使用 P87LPC762 单片机处理红外控制信号

宁波大学　张卫强

Philips 公司推出的 51LPC 系列单片机 P87LPC762 具有速度快，可靠性高的特点。在相同工作频率下，指令处理速度是标准 51 的两倍，内建的看门狗可以有效防止程序飞跑。该系列单片机的详细情况可到 http://www.zlgmcu.com/ 查询。这里详细介绍它在红外控制中的一个应用。

红外控制系统中的红外发送电路采用 NB9148，它是用作通用红外遥控发射器的 CMOS 大规模集成电路，与 NB9149 相配可完成 10 个功能控制，与 NB9150 相配可完成 18 个功能控制，可发射的指令达 75 个，其中 63 个是连续指令，可多键组合，12 个是单发指令，只能单键使用。

一．NB9148 简介

1．主要特点：

源电压范围较宽：2.2V-5.5V

MOS 工艺保证了极低的功耗

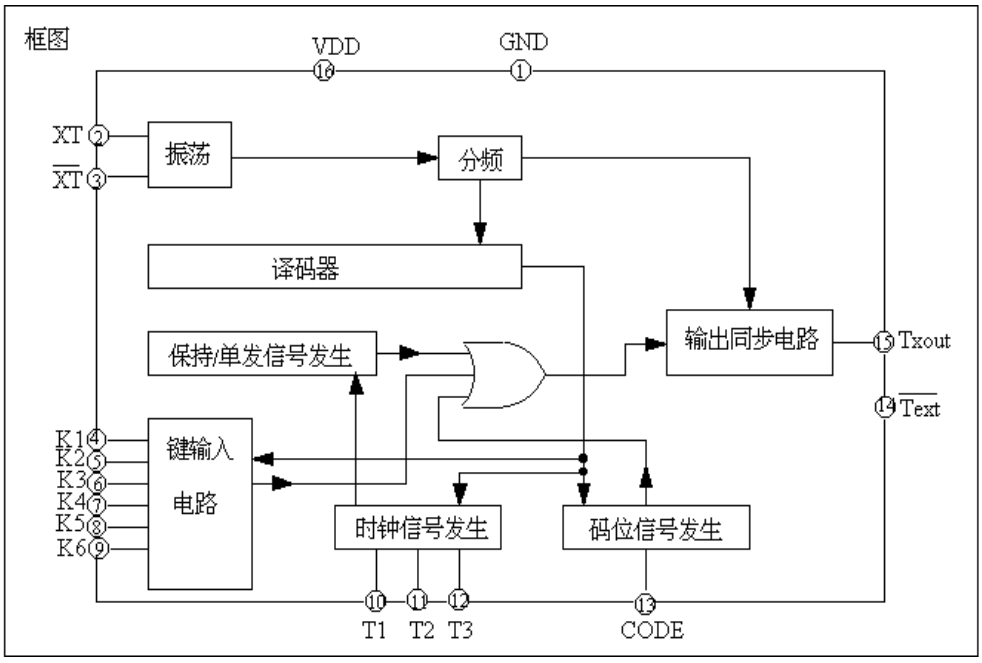
多键组合

围元件少

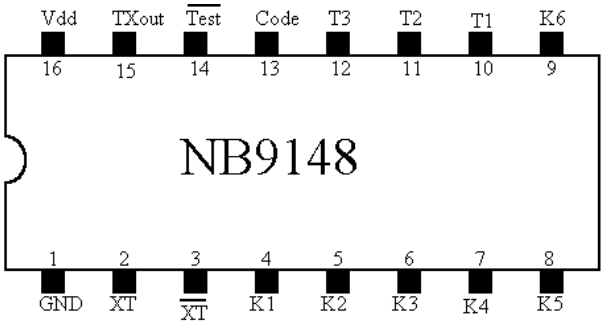
位与其它模式兼容

需外接 LC 或陶瓷振荡器即产生振荡

2．内部结构如下图所示：



3．电路外形图如下：



NB9148管脚图 DIP 16 PIN

4．极限参数表（Ta=25℃）

参数	符号	极限值	单位
电源电压	V_{DD}	6.0	V
输入/输出电压	V_{IN}	Vss-0.3 -- V_{DD}+0.3	V
功耗	P_D	200	mW
工作温度	T_{opr}	-20-75	℃
存储温度	T_{stg}	-55-125	℃
T_{XOUT}“1”输出电流	I_{OUT}	-5	mA

