{26}+x^{23}+\dots+x^2+x+1$	0x1'04C11DB7	0x04C11DB7	ZIP, RAR, IEEE 802 LAN /FDDI, IEEE1394, PPP-FCS

## 7.9.4 配置 CRC8 和 CRC16 示例

UM2002B 配置 CRC 长度、多项式和初始值寄存器如下：

### 1. CRC8 的 $X^8+X^2+X^1+1$ 配置举例（简记式为 0x07）



如上图所示，对应配置为：

Reg94[1:0] = 0x00; crc\_len 设置

Reg95 = 0x07

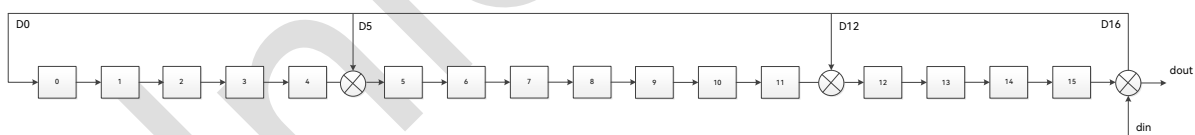
Reg96 = 0x00

Reg97 = 0x00

Reg98 = 0x00

Reg99, Reg9A, Reg9B, Reg9C 根据实际情况配置 CRC 的初始值。

### 2. CRC 的 $X^{16}+X^{12}+X^5+1$ 的配置举例（简记式为 0x1021）



如上图所示，对应配置为：

Reg94[1:0] = 0x01; crc\_len 设置 16bit

Reg95 = 0x21

Reg96 = 0x10

Reg97 = 0x00

Reg98 = 0x00

Reg99, Reg9A, Reg9B, Reg9C 根据实际情况配置 CRC 的初始值

## 8 接口说明

### 8.1 SPI 接口

芯片内置 SPI 从机模块，通过标准 4 线 SPI 接口与主机 MCU 进行通讯。SPI\_CS、SPI\_CLK、SPI\_MOSI 和 SPI\_MISO。SPI 接口可在最大 16MHz 下工作运行。SPI 接口模式的时钟极性为正，在时钟下降沿采样数据，在时钟上升沿输出数据，地址和数据部分都是从 MSB 开始传送。芯片内部访问都是以 SPI 读写寄存器的方式，第一个字节为地址，后面跟一个字节的数据。如果在访问 FIFO 对应的地址时，可以在一个 SPI\_CS 为低的周期内按字节方式连续的访问，SPI 接口控制器会自动增加访问地址，在访问 FIFO 数据时，地址和数据之间至少要等 3 个芯片的系统时钟，以便芯片确定 FIFO 指针地址。

在没有晶振时钟时，SPI 接口不能写数据，但仍然可以读寄存器数据。当访问寄存器的时候，SPI\_CS 要拉低。然后首先发送一个 R/W 位，之后是 7 位的寄存器地址。RW=0 表示写，RW=1 表示读。

SPI 默认是 4 线的，在上电后可配置成 3 线。在 SPI 3 线模式下，SPI\_MOSI 同时用于数据输入和输出，在读寄存器数据时，接口会在地址和数据之间对 SPI\_MOSI 的方向进行切换。

