

# 红外数据传输

## 一、 红外通信原理

红外遥控有发送和接收两个组成部分。发送端采用单片机将待发送的二进制信号编码调制为一系列的脉冲串信号，通过红外发射管发射红外信号。红外接收完成对红外信号的接收、放大、检波、整形，并解调出遥控编码脉冲。为了减少干扰,采用的是价格便宜性能可靠的一体化红外接收头(HS0038，它接收红外信号频率为 38kHz，周期约 26μ s) 接收红外信号，它同时对信号进行放大、检波、整形得到 TTL 电平的编码信号，再送给单片机，经单片机解码并执行去控制相关对象。如图 1 所示：

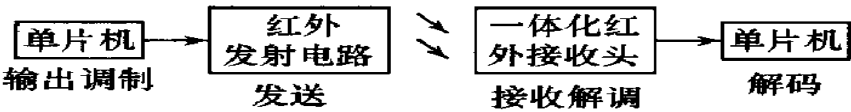


图 1

红外发送部分由 51 单片机、键盘、红外发光二极管和 7 段数码管组成。键盘用于输入指令， 51 单片机检测键盘上按键的状态，并对红外信号进行调制，发光二极管产生红外线，数码管用来显示发送的键值。图 2 红外发射电路

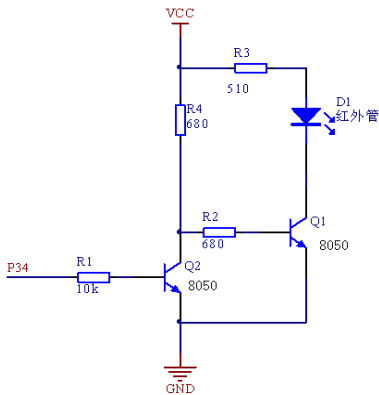


图 2 红外发射电路

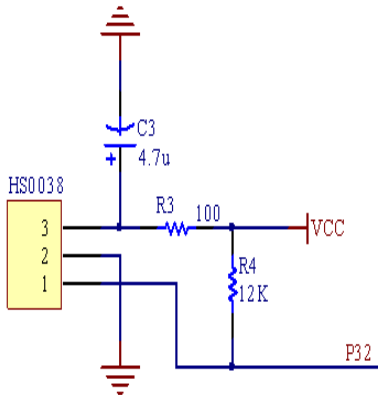


图 3 红外接收电路

红外接收部分由 51 单片机、一体化红外接收头 HS0038 和 7 段数码管组成。51 单片机检测 HS0038，并对 HS0038 接收到的数据解码，通过数码管显示接收到的键值。图 3 红外接收电路

## 二、 编码、解码

### (1) 二进制信号的调制

二进制信号的调制由单片机来完成，它把编码后的二进制信号调制成频率为 38kHz 的间断脉冲串，相当于用二进制信号的编码乘以频率为 38kHz 的脉冲信号得到的间断脉冲串，即是调制后用于红外发射二极管发送的信号如图 4 二进制码的调制所示。

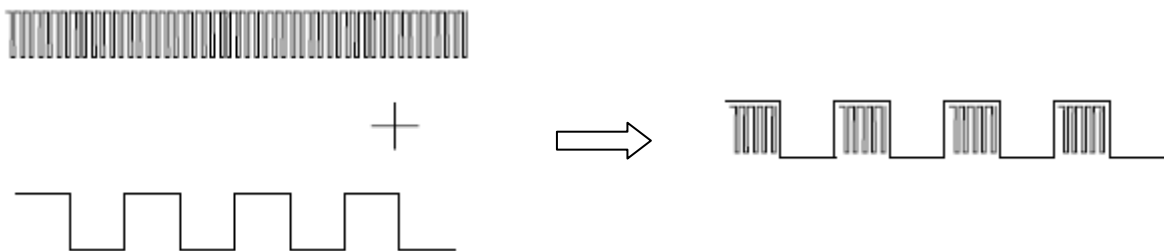


图 4 二进制码的调制

(2) 红外接收需先进行解调，解调的过程是通过红外接收管进行接收的。其基本工作过程为：当接收到调制信号时，输出高电平，否则输出为低电平，是调制的逆过程（图 5 解调）。HS0038 是一体化集成的红外接收器件，直接就可以输出解调后的高低电平信号；红外接收器 HS0038 的应用电路（图 6）。