5．电参数表(V_{DD}=3V, Ta=25℃,另有说明外)

项目				符号	测试条件	最小	典型	最大	单位
电源电压				V_{DD}	所有功能操作	2.2	—	5.0	V
工作电流				I_{DD}	键通，无负载	—	—	1.0	mA
静态电流				I_{DS}	键开，不振荡	—	—	10	μA
输入端	K-K6CODE	输入电压	高电平	V_{IH}	—	2.0	—	3.0	V
			低电平	V_{IL}	—	0	—	0.5	V
	K1-K6	输入电流	高电平	I_{IH}	V_{IH}=3V	20	30	60	μA
			低电平	I_{IL}	V_{IL}=0V	-1.0	—	1.0	μA
	CODE TESTR	输入电流	高电平	I_{IH}	V_{IH}=3V	-1.0	—	1.0	μA
			低电平	I_{IL}	V_{IL}=0V	20	30	60	μA
输出端	T_1-T_3	输出电流	高电平	I_{OH}	V_{OH}=2V	—	—	-500	μA
			低电平	I_{OL}	V_{OL}=3V	50	—	—	μA
	T_{XOUT}	输出电流	高电平	I_{OH}	V_{OH}=2V	—	—	-0.1	mA
			低电平	I_{OL}	V_{OL}=2V	1.0	—	—	mA
振荡器反馈电阻				R_f	—	—	500	—	kΩ
振荡频率				f_{osc}	—	400	455	600	kHz

6．各管脚功能描述：

管脚号	符号	输入/输出	功能描述	
1，16	GND，Vdd		地/电源	提供电源
2，3	XT，NXT		振荡器	连接 455kHz 晶振等产生振荡，（内建反馈电阻）

4~9	K1-K6	I	键输入端	键矩阵键输入端。T1-T3×K1-K6 连成 18 键（内建下拉电阻）
10~12	T1-T3	0	时序信号输出端	键矩阵的数字时序信号输出
13	CODE	I	码位输入端	用作传输和接收的码位匹配用
14	NTEST	I	测试端	开路
15	T_{XOUT}	0	输出端	传输信号输出，信号 12 位一个周期，38kHz 载波调制

