无线模块编解码快速上手指南

推荐发射端：

远-T4A发射模块

￥1.86

推荐接收端：

远-R1A遥控接收模块

￥1.49

一、上图这种常规的 433Mhz 无线模块，发射端一般用 PT2262/EV1527/PT2240/HCS301 编码。如果选择 PT2262，接收端则直接搭配 2272，匹配好振荡电阻和烙好 A0-A7 脚就可以直接使用了。

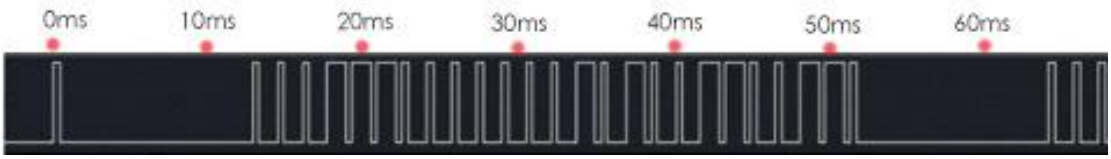
二、如果发射用 EV1527、PT2240、HCS301 等芯片，这些需要自已编写解码程序，单片机解码，要不断地处理 100us-20ms 脉冲识别，还要做别的任务，比如按键 LED 数码扫描等。

三、如果单片机的任务单一，可以用在 main 中循环查询端口的方式来不断地判断脉冲宽度，一般工程师都是用这种方式入手的，这种方式仅限 EMC153 这类低档 MCU，如果是 51 或 ST 不推荐，原因是单片机还是有其它任务，这些耗时较长，会直接造成无线接收的丢码，这种解码方式仅限入门用。

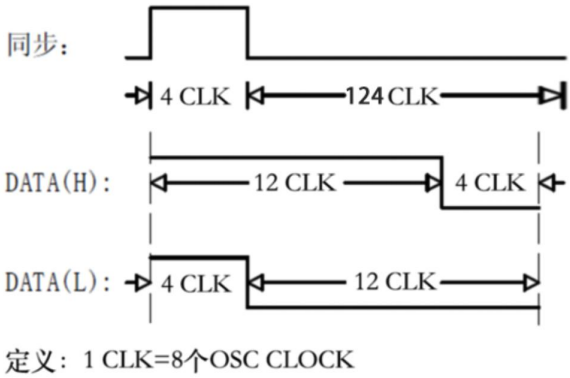
四、推荐解码方法是一个中断引脚加一个定时器，在中断里完成这部分的无线解码；虽然模块在静态时，有 30us—1000us 不等杂波，占用 MCU 太多的时间，这完全可以忽视；写好中断代码就好了，中断以外的时间处理其它任务完全没有问题。比如串口收发，SPI 通讯等。处理串口中断，和其它定时中断也基本不影响，如果串口波特率较高，实测有误码时，可以将串口中断的优先级别高于无线接收。

五、下面讨论具体解码方法，一般选择单片机具有上升沿/下降沿,双边沿中断的引脚来直接连接无线接收模块（老式 51 单片机仅有下降沿方式，后面讲此改进方法），注意不要灌电给接收模块，就是单片机 5V，模块选 3V 的，这样会改变接收模块的电压，应作处理，一般此脚设为上拉输入，但更建议将引脚设为高阻输入，这样减小接收模块的负载，有利其减小电压纹波。

六、如何解码？我们选观察下以下编码图，以 1527 编码为例说明：



上面图中间部分是一帧完整的数据，最有特征的就是最宽的部分，俗称同步脉冲，两个同步脉冲间的小脉冲是我们要解的编码。两个同步头间的高电平是 25bit，其中最后 1bit 是下一同步头的。我们只解其中的 24bit 即可。

