



# 远系列接收模块

## PY32F002A MCU 解码使用说明

面向初级用户的接线、下载、串口观察、源码理解、二次开发与故障排查指南

### 本文定位

本文说明如何使用远系列 ASK/OOK 接收模块接入 PY32F002A MCU，并运行示例程序直接解码遥控器/发射模块的数据。本文不等同于完整产品固件；对码、存储、按键匹配和输出控制需要客户按项目需求扩展。

| 项目     | 内容   |
|--------|--|
| 适用对象   | 第一次接触无线遥控接收、会基本使用 Keil/串口助手的初级工程师或技术客户                       |
| 示例平台   | 无线应用 MCU DEMO 板 (PY32F002A) + 远-R1L/远系列接收模块                  |
| 核心目标   | 接线后按遥控器，串口输出 3 字节 RF 编码，例如 RF:12 34 56                       |
| 建议配套工具 | Keil MDK、Puya PY32F0xx DFP、SWD 下载器、Type-C 数据线、串口助手、开发助手/信号助手 |
| 版本     | V1.0   |



## 1. 先看结论

### 一页速记

远系列接收模块把无线遥控信号解调成 DAT 脚上的高低电平脉冲; PY32F002A 用 PA1 外部中断捕获 DAT 边沿, 用 TIM1 计算脉宽, 用 rf.c 把 24bit 数据解出来, 再通过 USART1 串口打印。

### 1.1 初级用户怎么读本文

本文内容比较全, 但初次使用时不要从源码开始读。建议按下面路线走, 先得到一个可见的成功结果, 再逐步深入。

| 你的目标                  | 先看哪些章节      | 做到什么就算成功                    |
|-----------------------|-------------|-----------------------------|
| 只想确认模块能用              | 1、3、4、5、12  | 按遥控器后, 串口出现 RF:xx xx xx。    |
| 想接到自己的板子              | 3、6、9、11、12 | 自己的 MCU 能读到稳定 rf_us 脉宽。     |
| 想改成产品功能               | 7、8、13      | 能学习保存码值, 匹配后执行动作。           |
| 想移植到<br>51/STM8/STM32 | 9、10、11     | 目标平台能做到 DAT 边沿中断 + 1us 定时器。 |
| 遇到问题                  | 5、11、12     | 先判断是硬件、定时器、中断、串口还是 RF 信号问题。 |

### 1.2 最小成功标准

#### 第一次调试只追求一个结果

请先不要改学习码、Flash、继电器、低功耗等功能。第一次调试的唯一目标是: 模块接好、串口打开、按遥控器后能看到 RF:xx xx xx。只要能稳定看到这个输出, 说明无线接收、电源、天线、DAT 接线、中断和定时器主链路已经跑通。

| 阶段  | 检查项                     | 通过标准                                   |
|-----|-------------------------|--|
| 上电前 | 模块 VCC/GND/DAT/ANT 是否接对 | VCC 和 GND 不反接, DAT 接 PA1/RFDAT1, 天线已接。 |



|     |                      |                                       |
|-----|----------------------|---------------------------------------|
| 下载前 | SWD 接线和目标芯片          | Keil 能识别 PY32F002Ax5, 工程能 0 Error 编译。 |
| 串口前 | Type-C 数据线和 CH340 串口 | 电脑能看到串口号, 串口助手设置 9600, 8N1。           |
| 按键后 | RF 码值输出              | 每次按键能出现 RF:xx xx xx, 近距离码值基本一致。       |
| 拉远后 | 距离和稳定性               | 距离变远后输出减少是正常现象, 先优化天线和电源。             |

### 1.3 初次使用先不要改的地方

- 不要一开始就改 RF\_SYN\_MIN/MAX、RF\_PULSE\_MIN/MAX/MID。先用默认值跑通。
- 不要一开始就写 Flash 或 EEPROM。Flash 学习码逻辑应在串口能稳定输出后再做。
- 不要在中断函数里打印串口、延时、控制继电器或写存储器。
- 不要同时改电压、天线、引脚、主频和定时器。一次只改一个变量, 方便定位问题。

### 1.4 常用术语

| 术语         | 初级用户理解                                     |                                      |
|------------|--|--------------------------------------|
| ASK/OOK    | 一种常见遥控无线调制方式。客户通常不需要处理射频本身, 只看接收模块 DAT 输出。 |                                      |
| DAT        | 接收模块的数据输出脚, 会输出高低电平脉冲, MCU 根据脉宽解码。         |                                      |
| 同步头        | 一帧数据开始前较长的一段脉冲, 用来告诉 MCU: 后面要开始收数据。        |                                      |
| 脉宽         | 某个高电平或低电平持续了多久, 本文主要测低电平宽度, 单位是微秒。         |                                      |
| 24bit/3 字节 | 本例程把一帧数据解析成 3 个字节, 例如 RF:12 34 56。         |                                      |
| 二次校验       | 连续两帧数据一致才认为有效, 减少干扰误触发。                    |                                      |
| 学习码        | 产品把某个遥控器按键码保存下来, 之后只响应保存过的码。               |                                      |
| 输入捕获       | 定时器硬件自动记录边沿时间, 比普通 GPIO 中断更精确, 但配置更复杂。     |                                      |
| 事项         | 本例程配置                                      | 客户需要确认                               |
| 模块供电       | DEMO 板默认 VCC=3.3V (R6=0R, R7=NC)           | 模块和 MCU 电平必须匹配; 如改 5V, 确认 MCU IO 可承受 |



|        |   |                               |
|--------|---|-------------------------------|
| 接收数据脚  | 远-R1x 的 DATA/DAT -> RFDAT1 -> PA1           | DAT 只接一个即可; 建议浮空输入, 不要额外强拉    |
| SHUT 脚 | 部分模块 SHUT -> PA0/RF_SHUT                    | 无 SHUT 的模块可忽略; 有 SHUT 时确认有效电平 |
| 串口输出   | USART1, 9600bps, 8N1, 经 CH340N 到 USB Type-C | 电脑串口助手选择正确 COM 口和 9600 波特率    |
| 解码结果   | 连续两帧一致后打印 RF:xx xx xx                       | 输出的是示例码值, 不是已经完成按键动作逻辑        |

## 2. 资料包内容

本地目录中与客户使用直接相关的资料如下。HAL/CMSIS 库文件很多, 初级客户不需要逐个阅读, 先掌握下表即可。

| 文件/目录   | 用途                      | 初级用户阅读建议                                    |
|---|-------------------------|---|
| 远-R1L 接收模块规格书_V1.3.pdf                                    | 接收模块的参数、引脚、天线、开发工具说明    | 必须阅读: 参数、引脚、天线、注意事项                         |
| MCU_DEMO 原理图.pdf  | PY32F002A DEMO 板原理图     | 重点看 M1/M2/M6 模块座、U1 MCU、U8 CH340、R6/R7 电压选择 |
| PY32F002 远系列接收解码/程序说明.txt                                 | 工程师原始说明                 | 内容较短, 可作为本文的原始依据                            |
| Templates/PY32F002xx_Templates_LL/MDK-ARM/Project.uvprojx | Keil 工程文件               | 双击打开工程, 编译下载                                |
| Src/main.c  | 系统初始化和主循环               | 理解启动顺序                                      |
| Src/rf.c / rf.h   | 无线脉宽解码核心                | 二次开发最重要                                     |
| Src/myPy32drv.c / main_uart.c                             | GPIO、EXTI、TIM1、USART 配置 | 移植或改引脚时需要看                                  |
| PY32F002A 相关资料及 keil 支持包                                  | 芯片手册、DFP pack           | 环境安装和疑难问题时查看                                |



