

**ARM处理器及其应用**

**基础实验报告**

课题名称：SPWM与DDFS

院系： 电子信息与通信学院

专业： 电子信息工程

班级： 电信1204班

姓名： 黄衎

学号： U201213468

指导老师： 肖看

# 实验要求

基本要求：采用SPWM，实现DDFS功能，输出低频正弦波；采用板载按键可以调节输出波形频率；采用板载按键调制输出波形种类（三角波、方波、锯齿波等）。

扩展要求：采用通用红外遥控器能够设置输出信号的频率与波形；采用手机设置输出信号的频率与波形；其它功能。

# 实现情况

基本要求已经全部实现，在波形失真较小情况下，可以输出10Hz~10kHz的正弦波，10Hz~1kHz的方波、锯齿波、三角波。两个板载按键可以调节频率大小，按住左边的按键，则频率持续增大，按住右边的按键，则频率持续减小。

由于按键用来设定频率，所以无法用按键设定波形。在扩展要求中，实现了红外遥控功能。可以用通用红外遥控器设定波形，以及输入数字和确定键来设定频率。

想要实现手机红外遥控，但是调试时间有限，所以没有做，比较可惜。

# 实验原理

## PWM波

PWM是一种对模拟信号电平进行数字编码的方法。通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。

SPWM(Sinusoidal PWM）法是一种比较成熟的，目前使用较广泛的PWM法.前面提到的采样控制理论中的一个重要结论：冲量相等而形状不同的窄脉冲加在具有惯性的环节上时，其效果基本相同.SPWM法就是以该结论为理论基础，用脉冲宽度按正弦规律变化而和正弦波等效的PWM波形即SPWM波形控制逆变电路中开关器件的通断，使其输出的脉冲电压的面积与所希望输出的正弦波在相应区间内的面积相等，通过改变调制波的频率和幅值则可调节逆变电路输出电压的频率和幅值.

## 使用单片机生成

市场上使用的很多单片机都有生成SPWM控制波形的功能，该生成波形外接驱动电路即可驱动功率桥，达到逆变的目的。应该说，只要具有PWM模块和定时器模块的单片机都可以完成此任务。

具体实现即首先将正弦表赋值给数组。然后PWM波形发生模块每个PWM周期进入中断，在ISR中按照正弦表更改PWM比较器的值，依次循环即可。

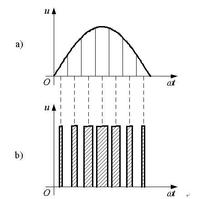


图 1 PWM（脉宽调制）示意图

## 红外解码

红外遥控的编码目前广泛使用的是： NEC Protocol 的 PWM(脉冲宽度调制)和 Philips

RC-5 Protocol 的 PPM(脉冲位置调制)。我的遥控器使用的是 NEC 协议，其特征如下：

1、 8 位地址和 8 位指令长度；

2、地址和命令 2 次传输（确保可靠性）

3、 PWM 脉冲位置调制，以发射红外载波的占空比代表“0”和“ 1”；

4、载波频率为 38Khz；

5、位时间为 1.125ms 或 2.25ms；

NEC 码的位定义：一个脉冲对应 560us 的连续载波，一个逻辑 1 传输需要 2.25ms（560us脉冲+1680us 低电平），一个逻辑 0 的传输需要 1.125ms（560us 脉冲+560us 低电平）。而遥控接收头在收到脉冲的时候为低电平，在没有脉冲的时候为高电平，这样，我们在接收头端收到的信号为：逻辑 1 应该是 560us 低+1680us 高，逻辑 0 应该是 560us 低+560us 高。

NEC 遥控指令的数据格式为：同步码头、地址码、地址反码、控制码、控制反码。同步码由一个 9ms 的低电平和一个 4.5ms 的高电平组成，地址码、地址反码、控制码、控制反码均是8 位数据格式。按照低位在前，高位在后的顺序发送。采用反码是为了增加传输的可靠性（可用于校验）。