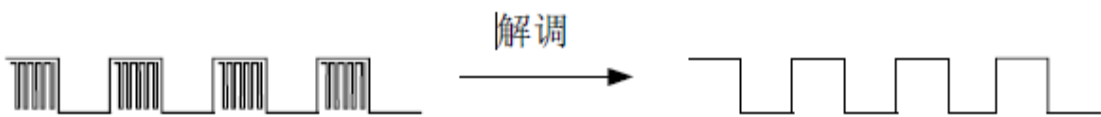


图 5 二进制码的解调

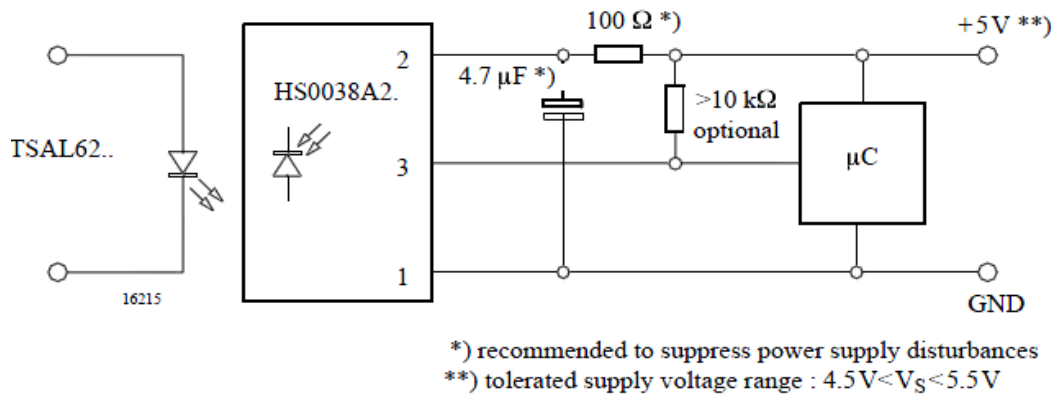


图 6 HS0038 的应用电路

(3) 红外遥控发射芯片采用 PPM 编码方式，当发射器按键按下后，将发射一组 108ms 的编码脉冲。遥控编码脉冲由前导码、16 位地址码（8 位地址码、8 位地址码的反码）和 16 位操作码（8 位操作码、8 位操作码的反码）组成。通过对用户码的检验，每个遥控器只能控制一个设备动作，这

样可以有效地防止多个设备之间的干扰。编码后面还要有编码的反码，用来检验编码接收的正确性，防止误操作，增强系统的可靠性。前导码是一个遥控码的起始部分，由一个 9ms 的高电平（起始码）和一个 4.5ms 的低电平（结果码）组成，作为接受数据的准备脉冲。以脉宽为 0.56ms、周期为 1.12ms 的组合表示二进制的“0”；以脉宽为 1.68ms、周期为 2.24ms 的组合表示二进制的“1”。

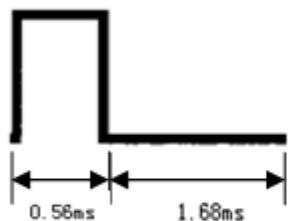


图 7 二进制码'1'

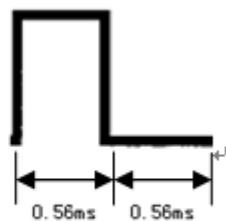


图 8 二进制码'0'

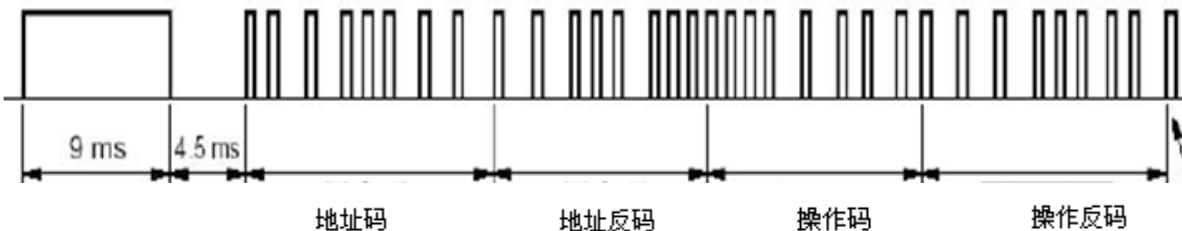


图 9 发送一组完整的编码脉冲

（4）单片机采用外部中断 INT0 管脚和红外接收头的信号线相连，中断方式为边沿触发方式。计算中断的间隔时间，来区分前导码、二进制的“1”、“0”码。并将 8 位操作码提取出来在数码管上显示。

红外接收头输出的原始遥控数据信号,正好和发射端倒向.也就是以前发射端原始信号是高电平,那接收头输出的就是低电平,反之.