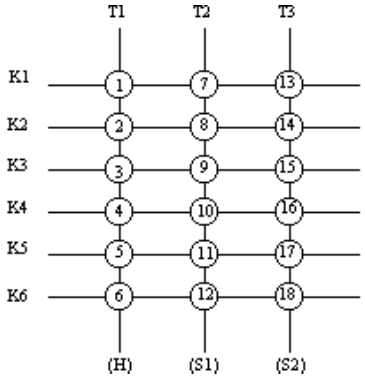
7．内部结构主要部分功能描述：

- 振荡电路

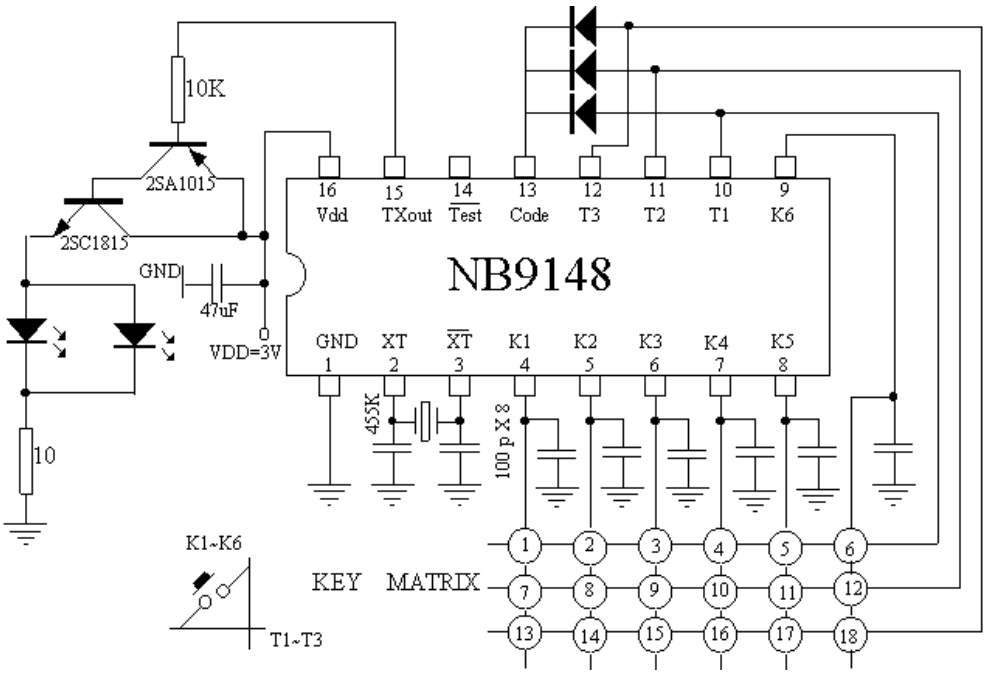
内含 CMOS 反相器及自偏置电阻。外接陶瓷振荡器或 LC 串联谐振回路即可组成振荡器。当振荡频率设定为 455kHz 时，则发射载波频率为 38kHz。只有当按键操作时才会产生振荡，以此降低功耗。

- 键输入

通过 K1~K6 输入和 T1~T3 的时序输出可连接 6×3 键盘矩阵，在 T1 这一列内的 6 个键（图中 1~6 号键）可以任意多键组合成 63 个状态，输出连续发射。处于 T_2 和 T_3 这两列的键（图中 7~18 号键）均只能单键使用，且每按一次只能发射一组控制脉冲，若一列上的数键同时按下，其优先次序为 K1、K2、K3、K4、K5、K6。在同一 K 线上的键无多键功能，若同时按下数键，其优先次序为 T_1、T_2、T_3。



8．典型应用线路如下：



9．发送命令格式

发送命令由 12 位码组成。其中 C1-C3 是用户码，用来确定不同的模式。每种组合有三个状态：01、10 和 11，而 00 状态

不用。H、S1 和 S2 是代表连续发送或单次发送的码，D1~D6 是发送的数据码。

C1	C2	C3	H	S1	S2	D1	D2	D3	D4	D5	D6
用户码			连发/单发码			键输入码					

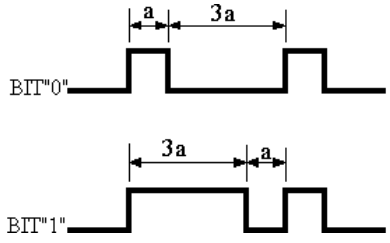
10．键与码的关系：

键号	数据									输出形式	键号	数据									输出形式
	H	S1	S2	D1	D2	D3	D4	D5	D6			H	S1	S2	D1	D2	D3	D4	D5	D6	
1	1	0	0	1	0	0	0	0	0	连续	10	0	1	0	0	0	0	1	0	0	单发
2	1	0	0	0	1	0	0	0	0	连续	11	0	1	0	0	0	0	0	1	0	单发
3	1	0	0	0	0	1	0	0	0	连续	12	0	1	0	0	0	0	0	0	1	单发
4	1	0	0	0	0	0	1	0	0	连续	13	0	0	1	1	0	0	0	0	0	单发
5	1	0	0	0	0	0	0	1	0	连续	14	0	0	1	0	1	0	0	0	0	单发
6	1	0	0	0	0	0	0	0	1	连续	15	0	0	1	0	0	1	0	0	0	单发
7	0	1	0	1	0	0	0	0	0	单发	16	0	0	1	0	0	0	1	0	0	单发
8	0	1	0	0	1	0	0	0	0	单发	17	0	0	1	0	0	0	0	1	0	单发
9	0	1	0	0	0	1	0	0	0	单发	18	0	0	1	0	0	0	0	0	1	单发