## 3. 硬件连接

### 3.1 模块引脚

远-R1L 引脚为 GND、DAT、DAT、VCC、ANT，其中两个 DAT 功能相同，接一个即可。



背面视图

| 远-R1x 引脚       | 连接到 DEMO 板 | 连接到<br>PY32F002A | 说明                             |
|----------------|------------|------------------|--------------------------------|
| VCC            | VCC        | VCC              | 接收模块电源正。DEMO 板默认 3.3V。         |
| GND            | GND        | VSS              | 电源地，必须与 MCU 共地。                |
| DATA           | RFDAT1     | PA1              | 接收数据输出脚，外部中断输入。两个 DATA 脚接一个即可。 |
| ANT            | 天线焊盘/弹簧天线  | -                | 必须接合适天线；天线不当会明显降低距离。           |
| SHUT/SC (部分型号) | RF_SHUT    | PA0              | 低电平使能，远-R6L 模块使用；无此脚时忽略。       |

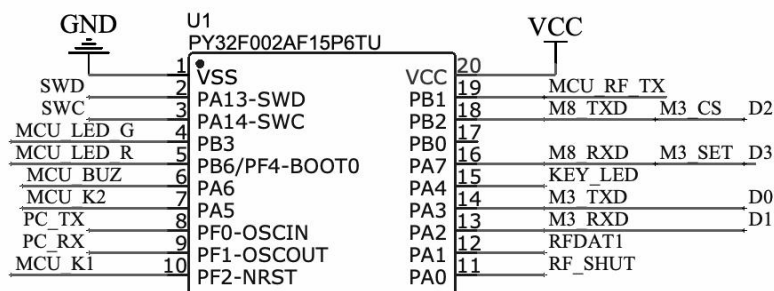
#### 电压选择

原理图中 R6=0R、R7=NC，默认把 V33 接到 VCC，也就是 3.3V。若客户需要让模块工作在 5V，需要改焊电阻选择，并同步确认 MCU 输入电平、模块 DAT 输出电平和系统电源设计。



## 3.2 DEMO 板关键连接

下图为原理图中 MCU 与接收模块、串口相关的关键区域。客户实际调试时只要抓住 PA1、PA0、USART1、VCC/GND 这几条线。



| 网络名         | MCU 引脚           | 例程宏/用途        | 注意事项                      |
|-------------|------------------|---------------|---------------------------|
| RFDAT1      | PA1              | p_rf_in       | 接收模块 DAT，双边沿触发中断。         |
| RF_SHUT     | PA0              | p_shut        | 部分远-R1x 模块的 SHUT 控制。      |
| MCU_RF_TX   | PB1              | p_rf_tx       | 例程保留发送 1527 的函数，接收演示可不使用。 |
| MCU_LED_G   | PB3              | p_led2        | 收到有效帧后翻转/闪烁，用于观察接收成功。     |
| PC_TX/PC_RX | PF0/PF1 经 CH340N | USART1 输出调试信息 | 串口助手设置 9600, 8N1。         |
| SWD/SWC     | PA13/PA14        | 下载调试          | 连接 SWD 下载器。               |

## 3.3 天线和电源

- 天线非常重要，不接天线/天线不匹配/靠近金属外壳/大面积地，都会明显缩短距离。
- 电源要稳定。接收模块灵敏度高，电源纹波、DC-DC 噪声、大电感磁场都会影响解码稳定性。
- 模块 VCC 附近建议保留 0.1uF 去耦，产品板上还可按实际负载增加 4.7uF/10uF。
- 调试距离问题时，先用信号助手看相对信号强弱，再用开发助手或本例程看码值是否稳定。



## 4. 开发环境与工程打开

| 项目       | 要求/路径  | 说明   |
|----------|--|--|
| Keil MDK | 建议 MDK5, 工程记录 ARMCC V5.06 update 5   | 如果用 ARM Compiler 6, 可能需要检查编译选项。                                    |
| 器件支持包    | PY32F002A 相关资料及 keil 支持包<br>/Puya.PY32F0xx_DFP.1.1.0.pack                          | 先安装 pack, 再打开工程。工程文件记录 PackID 为 1.2.3, 若 Keil 提示版本不一致, 选择已安装的兼容版本。 |
| 工程文件     | PY32F002 远系列接收解码<br>/Templates/PY32F002xx_Templates_LL<br>/MDK-ARM/Project.uvprojx | 双击打开。  |
| 目标芯片     | PY32F002Ax5  | Flash 20KB, SRAM 3KB。  |
| 编译宏      | PY32F002Ax5, USE_FULL_LL_DRIVER  | 工程已配置, 客户不要随意删除。   |
| 下载接口     | SWD: VCC、GND、SWDIO、<br>SWCLK, 可选 NRST  | 下载失败时降低 SWD Clock。   |
| 输出文件     | Objects/Project.hex  | 工程设置为生成 HEX。   |

### 4.1 第一次编译下载步骤

1. 安装 Keil MDK 和 Puya PY32F0xx DFP 支持包。
2. 用 Type-C 给 DEMO 板供电; 用 SWD 下载器连接 VCC、GND、SWDIO、SWCLK。
3. 打开 Project.uvprojx, 确认目标为 PY32F002Ax5\_Project。
4. 点击 Build, 看到 0 Error(s) 后再下载。
5. 下载完成后复位 MCU, 打开串口助手, 选择 CH340 对应串口。
6. 串口设置为 9600bps、8 数据位、无校验、1 停止位。
7. 按遥控器或让发射模块发码, 观察串口输出和 LED 闪烁。



## 5. 运行现象

| 动作         | 正常现象                         | 说明  |
|------------|------------------------------|---|
| 上电/复位      | 串口打印启动提示                     | 原始代码中提示文字编码可能因文件编码显示异常, 建议后续统一为 UTF-8 或 GB2312。 |
| 无遥控信号      | 串口无 RF 码值输出, LED 保持默认状态      | 这是正常状态。   |
| 按下已配套发射器按键 | 串口输出 RF:xx xx xx, LED2 翻转/闪烁 | 只有连续两帧一致后才认为有效。                                 |
| 长按按键       | 可能每隔一段时间重复输出同一 RF 码值         | 取决于发射器重复帧周期和 active_10ms 计时。                    |
| 距离变远/天线不好  | 码值偶尔不稳定、输出减少或无输出             | 先检查天线、电源、频率和脉宽阈值。                               |

### 串口输出示例

```
RF:12 34 56
```

```
RF:12 34 56
```

上面的 3 个字节是示例解码结果。实际产品中通常要把其中一部分作为地址码, 一部分作为按键值, 具体取决于发射端编码方式。

### 5.1 没有示波器时怎么判断问题

初级客户现场不一定有示波器, 所以文档应给一套低门槛判断方法。先判断“有没有信号”, 再判断“码值稳不稳”, 最后才去改代码。

| 工具/现象               | 能判断什么            | 下一步                          |
|---------------------|------------------|------------------------------|
| LED2 会闪, 串口也有 RF 输出 | 接收链路基本正常         | 进入学习码、按键匹配或产品动作开发。           |
| LED2 会闪, 串口无 RF 输出  | RF 可能已收到, 串口链路异常 | 检查 CH340、串口号、波特率、PF0/PF1 复用。 |





|                    |                           |                                    |
|--------------------|---------------------------|------------------------------------|
| LED2 不闪, 串口无 RF 输出 | 可能没收到信号或中断没进              | 检查 VCC/GND/DAT/ANT、PA1 中断、频率是否匹配。  |
| 开发助手能看到码, 本例程看不到   | 发射端正常, MCU 接线/中断/定时器可能有问题 | 重点查 DAT 到 PA1、EXTI LINE1、TIM1 1us。 |
| 信号助手显示很弱           | 无线链路或天线问题                 | 近距离测试, 换天线, 远离电源噪声和金属外壳。           |
| 近距离码稳定, 远距离不稳定     | 阈值、电源、天线或环境影响             | 先优化硬件, 再考虑调整 RF_PULSE_*。           |

