AVR 学习笔记十八、红外遥控键值解码实验

18.1 实例功能

红外线遥控是目前使用最广泛的一种通信和遥控手段。由于红外线遥控装置具有体积小、功耗低、功能强、成本低等特点，因而，继彩电、录像机之后，在录音机、音响设备、空调机以及玩具等其它小型电器装置上也纷纷采用红外线遥控。工业设备中，在高压、辐射、有毒气体、粉尘等环境下，采用红外线遥控不仅完全可靠而且能有效地隔离电气干扰。

在这个实验中，我们采用红外线遥控器和一体化红外接收头来进行红外遥控键值解码的实验，本实例分为三个功能模块，分别描述如下：

- 单片机系统：利用 ATmega16 单片机与一体化红外接收器组成红外接收电路。
- 外围电路：红外接收电路、串口电平转换电路。
- 软件程序：编写软件，实现接收并识别红外遥控器按键键值的程序。

通过本实例的学习，掌握以下内容：

- 红外遥控接收的电路设计程序实现。

18.2 器件和原理

18.2.1 红外遥控的基本知识

通用红外遥控系统由发射和接收两大部分组成。应用编/解码专用集成电路芯片来进行控制操作，如图 1 所示。发射部分包括键盘矩阵、编码调制、LED 红外发送器；接收部分包括光、电转换放大器、解调、解码电路。

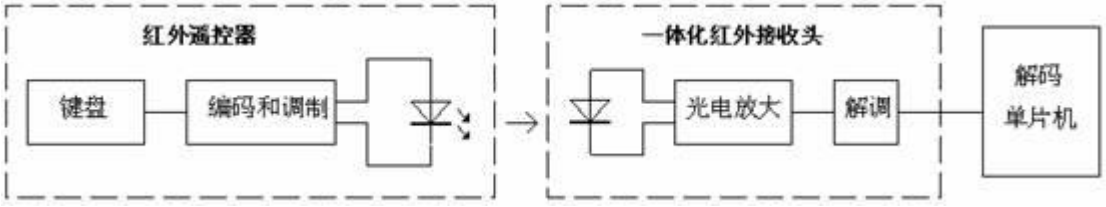

图 1 红外线遥控系统框图

18.2.2 遥控发射器及其编码

遥控发射器专用芯片很多，根据编码格式可以分成两大类，这里我们以运用比较广泛，解码比较容易的一类来加以说明，现以日本 NEC 的 uPD6121G 组成发射电路为例说明编码原理（一般家庭用的 DVD、VCD、音响都使用这种编码方式）。当发射器按键按下后，即有遥控码发出，所按的键不同遥控编码也不同。这种遥控码具有以下特征：

采用脉宽调制的串行码，以脉宽为 0.565ms、间隔 0.56ms、周期为 1.125ms 的组合表示二进制的“0”；以脉宽为 0.565ms、间隔 1.685ms、周期为 2.25ms 的组合表示二进制的“1”，其波形如图 2 所示。

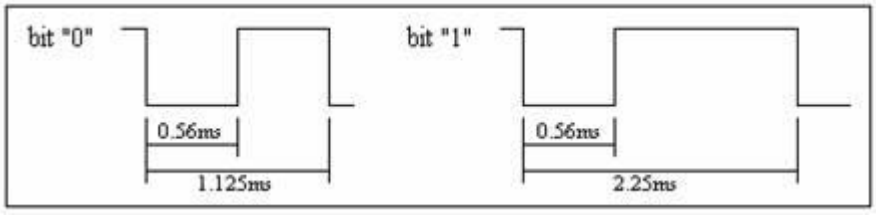

图 2 遥控码的“0”和“1”（注：所有波形为接收端的与发射相反）

上述“0”和“1”组成的 32 位二进制码经 38kHz 的载频进行二次调制以提高发射效率，达到降低电源功耗的目的。然后再通过红外发射二极管产生红外线向空间发射，如图 3 所示。