11．发送波形

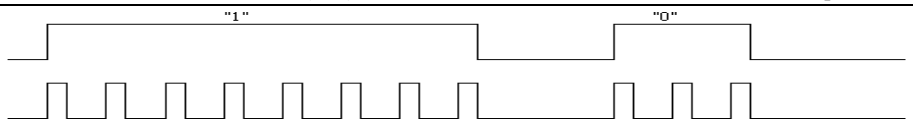
(1)“0”与“1”的识别

正脉冲的占空比为 1/4 时，代表“0”，正脉冲的占空比为 3/4 时，代表“1”。



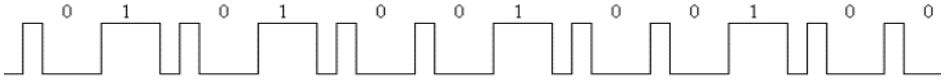
(2) 载波

无论是“0”还是“1”，它们被发射时，正脉冲是被调制在 38kHz（振荡频率为 455kHz 时）的载波上，载波的占空比为 1/3，这样有利于减小功耗。



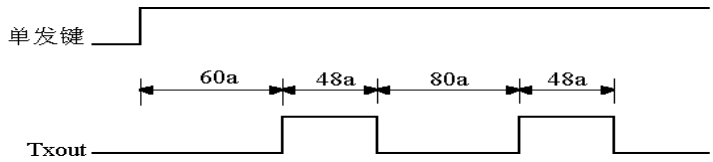
(3) 基本发送波形

每个发送周期按 C_1、C_2、C_3、H、S_1、S_2、D_1、D_2、D_3、D_4、D_5、D_6 的次序串行发送，总长度为 48a，其中 a 等于每个码周期的 1/4，其计算方法是 a=（$1/f_{osc}$）×192 秒。