## 6. 解码原理

远系列接收模块输出的是已经解调后的数字脉冲。MCU 不直接处理射频载波, 而是处理 DAT 脚上的高低电平时间。

| 步骤 | 代码位置                 | 做了什么                          | 初级用户理解                     |
|----|----------------------|-------------------------------|----------------------------|
| 1  | EXTI0_1_IRQHandler() | PA1 有边沿变化时进入中断                | 每次 DAT 从高到低或低到高都会通知 MCU。   |
| 2  | REC_RF_us_INT()      | 读取 TIM1->CNT 得到刚才那段低电平脉宽      | TIM1 每 1us 计数一次, 所以数值就是微秒。 |
| 3  | 同步头判断                | 6ms-18ms 认为是一帧开始              | 先找到一帧的起点, 再开始收 24bit。      |
| 4  | 位判断                  | <600us 记为 1, >=600us 记为 0     | 本例只按低电平宽度区分 0/1。           |
| 5  | 3 字节组包               | 每 8bit 存入 rf_dat[0..2]        | 一帧共 24bit。                 |
| 6  | 二次校验                 | 连续两帧与 rf_dat_buf 一致才置 b_rx_on | 降低误码导致误触发的概率。              |
| 7  | 主循环处理                | CHK_RF_DECODE() 打印码值          | 中断只快速收码, 主循环再做业务处理。        |



## 6.1 关键脉宽参数

| 宏/参数         | 当前值     | 含义         | 调试建议                      |
|--------------|---------|------------|---------------------------|
| RF_SYN_MIN   | 6000us  | 同步头最小宽度    | 如果发射端同步头偏短, 可适当降低。        |
| RF_SYN_MAX   | 18000us | 同步头最大宽度    | 过大容易误识别干扰, 过小可能收不到远距离信号。  |
| RF_PULSE_MIN | 200us   | 有效数据脉冲最小宽度 | 过滤过短毛刺。                   |
| RF_PULSE_MAX | 1600us  | 有效数据脉冲最大宽度 | 过滤异常长脉冲。                  |
| RF_PULSE_MID | 600us   | 0/1 判断分界   | 本例中低于 600us 记为 1, 否则记为 0。 |
| TIM1 计数      | 1us/计数  | 脉宽测量时间基准   | 换主频或换 MCU 时必须保持等效 1us。    |

### 为什么要留 50% 左右的容差

无线遥控产品受距离、电源、天线、环境干扰和发射端误差影响, 脉宽会有抖动。阈值过窄看似严格, 实际会缩短接收距离或导致码值不稳定; 阈值过宽又可能把噪声当成数据。建议先用示波器或开发助手测实际脉宽, 再调整 RF\_PULSE\_MIN/MAX/MID。

## 6.2 从发射波形到接收解码

接收端最容易让初学者困惑: 串口最后打印的是 3 个字节, 但接收模块 DAT 脚上实际出现的是一串高低电平。程序要做的事, 就是把这些高低电平按时间重新翻译成 0 和 1。

| 发射端动作     | 接收模块 DAT 上看到的结果       | 接收程序怎么理解                                   |
|-----------|-----------------------|--|
| 发射同步头     | 一段明显很长的低电平, 约 12ms 左右 | rf_us 落在 RF_SYN_MIN/RF_SYN_MAX 之间, 认为一帧开始。 |
| 发射逻辑 1    | 低电平较短, 约 400us        | rf_us < RF_PULSE_MID, 写入 bit=1。            |
| 发射逻辑 0    | 低电平较长, 约 1200us       | rf_us >= RF_PULSE_MID, 写入 bit=0。           |
| 连续发 24bit | 同步头后出现 24 个数据低电平宽度    | 每 8bit 组成 1 字节, 最终得到 3 字节。                 |
| 重复发多帧     | 相同 24bit 数据重复出现       | 连续两帧一致后才置 b_rx_on, 降低误触发。                  |

### 低电平宽度判定规则



接收端只测“低电平持续时间”的简化解理解:

同步头低电平: 约 6000us 到 18000us -> 一帧开始

数据位低电平: 约 200us 到 1600us -> 有效数据位

如果低电平 < 600us -> 记为 1

如果低电平 >= 600us -> 记为 0

### 6.3 默认 11 22 33 如何被接收端还原

为了让客户能把示波器、程序和串口输出对上, 下面以发射端默认码 0x11 0x22 0x33 为例, 反向说明接收程序如何恢复出 24bit 数据。

| 十六进制           | 二进制      |   | 接收端最终打印             |                      |
|----------------|----------|---|---------------------|----------------------|
| 0x11           | 00010001 |   | RF:11 xx xx 的第 1 字节 |                      |
| 0x22           | 00100010 |   | RF:xx 22 xx 的第 2 字节 |                      |
| 0x33           | 00110011 |   | RF:xx xx 33 的第 3 字节 |                      |
| 目标字节           | 位序       | 值 | DAT 低电平宽度           | 程序判断                 |
| rf_dat[0]=0x11 | bit7     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit6     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit5     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit4     | 1 | 短低电平, 约 400us       | rf_us < 600 -> 写入 1  |
| rf_dat[0]=0x11 | bit3     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit2     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit1     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[0]=0x11 | bit0     | 1 | 短低电平, 约 400us       | rf_us < 600 -> 写入 1  |
| rf_dat[1]=0x22 | bit7     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[1]=0x22 | bit6     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[1]=0x22 | bit5     | 1 | 短低电平, 约 400us       | rf_us < 600 -> 写入 1  |
| rf_dat[1]=0x22 | bit4     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[1]=0x22 | bit3     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[1]=0x22 | bit2     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |
| rf_dat[1]=0x22 | bit1     | 1 | 短低电平, 约 400us       | rf_us < 600 -> 写入 1  |
| rf_dat[1]=0x22 | bit0     | 0 | 长低电平, 约 1200us      | rf_us >= 600 -> 写入 0 |



|                |      |   |                |                      |
|----------------|------|---|----------------|----------------------|
| rf_dat[2]=0x33 | bit7 | 0 | 长低电平, 约 1200us | rf_us >= 600 -> 写入 0 |
| rf_dat[2]=0x33 | bit6 | 0 | 长低电平, 约 1200us | rf_us >= 600 -> 写入 0 |
| rf_dat[2]=0x33 | bit5 | 1 | 短低电平, 约 400us  | rf_us < 600 -> 写入 1  |
| rf_dat[2]=0x33 | bit4 | 1 | 短低电平, 约 400us  | rf_us < 600 -> 写入 1  |
| rf_dat[2]=0x33 | bit3 | 0 | 长低电平, 约 1200us | rf_us >= 600 -> 写入 0 |
| rf_dat[2]=0x33 | bit2 | 0 | 长低电平, 约 1200us | rf_us >= 600 -> 写入 0 |
| rf_dat[2]=0x33 | bit1 | 1 | 短低电平, 约 400us  | rf_us < 600 -> 写入 1  |
| rf_dat[2]=0x33 | bit0 | 1 | 短低电平, 约 400us  | rf_us < 600 -> 写入 1  |

### 示波器对照方法

先找到最长的低电平, 那是同步头。同步头后面开始数 24 个低电平宽度: 短低电平记 1, 长低电平记 0。按上表数完, 应得到 00010001 00100010 00110011, 串口打印 RF:11 22 33。