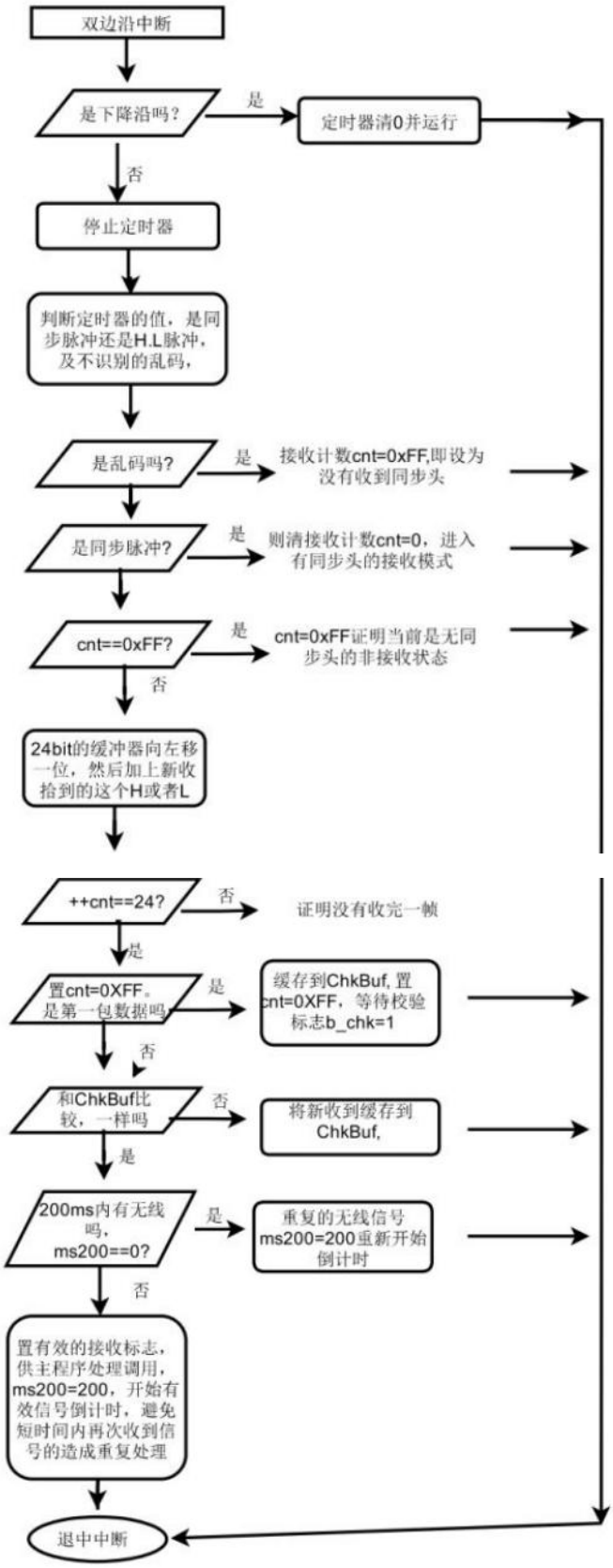


上面图的 CLK 是指振荡电阻的振荡周期，我们可以忽视，直接用示波器量下 4CLK 的宽度就好了，不用对照 DATASHEET 去找这个 CLK 到底是多少。请仔细观察上面的图，它的 H 和 L 周期是一致的，只是颠倒了位置.所以为了提高解码效率，我们虽是双边沿中断触发，但只检查低电平的持续时间是多少就行了，高电平直接无视,请看下附流程图。

上升沿时判断定时器的脉冲宽度，请注意，1527 这类编码必须要接收正确的两次信号才可以向外输出，才能证明数据收到了，这是许多新入门者常犯的错误，因为无线的乱码有可能会形成这种巧合的接收，请务必校验两次，否则产品批量出货后可能会有麻烦。

HCS301 的编码只需接收一次就行了，它一包 64bit 位中自带校验码，不需要我们这种重复两次的方法来校验。Chkbuf 中的数据最好也加个时间限制，超时后，校验位为清 0，下次最收到码时，又成为了第一包数据了。数据收到后，交由主程序中查询去处理，有可能还要有 E2 中的数据再次比较，证明是否有无对码等的操作。