三、 程序

（1）发送程序

```
#include <reg52.h>

static bit OP;          //红外发射管的亮灭
static unsigned int count;    //延时计数器
static unsigned int endcount; //终止延时计数
static unsigned int temp;    //按键
static unsigned char flag;    //红外发送标志
static unsigned char num;
sbit ir_in=P3^4;
char iraddr1; //十六位地址的第一个字节
char iraddr2; //十六位地址的第二个字节
unsigned char code table[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
```

```

0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e}; // 共阳数码管 0~f
void SendIRdata(char p_irdata);
void delay(unsigned int);
void keyscan();

/*****主函数*****/
void main(void)
{
    num=0;
    P2=0x3f;
    count = 0;
    flag = 0;
    OP = 0;
    ir_in= 0;

    EA = 1; // 允许CPU中断
    TMOD = 0x11; // 设定定时器0和1为16位模式1
    ET0 = 1; // 定时器0中断允许

    TH0 = 0xFF;
    TL0 = 0xE6; // 设定值0为38K 也就是每隔26us中断一次
    TR0 = 1; // 开始计数

    iraddr1=3; // 00000011
    iraddr2=252; // 11111100

    do{keyscan();
    }while(1);
}

/*****定时器0中断处理 *****/
void timeint(void) interrupt 1
{
    TH0=0xFF;
    TL0=0xE6; // 设定值为38K 也就是每隔26us中断一次
    count++;

    if (flag==1)
    {
        OP=~OP;
    }
    else
    {
        OP = 0;
    }
    ir_in= OP;
}

void SendIRdata(char p_irdata)
{

```

```

int i;
char irdata=p_irdata;

//发送9ms的起始码
endcount=223;
flag=1;
count=0;
do{}while(count<endcount);

/*****发送4.5ms的结果码*****/
endcount=117;
flag=0;
count=0;
do{}while(count<endcount);

/*****发送十六位地址的前八位*****/
irdata=iraddr1;
for(i=0;i<8;i++)
{

/*****先发送0.56ms的38KHZ红外波（即编码中0.56ms的低电平）*****/
endcount=10;
flag=1;
count=0;
do{}while(count<endcount);

/*****停止发送红外信号（即编码中的高电平）*****/
if(irdata-(irdata/2)*2) //判断二进制数个位为1还是0
{
endcount=41; //1为宽的高电平
}
else
{
endcount=15; //0为窄的高电平
}
flag=0;
count=0;
do{}while(count<endcount);

irdata=irdata>>1;
}

/*****发送十六位地址的后八位*****/
irdata=iraddr2;
for(i=0;i<8;i++)
{
endcount=10;
flag=1;
count=0;
do{}while(count<endcount);

if(irdata-(irdata/2)*2)
{
endcount=41;
}
else
{

```

```

endcount=15;
}
flag=0;
count=0;
do{}while(count<endcount);

irdata=irdata>>1;
}

/*****发送八位数据*****/
irdata=p_irdata;
for(i=0;i<8;i++)
{
endcount=10;
flag=1;
count=0;
do{}while(count<endcount);

if(irdata-(irdata/2)*2)
{
endcount=41;
}
else
{
endcount=15;
}
flag=0;
count=0;
do{}while(count<endcount);

irdata=irdata>>1;
}

/*****发送八位数据的反码*****/
irdata=~p_irdata;
for(i=0;i<8;i++)
{
endcount=10;
flag=1;
count=0;
do{}while(count<endcount);

if(irdata-(irdata/2)*2)
{
endcount=41;
}
else
{
endcount=15;
}
flag=0;
count=0;
do{}while(count<endcount);

irdata=irdata>>1;
}

```

```

endcount=10;
flag=1;
count=0;
do {}while (count<endcount);
flag=0;
}

void delay(unsigned int z)
{
    unsigned char x,y;
    for (x=z;x>0;x--)
    for (y=110;y>0;y--);
}

/*****4×4键盘扫描按下按键发射数据*****/
void keyscan()
{
    P1=0xfe;
    temp=P1;
    temp=temp&0xf0;
    while (temp!=0xf0)
    {
        temp=P1;
        switch (temp)
        {
            case 0xee:num=1;
            break;
            case 0xde:num=2;
            break;
            case 0xbe:num=3;
            break;
            case 0x7e:num=4;
            break;
        }
        while (temp!=0xf0)
        {
            temp=P1;
            temp=temp&0xf0;
        }
        P2=table[num-1];
        SendIRdata(table[num-1]);
    }
    P1=0xfd;
    temp=P1;
    temp=temp&0xf0;
    while (temp!=0xf0)
    {
        temp=P1;
        switch (temp)
        {
            case 0xed:num=5;
            break;
            case 0xdd:num=6;
            break;
            case 0xbd:num=7;
            break;
            case 0x7d:num=8;