## 6.4 接收状态机变量说明

理解接收程序时, 先把下面几个变量看懂。它们共同描述“当前有没有找到同步头、收到了第几位、是否通过两帧校验”。

| 变量             | 类型/范围   | 作用                    | 初学者理解                              |
|----------------|---------|-----------------------|------------------------------------|
| b_rx_start     | 0/1     | 是否已经找到同步头, 是否开始接收数据位。 | 等于 1 表示“后面的有效脉宽要按 0/1 来收”。         |
| b_rx_on        | 0/1     | 是否已经收到连续两帧一致的有效码。     | 中断置 1, 主循环 CHK_RF_DECODE() 处理后清 0。 |
| rf_rx_cnt      | 0-7     | 当前这个字节已经收了几个 bit。     | 收满 8bit 后清 0, 进入下一个字节。             |
| rf_array       | 0-2     | 当前正在写 rf_dat 的第几个字节。  | 0 是第 1 字节, 1 是第 2 字节, 2 是第 3 字节。   |
| rf_dat[3]      | 3 字节数组  | 本次正在接收的一帧数据。          | 中断逐 bit 写入这里。                      |
| rf_dat_buf[3]  | 3 字节数组  | 上一次保存的完整帧, 也作为最终输出缓冲。 | 新帧和它一致时, 才认为两帧一致。                  |
| RF.active_10ms | 10ms 单位 | 接收活跃窗口, 避免重复打印太密。     | 设置为 15 表示约 150ms。                  |
| rf_us          | 微秒值     | 刚刚测到的低电平宽度。           | 解码 0/1 的直接依据。                      |



## 6.5 REC\_RF\_us\_INT() 的完整判断流程

这个函数是接收端最核心的函数。它不是一次性收到 3 个字节，而是每来一个边沿处理一小步，靠状态变量慢慢拼出完整 24bit。

### 边沿处理逻辑

DAT 下降沿，也就是 `p_rf_in == 0`：

1. `TIM1->CNT = 0`

2. 立即 `return`

解释：低电平刚开始，先从 0 计时。

DAT 上升沿，也就是 `p_rf_in == 1`：

1. `rf_us = TIM1->CNT`

2. `TIM1->CNT = 0`

3. 用 `rf_us` 判断刚刚那段低电平是同步头、数据 1、数据 0，还是噪声。

| 判断顺序 | 条件   | 程序动作  | 原因                   |
|------|--|---|----------------------|
| 1    | <code>b_rx_on</code> 已经为 1                             | 直接 <code>return</code>  | 上一组有效码还没被主循环处理，避免覆盖。 |
| 2    | <code>RF_SYN_MIN &lt; rf_us &lt; RF_SYN_MAX</code>     | <code>b_rx_start=1</code> , <code>rf_rx_cnt=0</code> ,<br><code>rf_array=0</code> | 找到同步头，重新开始收一帧。       |
| 3    | <code>b_rx_start == 0</code>                           | 直接 <code>return</code>  | 还没找到同步头，普通脉冲不能当数据。   |
| 4    | <code>rf_us</code> 超出<br><code>RF_PULSE_MIN/MAX</code> | <code>b_rx_start=0</code> ，丢弃本帧   | 数据脉宽异常，说明本帧不可靠。      |
| 5    | 有效数据脉宽   | <code>rf_dat[rf_array] &lt;= 1</code>   | 为新 bit 腾出最低位。        |
| 6    | <code>rf_us &lt; RF_PULSE_MID</code>                   | <code>rf_dat[rf_array] += 1</code>  | 短低电平代表 1；长低电平默认保持 0。 |
| 7    | <code>rf_rx_cnt</code> 收满 8                            | 进入下一个字节   | 每 8bit 组成 1 字节。      |
| 8    | <code>rf_array</code> 收满 3                             | 进入两帧比较  | 3 字节就是 24bit。        |
| 9    | 本帧等于 <code>rf_dat_buf</code>                           | <code>b_rx_on=1</code> ，LED 翻转  | 连续两帧一致，认为有效。         |
| 10   | 本帧不等于 <code>rf_dat_buf</code>                          | 保存本帧到 <code>rf_dat_buf</code>   | 先记为第一帧，等待下一帧再确认。     |



## 6.6 为什么要连续两帧一致

无线接收模块在没有真实遥控信号时, 也可能因为环境噪声输出一些毛刺。毛刺偶尔可能刚好符合某些脉宽条件。如果只收到一帧就执行动作, 产品可能误触发。

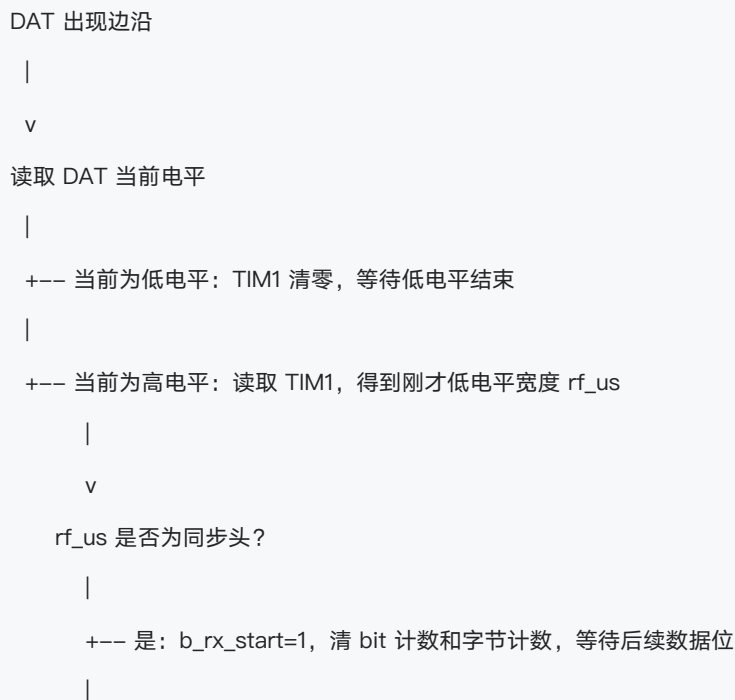
| 做法           | 优点               | 风险               |
|--------------|------------------|------------------|
| 收到一帧立即输出     | 响应最快, 代码最简单。     | 抗干扰差, 批量产品容易误触发。 |
| 连续两帧一致再输出    | 抗干扰明显更好, 仍然响应较快。 | 需要发射端重复发送多帧。     |
| 连续多帧或加入地址白名单 | 更安全, 适合门控/安防类。   | 响应稍慢, 逻辑更复杂。     |

### 接收端和发射端要配套

发射端如果只发送 1 帧, 接收端可能永远等不到“两帧一致”。所以调接收时建议先用成熟 1527 遥控器, 或者让 MCU 发射端重复发送 25-30 帧。

## 6.7 完整解码流程图

### 接收解码状态机





```
+++ 否: 是否已经找到同步头?  
|  
+++ 否: 忽略这个脉冲  
|  
+++ 是: rf_us 是否为有效数据脉宽?  
|  
+++ 否: 丢弃本帧, 等待下一个同步头  
|  
+++ 是: 按 RF_PULSE_MID 判断 0/1, 写入 rf_dat  
|  
v  
是否收满 24bit?  
|  
+++ 否: 继续等待下一位  
|  
+++ 是: 与 rf_dat_buf 比较  
|  
+++ 相同: b_rx_on=1, 主循环可以打印 RF:xx xx xx  
|  
+++ 不同: 保存为第一帧, 等待下一帧确认
```