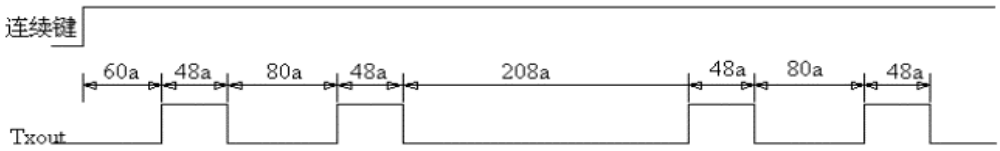
(4) 单发信号

凡是按下单发键时，输出码只发送两个周期。



(5) 连续信号

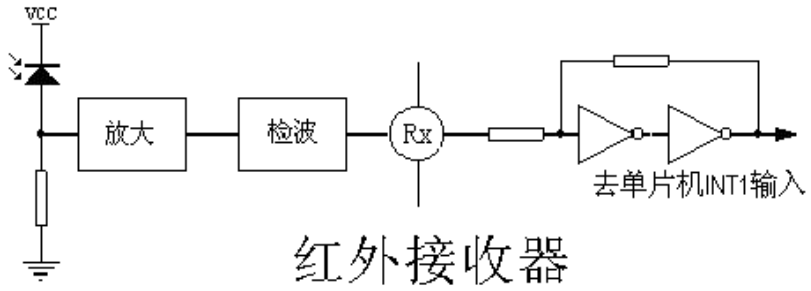
凡是按下连续键时，输出码将连续发送，在每两组信号之间停顿 208a



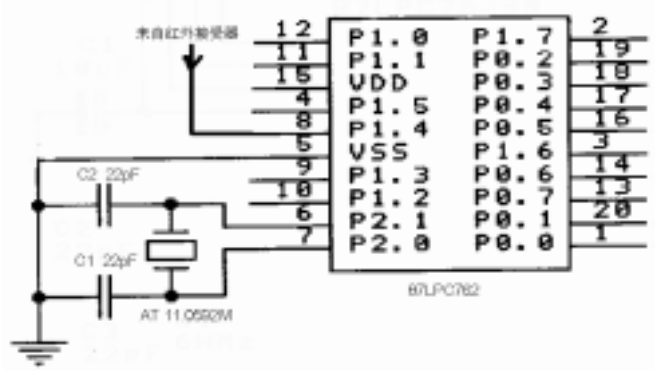
二、接收处理电路

控制系统不采用与其配套的接收电路，采用通用的红外接受器接受到调制后的方波脉冲序列，然后由 P87LPC762 处理并进行相应的控制。这种方案与采用配套电路的方案相比，具有成本低，处理灵活多变，实时控制性能佳等的诸多优点。

P87LPC762 的电源采用 5V，由一个红外接受器接受到由 NB9148 发出的经过检波去除 38kHz 载波后的高电平为 5V 的方波信号。下面是一个三端通用红外接受器的内部线路示意图，内部包含对接收信号进行整形的斯密特触发器。