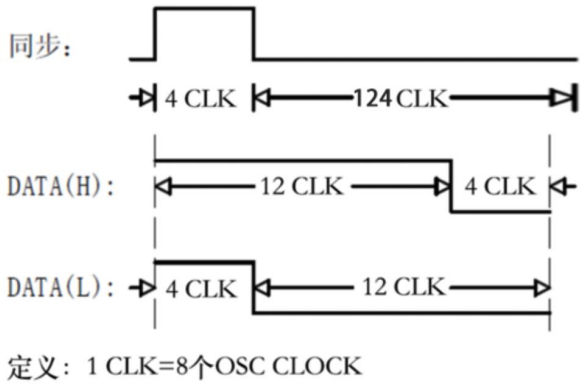
请参见下面的流程图：


双边沿中断
是下降沿吗?
是
定时器清0并运行
否
停止定时器
判断定时器的值，是同步脉冲还是H.L脉冲，及不识别的乱码，
是乱码吗?
是
接收计数cnt=0xFF,即设为没有收到同步头
是同步脉冲?
是
则清接收计数cnt=0，进入有同步头的接收模式
cnt==0xFF?
是
cnt=0xFF证明当前是无同步头的非接收状态
否
24bit的缓冲器向左移一位，然后加上新收拾到的这个H或者L
++cnt==24?
否
证明没有收完一帧
是
置cnt=0XFF。是第一包数据吗
是
缓存到ChkBuf, 置cnt=0XFF，等待校验标志b_chk=1
否
和ChkBuf比较，一样吗
否
将新收到缓存到ChkBuf,
是
200ms内有无线吗，ms200==0?
是
重复的无线信号ms200=200重新开始倒计时
否
置有效的接收标志，供主程序处理调用，ms200=200，开始有效信号倒计时，避免短时间内再次收到信号的造成重复处理
退中中断

七、EV1527 编码芯片之无线解码原理说明

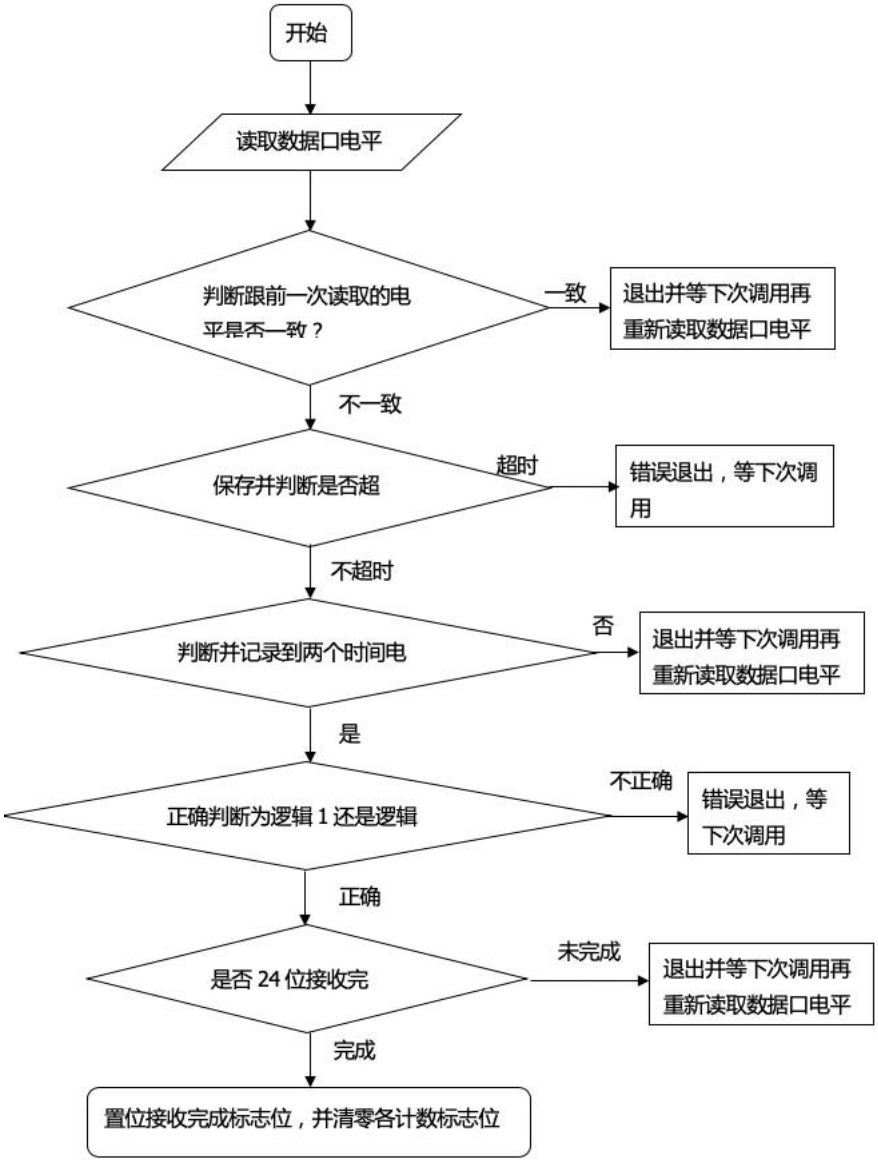
EV1527 是一片由 CMOS 设计制造的可预烧内码的学习码编码 IC，由软件解码，每次发 4 帧，每帧 24 位加一个同步码。此 24 位中，前 20 位为芯片内码（即 ID，共有 2 的 20 次方，即 1048576 组组合，大大降低使用上编码重复的几率），后 4 位是按键值（即数据）。