## 6.8 初学者最容易误解的点

| 误解               | 正确理解   | 怎么验证                                |
|------------------|--|-------------------------------------|
| 接收模块直接输出串口数据     | 接收模块只输出 DAT 脉冲, 串口 RF:xx xx xx 是 MCU 解码后打印的。 | 断开 MCU 串口不会影响 DAT 波形。               |
| 只要有 DAT 波形就一定能解码 | 还要看脉宽是否落在同步头和数据位阈值内。                         | 示波器测低电平宽度。                          |
| 只开一个边沿中断也可以      | 本例需要下降沿清零、上升沿读数, 推荐双边沿。                      | 按键时统计上升沿和下降沿是否都进中断。                 |
| TIM1 大概计时就行      | RF_PULSE_MID 是 600us, 定时误差太大会直接把 0/1 判反。     | 用已知 400us/1200us 波形校准。              |
| 收到第一帧就应该打印       | 示例要求两帧一致后才打印。                                | 看 b_rx_on 是否在第二帧后置 1。               |
| EXTI line 可以随便选  | DAT 接 PA1, EXTI source 必须选 LINE1。            | 检查 LL_EXTI_SetEXTISource 是否为 LINE1。 |
| 可以在中断里 printf    | 串口打印很慢, 会影响后续边沿捕获。                           | 中断里只做轻量解码, 打印放主循环。                  |
| 调不通就先改阈值         | 阈值最后再改, 先确认接线、频率、天线、定时器和中断。                  | 按 04 排查清单逐项确认。                      |





## 7. 源码详细导读

### 7.1 main.c: 启动顺序和主循环

建议给客户看的 main.c 注释版

```
int main(void)
{
    APP_SystemClockConfig();    // 配置系统时钟, 当前使用 HSI 8MHz
    RCC_INI();                  // 打开 GPIOA/GPIOB/GPIOF、USART1、TIM1 等外设时钟
    GPIO_INI();                  // 初始化普通 GPIO 输入/输出
    APP_ConfigUsart(USART1);     // 配置 USART1: 9600bps、8N1
    SET_PF1_for_UART_or_GPIO(1); // 将 PF0/PF1 复用为 USART1, 引出到 CH340N
    APP_ConfigTIM1();             // 配置 TIM1 为 1us 计数, 用于测 DAT 脉宽
    GPIO_rf_INT_en_or_dis(1);    // 开启 PA1 外部中断, 双边沿触发
    systick_init();              // SysTick 每 10ms 中断一次
    myDelay_ms(50);              // 等待电源和外设稳定
    USER_INI();                 // 清零业务变量

    while (1)
    {
        SEC_EXE();               // 周期性打印/LED 状态处理
        CHK_RF_DECODE();         // 如果收到有效 RF 数据, 在这里处理和打印
    }
}
```

| 函数                      | 作用                      | 客户改动建议                   |
|-------------------------|-------------------------|--------------------------|
| APP_SystemClockConfig() | 设置 HSI 8MHz、AHB/APB 不分频 | 换主频时同步修改 TIM1 和 SysTick。 |
| GPIO_INI()              | 设置 PA/PB/PF 输入输出        | 换引脚时从这里和 ALL.H 的宏一起改。    |
| APP_ConfigUsart()       | 初始化串口                   | 只改波特率时改 BaudRate。        |
| APP_ConfigTIM1()        | 配置 1us 计时               | 移植核心, 必须保证 1us 分辨率。      |



|                         |             |                   |
|-------------------------|-------------|-------------------|
| GPIO_rf_INT_en_or_dis() | 打开 DAT 外部中断 | 确认 EXTI 线和端口选择正确。 |
| CHK_RF_DECODE()         | 主循环处理收到的码值  | 客户业务逻辑应优先写在这里。    |

## 7.2 rf.c: 接收中断解码

### REC\_RF\_us\_INT() 注释版

```
void REC_RF_us_INT(void)
{
    u16 rf_us;

    if (p_rf_in == 0)          // DAT 变为低电平: 从现在开始计时
    {
        TIM1->CNT = 0;
        return;
    }

    rf_us = TIM1->CNT;          // DAT 变为高电平: 读取刚才低电平持续时间
    TIM1->CNT = 0;

    if (b_rx_on) return;        // 上一帧还没处理完, 先不接收新数据

    if (rf_us > RF_SYN_MIN && rf_us < RF_SYN_MAX)
    {
        b_rx_start = 1;        // 找到同步头, 准备开始接收 24bit 数据
        rf_rx_cnt = 0;
        rf_array = 0;
        return;
    }

    if (b_rx_start == 0) return; // 还没找到同步头, 其他脉冲都忽略

    if (rf_us < RF_PULSE_MIN || rf_us > RF_PULSE_MAX)
```



```
{
    b_rx_start = 0;          // 脉宽异常, 丢弃本帧
    return;
}

rf_dat[rf_array] <= 1;      // 腾出最低位
if (rf_us < RF_PULSE_MID)
    rf_dat[rf_array] += 1;   // 短低电平记为 1, 长低电平记为 0

if (++rf_rx_cnt != 8) return; // 每 8bit 组成 1 字节
rf_rx_cnt = 0;

if (++rf_array < 3) return; // 共接收 3 字节, 即 24bit
b_rx_start = 0;

if (rf_dat_buf[0] == rf_dat[0] &&
    rf_dat_buf[1] == rf_dat[1] &&
    rf_dat_buf[2] == rf_dat[2])
{
    b_rx_on = 1;          // 连续两帧一致, 认为有效
    LL_GPIO_TogglePin(p_led2); // 翻转 LED, 提示收码成功
}
else
{
    rf_dat_buf[0] = rf_dat[0]; // 第一帧先保存, 等下一帧对比
    rf_dat_buf[1] = rf_dat[1];
    rf_dat_buf[2] = rf_dat[2];
}
}
```

### 中断里为什么不要做复杂业务

DAT 边沿来得很快, 中断函数必须尽量短。建议中断中只完成脉宽测量和组包, 学习码、Flash 存



储、继电器控制、串口协议解析等业务放到主循环或定时任务中做。

## 7.3 CHK\_RF\_DECODE(): 有效码值处理入口

客户业务建议从这里扩展

```
void CHK_RF_DECODE(void)
{
    if (b_rx_on == 0) return;      // 没收到有效码, 直接返回
    b_rx_on = 0;                   // 取走本次有效码

    if (rf_dat_buf[0] == 0xff &&
        rf_dat_buf[1] == 0xff &&
        rf_dat_buf[2] == 0xff) return; // 过滤异常全 1

    if (rf_dat_buf[0] == 0 &&
        rf_dat_buf[1] == 0 &&
        rf_dat_buf[2] == 0) return;  // 过滤异常全 0

    if (RF.active_10ms == 0)
        printf("RF:%02X %02X %02X\r\n",
            rf_dat_buf[0], rf_dat_buf[1], rf_dat_buf[2]);

    RF.active_10ms = 15;           // 150ms 活跃窗口, 避免过密重复处理
}
```

- 学习模式: 在这里把当前 rf\_dat\_buf 保存到 Flash 或 RAM 列表。
- 匹配模式: 把当前 rf\_dat\_buf 与已保存的码值逐个比较。
- 输出控制: 匹配成功后控制继电器、LED、电机、蜂鸣器或串口上报。
- 安全策略: 可增加长按过滤、重复码过滤、地址码白名单、按键值映射等逻辑。