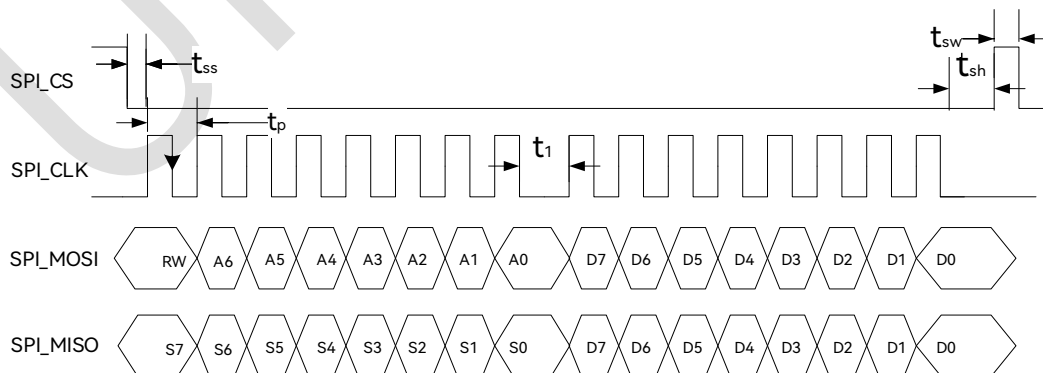


图 8-1: SPI 读写寄存器时序图

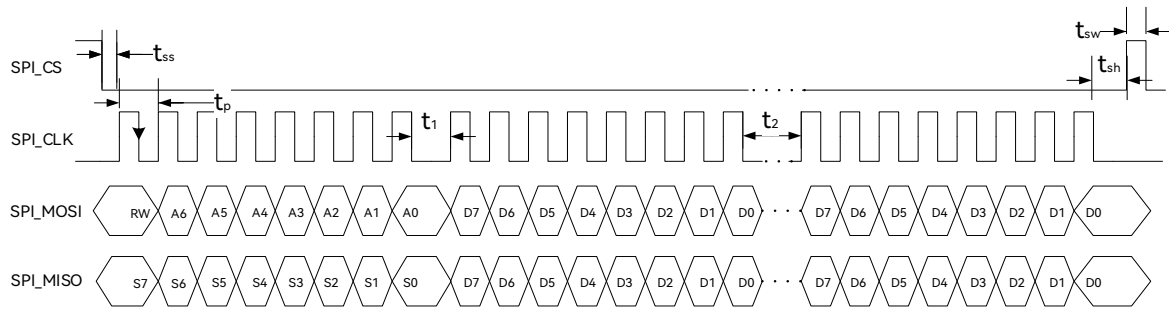


图 8-2: SPI 读写 FIFO 时序图

表 8-1: SPI 时序参数

Name	Min	Description
$t_{sw}$	100ns	两次 SPI 访问的间隔时间
$t_{ss}$	31.25ns	SPI_CS 和 SPI_CLK 的间隔时间
$t_{sh}$	31.25ns	SPI_CS 和 SPI_CLK 的间隔时间
$t_1$	32ns	地址和数据间隔时间
$t_2$	32ns	两个寄存器数据的时间间隔
$t_p$	62.5ns	SPI_CLK 时钟周期