输出信号送到单片机的 INT1 输入口，由 P87LPC762 解码并实施相应的操作：



接受信号编码的判断根据 9148 的编码规则。从编码表中可以看出，接受到的 12 位编码中，最后 6 位只有一个 1，每个循环组成一个编码组，这样每组有 6 个指令码，第一组 1—6 是连续发送的编码，7—12 是第一组单发编码，13—18 则是第二组单发编码。具体实施控制的其它外围电路可以根据大家的需要自己添加。下面介绍 P87LPC762 中的 C51 处理程序，可在 Keil C51 6.0 以上版本中编译，晶振采用 11.0592Mc，工作在每个机器周期 6 时钟的快速状态。

```
#include <REG768.H>                 /*Philips 87LPC768 寄存器定义头文件*/

#define REDINT      0x06            /*红外线间隔*/

    /*存放消息标志的可位寻址字节 Message,Message=NULL 时无任消息*/
unsigned char bdata Message;
    /*在接收过程中置位,检测标置位 Get 可以有效防止其它进程干扰接收,防止数据丢失.*/
    sbit RedMsg=Message^0;          /*红外遥控消息*/
    sbit RedRead=Message^1;         /*位接收过程标志*/
    sbit RedBit=Message^2;          /*接收到的位值*/
    sbit RSend =Message^3;          /*红外接收中重新发送标志*/

    /*存放遥控的字数据,低 4 位存放接收到的 bit 的位移,高 12 位从低到高存放接收到的 bit*/
unsigned char bdata RedDataL,RedDataH;
    /*RedDataH 字节低 6 位代表指令，只有一个 1，C1-C3 是用户码，H 表示连续，S1、S2 分别表示第一、二组单发*/
    sbit RedData0=RedDataL^4;       /*C1*/
    sbit RedData1=RedDataL^5;       /*C2*/
    sbit RedData2=RedDataL^6;       /*C3*/
    sbit RedData3=RedDataL^7;       /*H*/
    sbit RedData11=RedDataH^0;      /*D6*/
    sbit RedData10=RedDataH^1;      /*D5*/
    sbit RedData9=RedDataH^2;       /*D4*/
    sbit RedData8=RedDataH^3;       /*D3*/
    sbit RedData7=RedDataH^4;       /*D2*/
    sbit RedData6=RedDataH^5;       /*D1*/
    sbit RedData5=RedDataH^6;       /*S2*/
    sbit RedData4=RedDataH^7;       /*S1*/

unsigned char bdata State;          /*状态字节*/
    sbit RedControl=State^6;        /*遥控状态*/

    /*定时器 T00 的高位定时参数为 Timer，定时 256*Timer+(80--336)个周期共 139Timer+(43--182)us<37ms 定时器 T01 的高位定时节参数为 nTimer，定时 256*nTimer+(24--280)个周期 139nTimer+(13--152)us<9.1s,RedCon 存放红外接收时的载波计数*/
unsigned char data RedCon,Timer;
unsigned int nTimer;                /*定时整型参数*/

void main()
{
    IEN0=0x14;                      /*只打开 INT1 中断*/
    WDRST=0x1E;                     /*看门狗清 0*/
```

```
    WDRST=0xE1;
    WDCON=0x12;                 /*40-90ms 看门狗(>最大延时 37ms)*/
    TCON=0x40;                  /*定时器 1 开始工作, INT1 低电平触发*/
    TMOD=0x23;                  /*定时器 0 扩展成两个 8 位定时器 T00 和 T01 用于同步控制*/
    if((WDCON&0x30)!=0x30){     /*看门狗、陷阱复位时无需初始化*/
        Message=0;              /*无消息*/
        State=0;                /*正常复位无任何状态*/
    }

while(1){                       /*消息循环*/
        WDRST=0x1E;             /*看门狗清 0*/
        WDRST=0xE1;
        EX1= RedControl;        /*设置遥控中断 INT1 */
        if(RedMsg) {            /*执行遥控指令*/
            EX1=0;              /*在指令没有处理完之前不能重复中断*/
            switch(RedDataH){   /*这里加入红外指令的控制过程*/
                case 0x82:      /*Channel 1*/
                    break;
                case 0xA0:      /*Channel 2*/
                    break;
                /*……*/
                            }
            EX1=1;
            RedDataL=0;         /*复位红外数据*/
            RedDataH=0;         /*复位红外数据*/
            RedMsg=0;}          /*复位红外遥控消息*/
                }
        }

void Count0(void)   interrupt 1    using 3
{                               /*定时器 T00 中断,最大定时 37ms*/
    if(Timer!=0){               /*检测定时器 T00 的扩展高位*/
        Timer--;
        return;}
}

    /*INT1 用于红外解码状态，遥控解码数据处理,nTimer=1 定时 152--291us*/
void Inte1() interrupt 2    using 2
{
    for(nTimer=8;nTimer>1;nTimer--);/*使处理周期达到 51 机器周期=27.7us 使得 RedCon<32*/
    if(RedRead) RedCon++;       /*0 信号宽度 a=420us,1 信号宽度 a=1260us,周期 4a=1680us*/
    else{                       /*开始计数或者重新发送时开始计数*/
        RedBit=0;               /*复位接收位*/
        RedCon=0;               /*复位载波计数*/
        RedRead=1;              /*置位位接收标志*/
        if(!ET1){               /*首次接收时没有启动定时器 T01，接收第一个位*/
```

```
                TF1=0;              /*复位定时器 T01 益出标志*/
                ET1=1;              /*启动 T01 定时*/
                RedDataL=0;         /*复位红外数据*/
                RedDataH=0;         /*复位红外数据*/
                RSend=0;}           /*复位重新发送标志*/
            }
}

void Count1(void)  interrupt 3  using 3
{                                   /*定时器 T01 中断,最大定时 9.1s*/
    if(nTimer!=0){                  /*检测定时器 T01 的扩展高位*/
        nTimer--;
        return;}
    ET1=0;                          /*关闭 T01 定时*/
    if(RedRead){                    /*红外接收状态*/
        if((RedDataL&0xF)==12){     /*第一阶段接收已经结束*/
            RSend=1;                /*置位重新发送标志以便校验*/
            RedDataL&=0xF0;}        /*复位位指针以便校验*/
        if(RedCon>27-REDINT&&RedCon<27+REDINT) RedBit=1;