## 7.4 SysTick 与 RF\_TIM\_10ms\_CB()

SysTick 每 10ms 调用一次 RF\_TIM\_10ms\_CB(), 用于串口接收超时、工作计时、RF 活跃窗口和 1 秒节拍。

### 10ms 定时任务说明

```
void RF_TIM_10ms_CB(void)
{
    CHK_UART_rxTIMEOUT();      // 判断串口接收是否超时

    if (TIME.work_ms10 > 0)
        TIME.work_ms10--;      // 通用工作计时

    if (RF.active_10ms > 0)
    {
        if (--RF.active_10ms == 0)
        {
            rf_dat_buf[0] = 0;
            rf_dat_buf[2] = 0;    // 活跃窗口结束后清部分数据
        }
    }

    TIME.ms10++;
    if (TIME.ms10 >= 100)
    {
        TIME.ms10 = 0;
        SYS.b_tick = 1;         // 1 秒节拍, SEC_EXE() 使用
    }
}
```



## 7.5 需要修正或提醒的代码点

### 建议修正 EXTI 线选择

GPIO\_rf\_INT\_en\_or\_dis() 中 EXTI\_InitStructure.Line 使用 LL\_EXTI\_LINE\_1, 但后面 LL\_EXTI\_SetEXTISource() 写成 LL\_EXTI\_CONFIG\_LINE0。由于 DAT 实际接 PA1, 中断处理也检查 LINE\_1, 建议改为 LL\_EXTI\_CONFIG\_LINE1。否则移植或换库版本时可能出现 PA1 中断不触发。

### EXTI source 建议修正

```
// 建议修改前
LL_EXTI_SetEXTISource(LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_LINE0);

// 建议修改后
LL_EXTI_SetEXTISource(LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_LINE1);
```

- rf.c 中 TX1527() 是发送演示函数, 本接收说明书不把它作为必须功能。
- 当前示例没有实现 Flash 存储学习码, 不能误以为已经具备掉电记忆。

## 8. 二次开发: 从演示到产品

### 8.1 推荐的数据结构

#### 学习码保存结构示例

```
typedef struct
{
    uint8_t code[3];    // 24bit 遥控码
    uint8_t key_id;     // 客户定义的按键编号, 例如 1=开, 2=关
    uint8_t action;     // 客户定义的动作, 例如继电器吸合、翻转、点动
} RF_UserCode;
```



## 8.2 学习对码流程

8. 用户长按产品上的学习键, 进入学习模式, LED 慢闪。
9. 清空临时接收标志, 等待遥控器按键。
10. 收到连续两帧一致的 rf\_dat\_buf 后, 检查是否为全 0/全 FF/已存在码。
11. 确认有效后保存到 Flash 或 EEPROM 模拟区, 同时记录按键功能。
12. 保存成功后 LED 快闪或蜂鸣提示, 退出学习模式。
13. 正常工作模式下收到码值时, 与保存列表比较, 匹配后执行动作。

## 8.3 产品逻辑建议

| 场景     | 建议做法                               | 原因            |
|--------|------------------------------------|---------------|
| 门禁/卷帘门 | 地址码必须学习保存; 动作前再次确认重复帧              | 降低误触发风险。      |
| 遥控开关   | 区分点动、互锁、自锁、定时关闭等模式                 | 不同客户需求差异大。    |
| 低功耗产品  | 无信号时 MCU 进入 sleep/stop, DAT 或定时器唤醒 | 延长电池寿命。       |
| 批量生产测试 | 用开发助手看码值, 用信号助手看信号强弱               | 提高测试效率, 减少误判。 |
| 多遥控器   | 保存多个 RF_UserCode, 增加删除/清空功能        | 方便售后和重新配对。    |



## 9. 移植到其他 MCU 的总原则

很多客户会使用 51 单片机、STM8、STM32 或其他国产 MCU。移植时不要先纠结芯片品牌，也不要整包搬 PY32 的 HAL/LL 库。远系列接收模块输出的是 DAT 高低电平脉冲，MCU 只要能准确测出低电平脉宽，就可以复用相同的解码思路。

| 必须移植的能力 | 本例程实现                | 其他 MCU 的要求                     | 验收方法                          |
|---------|----------------------|--------------------------------|-------------------------------|
| GPIO 输入 | PA1 读取 p_rf_in       | DAT 脚可读当前电平，输入模式不要配置成模拟。       | 空闲和按键时能读到电平变化。                |
| 边沿中断    | EXTI0_1_IRQHandler() | 优先支持双边沿；如果只能单边沿，需要改成输入捕获或轮询方案。 | 按键时中断计数持续增加。                  |
| 1us 定时器 | TIM1->CNT            | 每次边沿读取并清零计数器，计数单位为微秒或可换算为微秒。   | 示波器测 400us 时，软件读数接近 400。      |
| 周期任务    | SysTick 10ms         | 用于重复码窗口、超时、业务节拍。               | RF.active_10ms 能按 10ms 递减。    |
| 调试输出    | USART1 printf        | 非必须，但初期强烈建议保留串口或 LED 调试。       | 收到码时能看到 RF:xx xx xx 或 LED 指示。 |

### 移植时绝对不能变的 5 件事

1. RF\_SYN\_MIN/MAX、RF\_PULSE\_MIN/MAX/MID 的单位仍然必须是微秒。
2. REC\_RF\_us\_INT() 中读取到的 rf\_us 必须代表一个低电平持续时间。
3. 中断里不要 printf、不要写 Flash、不要长延时。
4. DAT 输入不要被强上拉、强下拉或外部 RC 拉慢边沿。
5. 主频、定时器分频、编译优化改变后，必须重新确认脉宽读数。





## 9.1 推荐统一接口层

建议把底层 MCU 差异封装成下面几个函数。客户只需要按自己的芯片实现这些函数，解码状态机就可以保持一致。

### 统一接口伪代码

```
// 平台无关接口：客户按自己的 MCU 实现

uint8_t RF_DAT_Read(void);    // 读取 DAT 当前电平：0=低，1=高

void RF_Timer_Reset(void);    // 清零微秒定时器

uint16_t RF_Timer_GetUs(void); // 读取当前微秒计数

void RF_OnEdgeInterrupt(void); // DAT 边沿中断中调用，相当于 REC_RF_us_INT()

void RF_10msTask(void);       // 每 10ms 调用一次，相当于 RF_TIM_10ms_CB()


// 推荐中断流程

// DAT 下降沿：RF_Timer_Reset()

// DAT 上升沿：rf_us = RF_Timer_GetUs(); RF_Timer_Reset(); 进入脉宽解码
```

## 9.2 移植决策图

### 不同 MCU 的移植路线选择

DAT 引脚是否支持双边沿中断？

是 -> 使用原例程思路：下降沿清零计时，上升沿读取低电平宽度。

否 -> 是否支持定时器输入捕获？

是 -> 用输入捕获记录下降沿/上升沿时间差，得到低电平宽度。

否 -> 是否能用较高频率定时扫描 DAT？

是 -> 可做轮询采样，但占用 CPU，抗干扰较差，不推荐初级客户优先使用。

否 -> 建议更换支持 EXTI 或输入捕获的引脚/MCU。



## 9.3 平台差异速查

| 平台     | 推荐方式                     | 主要难点                                 | 文档中应提醒客户                                  |
|--------|--------------------------|--------------------------------------|---|
| 51 单片机 | 外部中断 + 定时器; 或<br>定时器轮询   | 传统 8051 中断边沿能力有限,<br>12T/1T 定时器换算容易错 | 先确认芯片型号、晶振、定<br>时器模式和 INT0/INT1 能<br>力。   |
| STM8   | GPIO EXTI + TIM2/TIM4    | 时钟树和预分频配置;<br>STM8S/STM8L 外设差异       | 先把 TIM 配成 1us, 再调 RF<br>解码。               |
| STM32  | EXTI + TIM; 可选输入捕<br>获   | HAL/LL/寄存器写法多, 工程结<br>构复杂            | 不要在 HAL 回调里做耗时业<br>务; TIM 计数频率设为<br>1MHz。 |
| 其他 MCU | GPIO 边沿中断 + 1us<br>timer | 中断响应延迟、IO 电平兼容、<br>低功耗唤醒             | 先验证脉宽读数, 再验证解<br>码。                       |