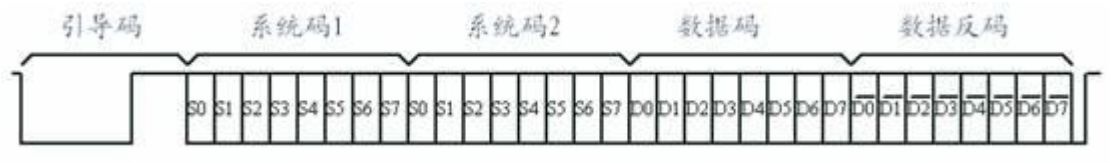

图 3 遥控信号编码波形图

UPD6121G 产生的遥控编码是连续的 32 位二进制码组，其中前 16 位为用户识别码，能区别不同的电器设备，防止不同机种遥控码互相干扰。该芯片的用户识别码固定为十六进制 01H；后 16 位为 8 位操作码（功能码）及其反码。UPD6121G 最多额 128 种不同组合的编码。

遥控器在按键按下后，周期性地发出同一种 32 位二进制码，周期约为 108ms。一组码本身的持续时间随它包含的二进制“0”和“1”的个数不同而不同，大约在 45～63ms 之间，图 4 为发射波形图。

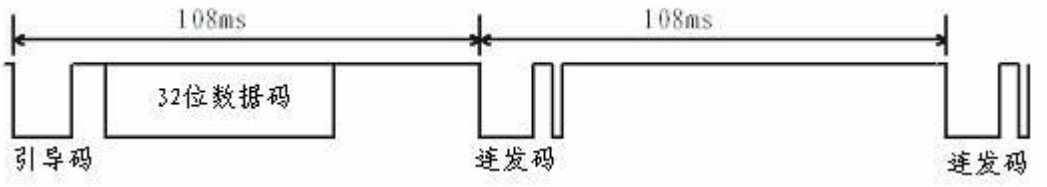

图 4 遥控连发信号波形

当一个键按下超过 36ms，振荡器使芯片激活，将发射一组 108ms 的编码脉冲,这 108ms 发射代码由一个引导码（9ms),一个结果码（4.5ms),低 8 位地址码（9ms~18ms),高 8 位地址码（9ms~18ms）,8 位数据码（9ms~18ms）和这 8 位数据的反码（9ms~18ms）组成。如果键按下超过 108ms 仍未松开，接下来发射的代码（连发码）将仅由起始码（9ms）和结束码（2.25ms）组成。

图 5 引导码

图 6 连发码

18.2.3 遥控信号接收

接收电路可以使用一种集红外线接收和放大于一体的一体化红外线接收器，不需要任何外接元件，就能完成从红外线接收到输出与 TTL 电平信号兼容的所有工作，而体积和普通的塑封三极管大小一样，它适合于各种红外线遥控和红外线数据传输。

接收器对外只有 3 个引脚：Out、GND、VCC 与单片机接口非常方便，如图 7 所示。

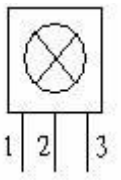

图 7 一体化红外接收器

① 脉冲信号输出接，直接接单片机的 IO 口。

② GND 接系统的地线（0V）；

③ VCC 接系统的电源正极（+5V）；

18.3 电路和连接

本实验主要有两部分电路模块组成：串口电平转换电路，一体化红外接收器电路。串口电平转换电路在前面的实例中我们已经做过介绍，在此不再重复。这里我们重点介绍一下一体化红外接收器的电路。根据上面的叙述，我们知道，用单片机于一体化红外接收器的连接只需要一根线，在此我们将一体化红外线接收器的脉冲信号输出端（引脚 1）连接到单片机的外部中断引脚 INT0(即 PD2 口)。而一体化接收器的电源和地分别连接到学习板的 VCC 和 GND 处。

18.4 程序设计

1、程序功能

在本实例中，我们利用串口将单片机从一体化红外接收器接收到的红外遥控键值发送到计算机上，通过计算机的串口助手观察接收到的数据。

编程过程中，我们利用单片机的外部中断 0 口进行检测，一旦检测到有红外遥控信号出现，则程序进入外部中断处理程序，在处理数据过程中关闭外部中断，直到接收完数据，再将外部中断打开。