        else RedBit=0;              /*低电平计数 9 表示 0，27 表示 1*/
        if(RSend){                  /*检验重复发送的数据是否与第一次符合*/
            switch(RedDataL&0xF){
                case 0:             /*检验重复发送的第 1 位数据*/
                    if(RedBit!=RedData0) goto RClear;
                    break;
                case 1:             /*检验重复发送的第 2 位数据*/
                    if(RedBit!=RedData1) goto RClear;
                    break;
                case 2:             /*检验重复发送的第 3 位数据*/
                    if(RedBit!=RedData2) goto RClear;
                    break;
                case 3:             /*检验重复发送的第 4 位数据*/
                    if(RedBit!=RedData3) goto RClear;
                    break;
                case 4:             /*检验重复发送的第 5 位数据*/
                    if(RedBit!=RedData4) goto RClear;
                    break;
                case 5:             /*检验重复发送的第 6 位数据*/
                    if(RedBit!=RedData5) goto RClear;
                    break;
                case 6:             /*检验重复发送的第 7 位数据*/
                    if(RedBit!=RedData6) goto RClear;
                    break;
                case 7:             /*检验重复发送的第 8 位数据*/
                    if(RedBit!=RedData7) goto RClear;
                    break;
```

```
            case 8:             /*检验重复发送的第 9 位数据*/
                if(RedBit!=RedData8) goto RClear;
                break;
            case 9:             /*检验重复发送的第 10 位数据*/
                if(RedBit!=RedData9) goto RClear;
                break;
            case 10:            /*检验重复发送的第 11 位数据*/
                if(RedBit!=RedData10) goto RClear;
                break;
            case 11:            /*检验重复发送的第 12 位数据*/
                if(RedBit!=RedData11) goto RClear;
                RedMsg=1;       /*接受到经过检验正确的编码后，置位遥控消息*/
                RedBit=0;       /*复位接收位*/
                RSend=0;        /*复位重新发送标志*/
                RedRead=0;      /*复位接收过程标志*/
                RedCon=0;       /*复位载波计数*/
                return;
            default:            /*重复发送的数据多于 12 位时判断为错误*/
                goto RClear;
            }
        }
    else{
        switch(RedDataL&0xF){
            case 0:             /*保存首次发送的第 1 位数据*/
                RedData0=RedBit;
                break;
            case 1:             /*保存首次发送的第 2 位数据*/
                RedData1=RedBit;
                break;
            case 2:             /*保存首次发送的第 3 位数据*/
                RedData2=RedBit;
                break;
            case 3:             /*保存首次发送的第 4 位数据*/
                RedData3=RedBit;
                break;
            case 4:             /*保存首次发送的第 5 位数据*/
                RedData4=RedBit;
                break;
            case 5:             /*保存首次发送的第 6 位数据*/
                RedData5=RedBit;
                break;
            case 6:             /*保存首次发送的第 7 位数据*/
                RedData6=RedBit;
                break;
            case 7:             /*保存首次发送的第 8 位数据*/
                RedData7=RedBit;
```

```
                    break;
                case 8:                 /*保存首次发送的第 9 位数据*/
                    RedData8=RedBit;
                    break;
                case 9:                 /*保存首次发送的第 10 位数据*/
                    RedData9=RedBit;
                    break;
                case 10:                /*保存首次发送的第 11 位数据*/
                    RedData10=RedBit;
                    break;
                case 11:                /*保存首次发送的第 12 位数据*/
                    RedData11=RedBit;
                    break;
                default:                /*首次发送的数据多于 12 位时判断为错误*/
                    goto RClear;
                }
            }
        RedDataL++;                 /*位位移加 1*/
        RedBit=0;                   /*复位接收位*/
        RedRead=0;                  /*复位接收过程标志*/
        RedCon=0;                   /*复位载波计数*/
        nTimer=423;                 /*用定时 140a 检测同步信号 208a*/
        TF1=0;                      /*复位定时器 T01 益出标志*/
        ET1=1;}                     /*启动定时器*/
    else if(RSend){                 /*在位接收没有结束时发生定时中断需要复位接收信息(同步)*/
    RClear:
    RedDataL=0;                     /*复位红外数据*/
    RedDataH=0;                     /*复位红外数据*/
    RedBit=0;                       /*复位接收位*/
    RedRead=0;                      /*复位接收过程标志*/
    RSend=0;                        /*复位重新发送标志*/
    RedCon=0;                       /*复位载波计数*/
    ET1=0;                          /*关闭 T01 定时*/
    }
}
```