## 10. 常见 MCU 平台移植说明

### 10.1 51 单片机移植

51 型号差异大, 传统 8051、增强型 1T 8051、STC/Nuvoton/Silicon Labs 等实现方式可能不同。

| 项目           | 建议写法   | 常见问题   |
|--------------|--|--|
| DAT 接线       | 优先接 INT0/P3.2 或 INT1/P3.3; 如果芯片有更多外<br>部中断, 也可接支持中断的普通 IO。           | 接到普通 IO 但没有中断, 只能轮询,<br>容易丢边沿。                 |
| 中断方式         | 若支持双边沿, 按原例程移植; 若只支持下降沿, 则<br>下降沿清定时器, 上升沿需要用捕获/轮询补充。                | 传统 8051 常见外部中断只支持低电平<br>或下降沿。                  |
| 定时器          | 用 16 位定时器测微秒。12MHz 传统 12T 8051 下定<br>时器 1 计数约 1us; 1T 8051 要重新分频或换算。 | 把 12T 和 1T 混淆, 导致 400us 被读<br>成 33us 或 4800us。 |
| 定时器溢出        | 同步头最长约 18ms, 16 位 1us 计数最大 65ms, 足<br>够; 但要清除溢出标志。                   | 溢出标志未处理会导致异常大脉宽。                               |
| 串口调试         | 建议保留 9600bps 串口打印; 无串口时至少用 LED<br>显示收码成功。                            | 没有调试输出时客户无法判断是没收<br>到信号还是业务没执行。                |
| Flash/EEPROM | 学习码保存方式按具体 51 型号实现, 注意擦写寿命<br>和掉电保护。                                 | 在中断中写 Flash 会导致丢码或死机。                          |



## 51 外部中断移植框架

```
// 51 思路示例, 非完整可编译代码
void INT0_ISR(void) interrupt 0
{
    if (RF_DAT_Read() == 0)
    {
        RF_Timer_Reset();      // 下降沿: 开始测低电平
    }
    else
    {
        rf_us = RF_Timer_GetUs(); // 上升沿: 得到低电平宽度
        RF_Timer_Reset();
        RF_DecodePulse(rf_us);   // 进入通用解码逻辑
    }
}
```

### 51 单片机特别提醒

如果客户的 51 外部中断不能双边沿触发, 不建议初级客户直接照搬原例程。优先选择支持双边沿中断的增强型 51, 或改用定时器输入捕获/高速轮询方案。文档中应要求客户先确认芯片手册, 而不是只看“8051 兼容”。



## 10.2 STM8 系列移植

STM8 一般可以用 GPIO 外部中断配合 TIM2/TIM4 完成。STM8S、STM8L、STM8AF 不同系列寄存器和库函数名字不同, 因此说明书应给方法和检查点, 不建议写死某一个库函数作为唯一答案。

| 项目     | 建议写法  | 常见问题                            |
|--------|---|---------------------------------|
| DAT 接线 | 接支持 EXTI 的 GPIO, 例如 PA/PB/PC/PD 某一路, 具体看封装和芯片手册。        | 同一端口中断共用入口, 清标志时清错线。            |
| 中断触发   | 配置为上升沿和下降沿均触发; 在中断里读 DAT 当前电平判断边沿方向。                    | 只配置下降沿, 无法得到低电平结束时间。            |
| 定时器    | TIM2/TIM4 配成 1MHz 计数, 1 个计数=1us; ARR 至少大于 20000。        | 系统时钟从 16MHz 改 8MHz 后, 预分频未同步修改。 |
| 库差异    | SPL、裸寄存器、IAR/Cosmic 工程写法不同, 核心仍是 GPIO EXTI + 1us timer。 | 客户复制不同编译器例程后头文件不匹配。             |
| 中断优先级  | DAT 中断优先级要高于普通业务, 但不要长时间关闭全局中断。                         | 串口打印或延时占用中断, 导致丢边沿。             |
| 低功耗    | 若进入 Halt/Active-halt, 确认 DAT 对应 EXTI 能唤醒, 唤醒后定时器重新初始化。  | 低功耗唤醒后时钟未恢复, 定时器不再是 1us。        |

### STM8 EXTI 移植框架

```
// STM8 思路示例, 非完整可编译代码
INTERRUPT_HANDLER(EXTI_PORTx_IRQHandler, vector)
{
    if (RF_DAT_Read() == 0)
    {
        RF_Timer_Reset();          // DAT 下降沿
    }
    else
    {
        uint16_t rf_us = RF_Timer_GetUs();
        RF_Timer_Reset();
        RF_DecodePulse(rf_us);     // DAT 上升沿, 解码低电平宽度
    }
    // 按具体 STM8 系列要求清除中断标志
}
```



## 10.3 STM32 系列移植

STM32 客户通常分为 HAL 用户、LL 用户和寄存器用户。应把核心要求说清楚：EXTI 负责捕获 DAT 边沿，TIMx 负责 1us 计时，业务处理仍放在主循环。

| 项目       | HAL 用户                                    | LL/寄存器用户   | 常见问题                            |
|----------|---|--|---------------------------------|
| DAT GPIO | CubeMX 配成 GPIO_EXTI, 触发选择 Rising/Falling  | 配置 MODER 输入、EXTI line、SYSCFG EXTI source、RTSR/FTSR           | CubeMX 只选了 Rising 或只选了 Falling。 |
| 中断入口     | HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) | EXTIxx_IRQHandler 中清 pending 后调用解码                           | 没有判断 pin, 其他 EXTI 也进来导致误处理。     |
| 定时器      | TIMx Prescaler 配到 1MHz 计数频率               | $PSC = \text{timer\_clock} / 1000000 - 1$ , $ARR \geq 20000$ | APB 分频后 TIM 时钟可能翻倍, PSC 算错。     |
| 输入捕获替代   | HAL_TIM_IC_CaptureCallback 可测边沿时间差        | 配置 CCx 双边沿或切换极性  | 输入捕获更准, 但对初级客户配置复杂。             |
| RTOS     | 中断中只置标志或轻量解码, 业务用任务处理                     | 注意临界区和 volatile  | 任务关中断时间过长, 漏边沿。                 |
| 低功耗      | Stop 后恢复系统时钟和 TIM                         | 唤醒后重新打开定时器/EXTI  | Stop 唤醒后 TIM 频率改变。              |

### STM32 HAL 移植框架

```
// STM32 HAL 思路示例, 非完整可编译代码
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin != RF_DAT_Pin) return;

    if (HAL_GPIO_ReadPin(RF_DAT_GPIO_Port, RF_DAT_Pin) == GPIO_PIN_RESET)
    {
        __HAL_TIM_SET_COUNTER(&htimx, 0);    // 下降沿: 开始测低电平
    }
    else
    {
        uint16_t rf_us = __HAL_TIM_GET_COUNTER(&htimx);
    }
}
```



```
__HAL_TIM_SET_COUNTER(&htimx, 0);  
RF_DecodePulse(rf_us);           // 上升沿: 解码低电平宽度  
}  
}
```

### STM32 定时器时钟提醒

STM32 的 TIM 时钟不一定等于系统主频。部分系列中, 当 APB 预分频不为 1 时, 定时器时钟可能是 PCLK 的 2 倍。文档中应提醒客户用示波器或调试变量验证: 400us 脉宽读数应接近 400, 而不是只相信 CubeMX 显示。