其编码图形如下所示：



其中逻辑 1 是一个长高电平加一个短低电平；逻辑 0 是一个短高电平加一个长低电平。同步码在 24 位编码后面。这样解码的时候只要检测高电平和低电平的持续时间就可以判断是逻辑 1 还是逻辑 0 了，因为它们都是以高电平开头的，另外末尾的一个同步码的第一个上升沿刚好可以作为第 24 位的低电平的持续时间的判断用。

因为 ASK 接收模块当没有接收到有效信号的时候数据脚会输出杂乱无章的波形，不便于用中断法解码，宜用查询法，其 EV1527 的解码原理如下流程图所示：其中需要用到定时器，定时器设为 50us 中断一次，一直开启的。



具体解码函数如下所示：

需要用到的变量：

```
extern unsigned char timer_4_count;
extern unsigned char timer_4_countover;
unsigned char in_bit = 0;
unsigned char rx_start = 0;
unsigned char rx_data_ok = 0;
unsigned char recvbit[4];
unsigned char recvbitcount = 0;
```

```
unsigned char recvbyte[40];
unsigned char recvbytecount = 0;
unsigned char Recv_data[5];
unsigned char in_bit_n = 0;
//////////////////接收函数//////////////////////
void Recieve()
{
  //一进来就先把引脚的状态读取了，然后判断跟前面的是否一样，不一样的时候才进行后续运算
  in_bit_n = inport; //inport 是 ASK 模块的数据脚
  if(in_bit == in_bit_n)
  {
    return;
  }
  in_bit = in_bit_n;
  //P3_7 = in_bit; //把值丢给 LED 口
  if(timer_4_countover)
  {//超时错误
RecieveError();
return;
  }
  // 接收 4 次电平变化，才能确定 1 bit
  if((timer_4_count > min_time_l)&&(timer_4_count < max_time_l))
  {//窄脉冲,4~14,就是 200us~700us
    if(in_bit) //高电平,现在为高电平，其实之前是低电平的
    {
recvbit[recvbitcount] = 0x00; //低短
    }
    else//低电平
    {
recvbit[recvbitcount] = 0x01; //高短
    }
  }
  else if((timer_4_count > min_time_h)&&(timer_4_count < max_time_h))
  {//宽脉冲，16~60，就是 800us~3000us
    if(in_bit)
    {
recvbit[recvbitcount] = 0x02; //低长
    }
    else
    {
recvbit[recvbitcount] = 0x03; //高长
    }
  }
```

```
    else
    {//出错
RecieveError();
return;
    }
    timer_4_count = 0;
    timer_4_countover = 0;
  // 1527
    recvbitcount++;
    if(recvbitcount < 2)
    {
return;
    }
    else
    {
        //这里判断的电平，应该是跟实际的相反的，因为只有电平变化了，才会做相应处理，不变化的话是直接退出的。
        if((recvbit[0] == 1)&&(recvbit[1] == 2))      //高短低长
        {
recvbyte[recvbytecount] = 0;
        }
        else if((recvbit[0] == 3)&&(recvbit[1] == 0))    //高长低短
        {
recvbyte[recvbytecount] = 1;
        }
        else
        {
RecieveError();
return;
        }
    }
    recvbytecount++; //接收到的字节数加 1。
    recvbitcount = 0; //
    if(recvbytecount < RECV_BIT_NUMBER)
    {//  未接收完
        return;
    }
    recvbytecount = 0;
    timer_4_count = 0;
    rx_data_ok = 1;
}
```