# 9 应用参考

## 9.1 应用参考电路图

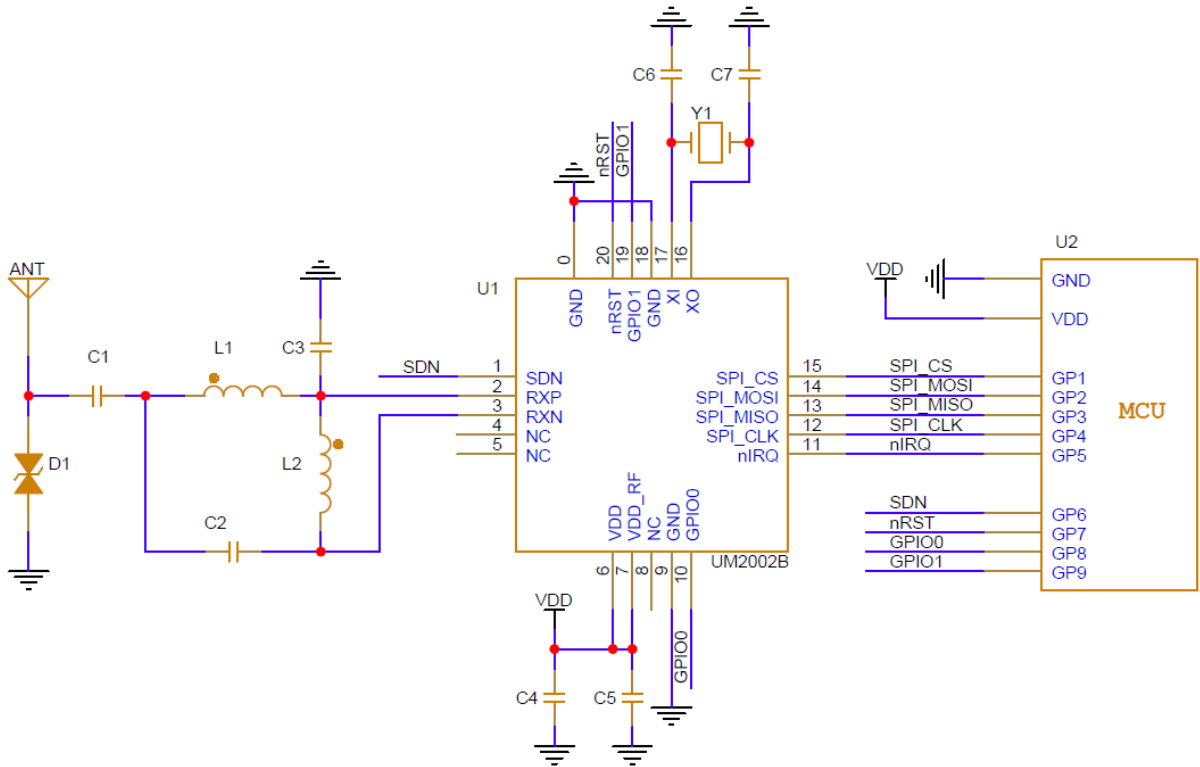


图 9-1: 应用参考电路图

## 9.2 参考参数

表 9-1: 参考参数表

位号	描述	元件值				单位
		315	433.92	868	915	MHz
C1	±10%, 0402, X7R, 50V, YAGEO	220	220	220	220	pF
C2	±5%, 0402, NP0, 50V, YAGEO	5.6	3	2.4	2.4	pF
C3	±5%, 0402, C0G, 50 V,, YAGEO	4.7	8.2	3.3	3.3	pF
C4	±10%, 0402, X7R, 50V, SAMSUNG	2.2	2.2	2.2	2.2	uF
C5	±10%, 0402, X7R, 50V, SAMSUNG	100	100	100	100	nF
C6	±10%, 0402, X7R, 50V, YAGEO	12	12	12	12	pF
C7	±10%, 0402, X7R, 50V, YAGEO	12	12	12	12	pF
L1	±5%, 0402, 贴片绕线电感, Sunlord	68	33	15	15	nH
L2	±5%, 0402, 贴片绕线电感, Sunlord	82	33	12	12	nH
D1	ESD 保护二极管	NC	NC	NC	NC	NC
Y1	3225 贴片无源晶振±10PPM 12pF, YXC (扬兴晶振)	26	26	26	26	MHz



# 10 封装尺寸

## 10.1 QFN20 (4\*4mm)

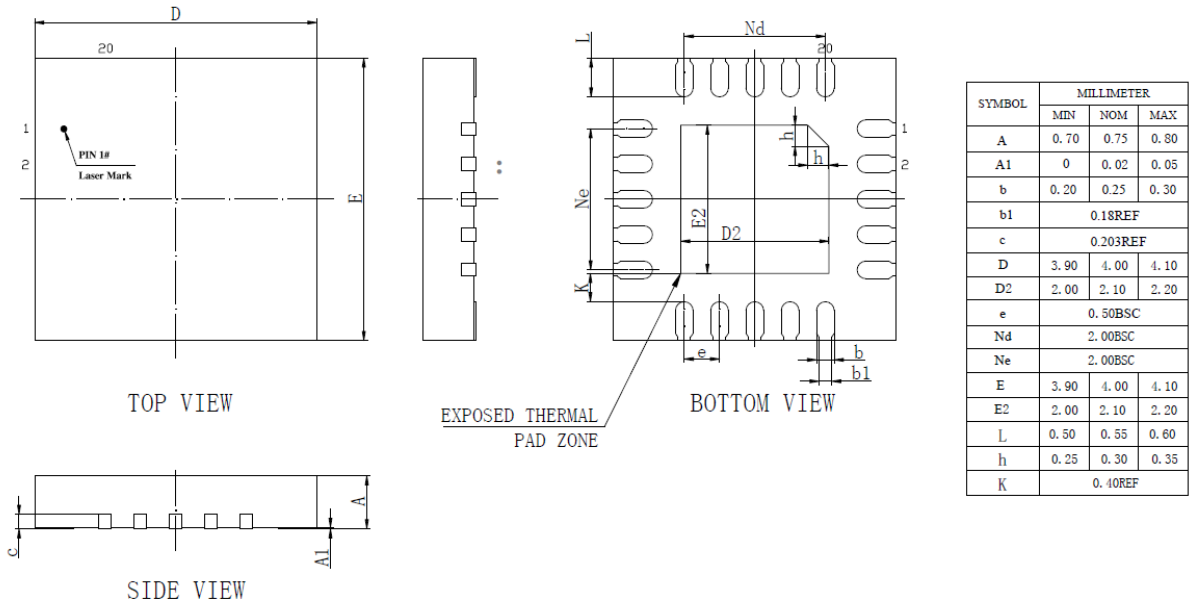


图 10-1: QFN20 封装图

# 11 版本维护

版本	日期	描述
V1.0	2022.08.15	初始版
V1.1	2023.09.17	芯片名称更改为 UM2002B; 修改部分寄存器描述; 更新电气参数值, 且表格重新排版; 调整功能描述、芯片运行、数据处理机制内容及排版。
V1.2	2024.07.20	删除原 1.2 主要特性; 修改部分引脚功能描述; 删除 Reg1B & Reg1D & Reg30& Reg44-48 & Reg5A 相关描述, 并更新部分寄存器描述; 调整第 5 章功能描述内容; 更新应用参考原理图; 删除电气参数章节; 更换字体并更新排版。