## 10.4 输入捕获方案何时使用

如果某些 MCU 不方便做双边沿 GPIO 中断, 但定时器支持输入捕获, 可以用输入捕获直接记录 DAT 边沿时间。该方案更准确, 但配置更复杂, 建议作为进阶方案。

| 方案                     | 优点            | 缺点                   | 推荐程度             |
|------------------------|---------------|----------------------|------------------|
| GPIO 双边沿中断 + 1us timer | 最容易理解, 最接近本例程 | 中断响应延迟会带来少量误差        | 初级客户首选           |
| 定时器输入捕获                | 硬件记录边沿, 精度更高  | 配置复杂, 平台差异大          | 中高级客户可选          |
| 定时轮询 DAT               | 不依赖外部中断       | 占用 CPU, 容易漏短脉冲, 抗干扰差 | 不推荐, 除非 MCU 资源受限 |



## 11. 移植常见问题专项排查

| 现象              | 最可能原因                         | 如何确认                                | 处理建议                              |
|-----------------|-------------------------------|-------------------------------------|-----------------------------------|
| 所有码都收不到         | DAT 引脚没有中断或中断线选错              | 按键时中断计数不增加                          | 检查 GPIO 端口映射、EXTI line、NVIC/全局中断。 |
| 能进中断但码不对        | 定时器不是 1us 或读的是高电平宽度           | 打印 rf_us, 短脉冲应约 400us, 长脉冲应约 1200us | 重新计算预分频; 确认下降沿清零、上升沿读取。           |
| 近距离能收, 远距离不稳    | 阈值太窄、天线差、电源噪声                 | 近距离 rf_us 稳定, 远距离波动大                | 适当放宽 RF_PULSE_MIN/MAX, 优化天线和电源。   |
| 长按只输出一次         | 重复码窗口或业务过滤过强                  | 观察 RF.active_10ms、发射器重复周期           | 按产品需求调整重复处理间隔。                    |
| 偶尔输出全 00 或 FF   | 干扰或起始位误判                      | 看是否集中出现在电机/继电器动作时                   | 保留全 0/全 FF 过滤, 增加电源滤波和二次校验。       |
| STM32 HAL 中断不回调 | CubeMX 未启用 EXTI IRQ 或 pin 不匹配 | 断点看 EXTI_IRQHandler 是否进入            | 启用 NVIC; 在回调里判断正确 GPIO_Pin。       |
| STM8 进入低功耗后不再接收 | 唤醒后时钟/TIM/EXTI 未恢复            | 唤醒后打印 TIM 频率或 LED 节拍                | 唤醒流程中重新初始化时钟和定时器。                 |
| 51 定时读数差 12 倍   | 12T/1T 机器周期理解错误               | 同一脉冲读数固定差 12 倍                      | 按具体芯片手册重新计算定时器 tick。              |

## 12. 常见问题排查

| 问题          | 优先检查                                    | 可能原因                    | 处理建议                          |
|-------------|---|-------------------------|-------------------------------|
| 串口无任何输出     | Type-C 是否是数据线; 串口号是否正确; 波特率是否 9600      | CH340 驱动/串口选择错误; 程序未运行  | 先看设备管理器/串口列表, 再复位板子。          |
| 下载失败        | SWDIO/SWCLK/VCC/GND 是否接对; Keil Debug 设置 | 下载器未识别; SWD 时钟过高; 芯片未供电 | 降低 SWD Clock, 确认目标电压。         |
| 按遥控器无 RF 输出 | 模块 VCC/GND/DAT/ANT; 频率是否 315/433.92 匹配  | 接线错; 没天线; 发射端频率不匹配      | 先近距离测试, 再用信号助手看信号。            |
| LED 闪但串口没码  | USART1 引脚复用、CH340、串口助手                  | RF 已收到但打印链路异常           | 看 p_led2 是否翻转, 确认 PF0/PF1 复用。 |
| 码值不稳定       | 天线、电源、距离、脉宽阈值                           | 信号弱或阈值太窄                | 近距离先稳定, 再逐步拉                  |



|            |                        |                         |                           |
|------------|------------------------|-------------------------|---------------------------|
|            |                        |                         | 远; 必要时调整 RF_PULSE_*。      |
| 距离很短       | 天线长度/位置、电源纹波、外壳金属      | 天线不匹配; DC-DC 噪声; 模块周围干扰 | 换天线、远离大电感、加滤波。            |
| 偶尔误触发      | 是否只用单帧判断; 是否过滤全 0/全 FF | 干扰脉冲被误判                 | 保留二次校验, 增加地址白名单和动作确认。     |
| 换 MCU 后不工作 | 定时器单位、EXTI 线、DAT 电平    | 移植接口不等效                 | 用示波器验证 DAT 波形和计数值。        |
| 5V 供电后异常   | R6/R7、电平兼容、MCU VCC     | 模块 DAT 电平超过 MCU 供电电压    | 保持 3.3V 或加电平转换, 避免 IO 过压。 |

## 13. 出厂给客户前检查清单

| 检查项    | 通过标准   |
|--------|--|
| 资料完整   | 规格书、原理图、Keil 工程、本文档一起交付。                               |
| 工程可编译  | 在指定 Keil/Pack 环境下 0 Error, 生成 HEX。                     |
| 硬件默认电压 | 确认 R6/R7 与说明一致, 默认 3.3V。                               |
| 串口输出   | 上电有提示, 按键有 RF:xx xx xx。                                |
| 天线测试   | 至少近距离稳定, 建议用信号助手验证天线效果。                                |
| 代码修正   | 交付客户前应确认 EXTI LINE1 source 已修正, 源码中文注释无乱码。             |
| 移植示例   | 若提供 51/STM8/STM32 版本, 应至少验证 DAT 中断、1us 定时器、串口输出和近距离收码。 |
| 边界     | 明确本例程不含学习码持久化、不含产品动作逻辑。                                |





| 开发助手   | 信号助手  |
|--|---|
|             |              |
| 不同点:<br>1、测数据值<br>2、有编码类型要求(1527 等)  | 不同点:<br>1、测信号强度<br>2、不限编码( ASK/OOK 调制 )   |
| 用途:<br>1、显示遥控器/发射模块的地址码和按键值<br>2、显示遥控频率、脉宽、编码类型<br>3、遥控产品批量测试                                | 用途:<br>1、显示遥控器/发射模块信号强弱(相对值)<br>2、对比天线好坏<br>3、遥控产品批量测试  |
| 供电:<br>TYPE-C  | 供电:<br>TYPE-C   |
| <br>微信扫码购买 | <br>微信扫码购买 |



## 注释

源码注释。rf.c 文件开头

```
/*  
 * 远系列接收模块 DAT 输入说明  
 * 硬件连接: 远-R1x DATA -> RFDAT1 -> PY32F002A PA1  
 * 输入模式: 浮空输入, 外部中断双边沿触发  
 * 计时方式: TIM1 每 1us 计数一次, 在 DAT 下降沿清零, 在上升沿读取低电平宽度  
 * 解码格式: 同步头 + 24bit 数据; 连续两帧一致后置 b_rx_on  
 * 注意事项: 修改系统主频后, 必须重新确认 TIM1 是否仍为 1us 计数  
 */
```

| 注释对象                    | 说明                          |
|-------------------------|-----------------------------|
| RF_SYN_MIN/MAX          | 单位为 us, 说明同步头范围。            |
| RF_PULSE_MIN/MAX/MID    | 单位为 us, 说明 0/1 判定逻辑。        |
| rf_dat_buf[3]           | 保存 24bit 解码结果。              |
| b_rx_on                 | 有效帧标志, 由中断置位、主循环清零。         |
| RF.active_10ms          | 用于重复处理间隔/活跃窗口。              |
| GPIO_rf_INT_en_or_dis() | DAT 接 PA1, EXTI 必须选择 LINE1。 |