```

```

break;
}
while (temp!=0xf0)
{
temp=P1;
temp=temp&0xf0;
}
P2=table[num-1];
SendIRdata(table[num-1]);
}
P1=0xfb;
temp=P1;
temp=temp&0xf0;
while (temp!=0xf0)
{
temp=P1;
switch (temp)
{
case 0xeb:num=9;
break;
case 0xdb:num=10;
break;
case 0xbb:num=11;
break;
case 0x7b:num=12;
break;
}
while (temp!=0xf0)
{
temp=P1;
temp=temp&0xf0;
}
P2=table[num-1];
SendIRdata(table[num-1]);
}
P1=0xf7;
temp=P1;
temp=temp&0xf0;
while (temp!=0xf0)
{
temp=P1;
switch (temp)
{
case 0xe7:num=13;
break;
case 0xd7:num=14;
break;
case 0xb7:num=15;
break;
case 0x77:num=16;
break;
}
while (temp!=0xf0)
{
temp=P1;
temp=temp&0xf0;
}
}

```



```
P2=table[num-1];
SendIRdata(table[num-1]);
}
}
```

## (2) 接收程序

```
#include "reg52.h"
#define uchar unsigned char
#define uint unsigned int

uchar dis_num, num, num1, num2, num3;
sbit led=P1^0;

unsigned char code table[]={
0xc0, 0xf9, 0xa4, 0xb0,
0x99, 0x92, 0x82, 0xf8,
0x80, 0x90, 0x88, 0x83,
0xc6, 0xa1, 0x86, 0x8e}; //共阳数码管 0~f

sbit prem =P3^2; //定义遥控头的接收脚

uchar ram[4]={0,0,0,0}; //存放接受到的4个数据 地址码16位+按键码8位+按键码取反的8位

void delaytime(uint time) //延迟90uS
{
    uchar a,b;
    for(a=time;a>0;a--)
    {
        for(b=40;b>0;b--);
    }
}

void rem() interrupt 0 //中断函数
{
    uchar ramc=0; //定义接收了4个字节的变量
    uchar count=0; //定义现在接收第几位变量
    uint i=0; //此处变量用来在下面配合连续监测9MS 内是否有高电平
    prem=1;

    for(i=0;i<1100;i++) //以下FOR语句执行时间为8MS左右
    {
        if(prem) //进入遥控接收程序首先进入引导码的前半部判断,即:是否有9MS左右的低电平
        return; //引导码错误则退出
    }

    while(prem!=1); //等待引导码的后半部 4.5 MS 高电平开始的到来。
    delaytime(50); //延时大于4.5MS时间,跨过引导码的后半部分,来到真正遥控数据32位中
    //第一位数据的0.56MS开始脉冲

    for(ramc=0;ramc<4;ramc++) //循环4次接收4个字节
    {
        for(count=0;count<8;count++) //循环8次接收8位(一个字节)
        {
            while(prem!=1); //开始判断现在接收到的数据是0或者1,首先在这行本句话时,
            //保已经进入数据的0.56MS 低电平阶段
            //等待本次接受数据的高电平的到来。
            delaytime(9); //高电平到来后,数据0 高电平最多延续0.56MS,而数据1,高电平可
```

```

        //延续1.66MS大于0.8MS 后我们可以再判断遥控接收脚的电平，
        if (prem) //如果这时高电平仍然在继续那么接收到的数据是1的编码
        {
            ram[ramc]=(ram[ramc]<<1)+1; //将目前接收到的数据位1放到对应的字节中
            delaytime(11); //如果本次接受到的数据是1，那么要继续延迟1MS，这样才能跨入
                           //下个位编码的低电平中（即是开始的0.56MS中）
        }
        else //否则目前接收到的是数据0的编码
            ram[ramc]=ram[ramc]<<1; //将目前接收到的数据位0放到对应的字节中
    } //本次接收结束，进行下次位接收，此接收动作进行32次，正好完成4个字节的接收
}

if (ram[2]!=(^(ram[3]&0x7f))) //本次接收码的判断
{
    for (i=0;i<4;i++) //没有此对应关系则表明接收失败，清除接受到的数据
        ram[i]=0;
    return ;
}

dis_num=ram[2]; //将接收到的按键数据赋给显示变量
}

main()
{
    IT0=1; //设定INT0为边沿触发
    EX0=1; //打开外部中断0
    EA=1; //全局中断开关打开
    while(1)
    {
        switch(dis_num)
        {
            case 0x81: num=0; break;
            case 0xcf: num=1; break;
            case 0x92: num=2; break;
            case 0x86: num=3; break;
            case 0xcc: num=4; break;
            case 0xa4: num=5; break;
            case 0xa0: num=6; break;
            case 0x8f: num=7; break;
            case 0x80: num=8; break;
            case 0x84: num=9; break;
            case 0x88: num=10; break;
            case 0xe0: num=11; break;
            case 0xb1: num=12; break;
            case 0xc2: num=13; break;
            case 0xb0: num=14; break;
            case 0xb8: num=15; break;
        }
        P2=table[num];
        P1=0x01;
        delaytime(5);
    }
}

```