50us 定时器中断函数：

```
unsigned long Timer4Count = 0;
```

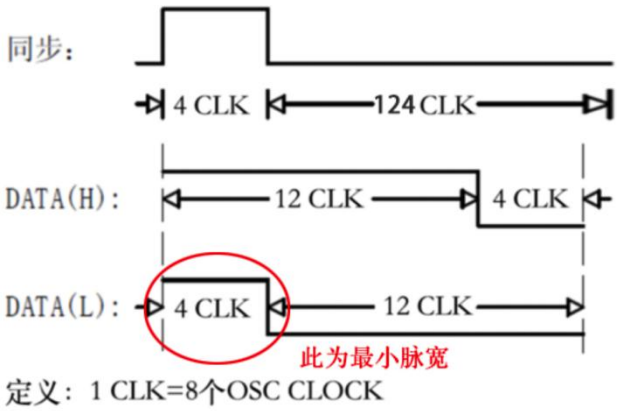
```
unsigned char timer_4_count = 0;
unsigned char timer_4_countover = 0;
#pragma vector = TIM4_OVR_UIF_vector
__interrupt __root void TIM4_UIF_HANDLER()
{
  static unsigned char tt4=0;
  //PA_ODR_ODR2=~PA_ODR_ODR2;
  tt4++;
  if(tt4>=20)
  {//1ms
    tt4=0;
    Timer4Count++;
    //PA_ODR_ODR3=~PA_ODR_ODR3;
  }
  timer_4_count++;
  if(timer_4_count == 0)
    timer_4_countover++;
  TIM4_SR_UIF=0;
}
```

程序功能：使用前需要对码，即按一下按键后松开，4 个 LED 全亮后熄灭只留 LED2，LED3 常亮，5s 钟内按下遥控器，可以发现 LED 分两组闪烁三次后熄灭即可完成对码。 以后该遥控器就可以工作。接收端收到无线信号后，先输出串口，然后判断 ID，ok 后点亮对应的 LED，100ms 后熄灭。

若想解其他最小脉宽的编码，只需修改程序中的如下四个变量即可：

```
#define max_time_h      60      //宽脉冲最大允许时间  data*TCC_time
#define min_time_h      16      //宽脉冲最小允许时间
#define max_time_l      14      //窄脉冲最大允许时间
#define min_time_l      4       //窄脉冲最小允许时间
```

注：所谓“最小脉宽”即 EV1527 的编码中四倍时钟，如下图所示：



特别提醒：

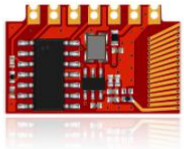
以上编解码方法对于刚接触无线模块的朋友来说，可能有一定门槛。强烈建议采用我司免开发模块，80%以上的客户都在选用，包括云米等。

免开发发射模块：

灵-T3A遥控模块

￥2.98

免开发接收模块：



灵-R1A接收模块

￥2.98

微信扫一扫

技术咨询+免费拿样品

微信扫一扫

产品购买+资料下载