2、函数说明

本实例主要有串口数据发送程序和外部中断处理红外接收程序，串口程序我们前面例子中已经介绍过，本实例的程序中不再详细说明。

红外遥控的数据接收主要在外部中断函数中进行处理：处理过程为：当有遥控键值发送的时候，红外一体化接收器的脉冲信号输出脚发生一个下降沿的电平变化，外部中断采用下降沿出发的方式接收到由外部中断事件发生，程序进入外部中断处理函数，首先关闭外部中断，然后根据一体化接收器脉冲信号输出引脚的高低电平变化时间判断红外遥控发送的数据，共有 4 个字节的数据，处理完这 4 个字节数据后，利用单片机的串口将数据发送到计算机。

3、编程说明

使用 WINAVR 开发环境，使用的是外部 12M 的晶振，所以需要将 makefile 文件中的时钟频率修改为 12M。另外在程序烧录到单片机的时候，熔丝位也要选择为外部 12M 晶振（**注意是晶振，不是外部振荡器**，一定不要选择错了，否则会导致单片机不能再烧写程序）。

4、程序代码

关于串口的收发函数，在此不再列出，直接打包到程序文件夹中。

下面列出主程序以及红外接收程序。

```
/*********************************************
****          AVR   红外接收范例                ***
**** MCU: ATmega16                                  ***
****  作者：     maweili                             ***
****  编译器：WINAVR                              ***
****                                                  ***
****       2009.3.27                ***
*********************************************/

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>     //中断函数头文件
#include "usart.h"

void INT_Init(void);     //外部中断初始化
void Delayus(unsigned int lus);              //us 延时函数
void Delayms(unsigned int lms);              //ms 延时函数


int main(void)
{

      Port_Init();
      Usart_Init();
      INT_Init();

      sei();               //使能全局中断

      while(1)
      {

      }
}

//外部中断初始化
void INT_Init(void)
{
      MCUCR |= (1 << ISC01);     //选择外部中断 0，下降沿触发中断
      GICR |= (1 << INT0);       //使能外部中断 0
      GIFR |= (1 << INTF0);       //清除 INT0 中断标志位

}

//
```

```
ISR(INT0_vect)
{
    unsigned char i,j,k = 0,addr[4] = {0};

    GICR = 0x00;      //禁止外部中?    关闭外部中断，开始接受数据
    for(i = 0;i < 14;i++)
    {
        Delayus(400);
        if(PIND & (1 << PD2))      //9MS 内有高电平，则判断为干扰，退出处理程序
        {
            GICR |= (1 << INT0);      //使能外部中断 4
            return;
        }
    }
    while(!(PIND & (1 << PD2)));   //等待 9ms 低电平过去
    for(i = 0;i < 4;i++)                    //
    {
        for(j = 0;j < 8;j++)                //
        {
            while(PIND & (1 << PD2)); //等待 4.5ms 高电平过去
            while(!(PIND & (1 << PD2))); //等待变高电平
            while(PIND & (1 << PD2))   //计算高电平时间
            {
                Delayus(100);
                k++;
                if(k >= 30)                 //高电平时间过长，则退出处理程序
                {
                    GICR |= (1 << INT0);      //使能外部中断
                    return;                 //
                }

            }
            addr[i] = addr[i] >> 1;           //接受一位数据
            if(k >= 8)
            {
                addr[i] = addr[i] | 0x80;     //高电平时间大于 0.56，则为数据 1
            }
            k = 0;                          //计时清零
        }
    }

    Usart_PutChar(addr[0]);             //通过串口发送接收到的 4 个字节
    Usart_PutChar(addr[1]);
    Usart_PutChar(addr[2]);
```

```
    Usart_PutChar(addr[3]);

    GICR |= (1 << INT0);      //使能外部中断
}

//
//us 级别的延时函数
void Delayus(unsigned int lus)
{
    while(lus--)
    {
        _delay_loop_2(3);          //_delay_loop_2(1)是延时 4 个时钟周期，参数为 3 则延时
12
                        //个时钟周期，本实验用 12M 晶体，则 12 个时钟周期为 12/12=1us
    }
}

//ms 级别的延时函数
void Delayms(unsigned int lms)
{
    while(lms--)
    {
        _delay_loop_2(3000);              //延时 1ms
    }
}
```