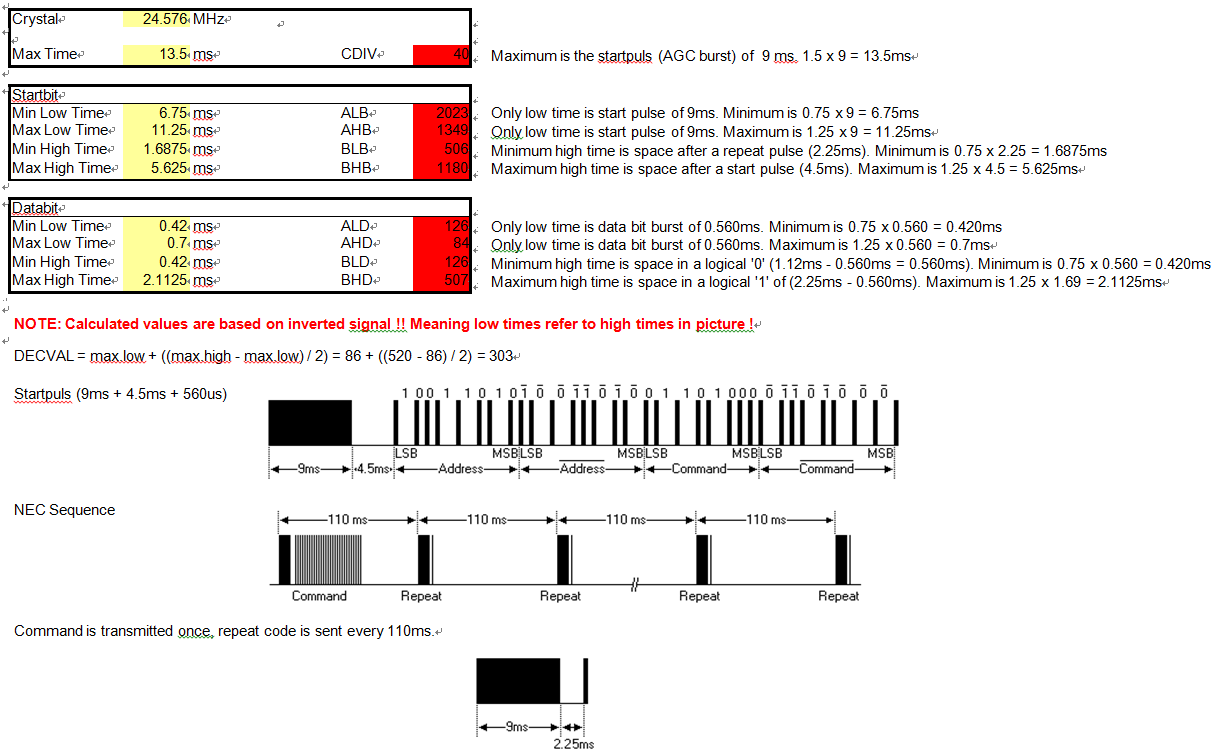


图 2 红外NEC协议

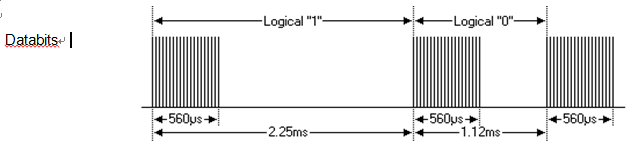


图 3 NEC协议中的一个bit位

一串红外命令（NEC命令）由9ms（560us\*16）的高电平引导、4.5ms（560us\*8）的低电平引导、32位命令组成。32位命令由8位的用户码（又叫地址）、8位用户码反码、8位命令码、8位命令码反码组成。先发LSB，后发MSB。

如果按键按着不动，则红外遥控器则会发送Repeat信号，间隔110ms，它由9ms高电平、2.25ms低电平、和560ms的高电平确认组成。

注意，发送信号经过调制、发送、解调接收后，得到的是反相的信号，即发送高电平则收到低电平信号、发送低电平收到高电平信号。

# 代码实现/关键代码解释

在视频中我简要说明了自己的思路，包括查表法实现占空比改变、状态机对红外命令解码、状态机实现遥控器类似菜单的功能。在这里对关键的代码进行说明。

## 定时查表法实现PWM

### PWM采样频率设定

#define PWM\_FREQ 40000

通过宏定义确定采样频率。

由于只能通过API设定周期，故我们设定周期为SysCtlClockGet()/PWM\_FREQ。

void APPinit()

{

//初始化DDFS相关数据

ddfsAPPState.waveChoose=2;

ddfsAPPState.freq=1000;

ddfsAPPState.pwmPeroid = SysCtlClockGet()/PWM\_FREQ;//50000000/20000=2500;

ddfsAPPState.pwmWidth = 1024;

ddfsAPPState.tablePointer = 0;

ddfsAPPState.ddfsDiff = 0;

ddfsAPPState.accPhase = 0;

//初始化PWM发生器

PWM\_init();

uart0\_init();

//初始化按键

ButtonsInit();SW\_init();

}

### 通过累加器查表

相位间隔的计算公式：

由于采样表中有1000个值，ddfsDiff=1000\*freq/PWM\_FREQ

注意把ddfsDiff和accPhase设定为float类型。

void PWMIntHandler()

{

//清除定时器中断

PWMGenIntClear(PWM1\_BASE, PWM\_GEN\_1, PWM\_INT\_CNT\_ZERO);

// GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x00);

ddfsAPPState.accPhase+=ddfsAPPState.ddfsDiff;

while(ddfsAPPState.accPhase>=CYCLE\_NUM) ddfsAPPState.accPhase-=CYCLE\_NUM;//循环

ddfsAPPState.tablePointer =(int)ddfsAPPState.accPhase;

ddfsAPPState.pwmNum = waveTable[ddfsAPPState.waveChoose][ddfsAPPState.tablePointer];

ddfsAPPState.pwmWidth = (int)ddfsAPPState.pwmNum \* MAX\_DUTY / 1024.0;

PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_3, ddfsAPPState.pwmWidth);

}

## 红外命令解码状态机



图 4 红外命令解码状态机

代码：

NECMachineStatue **StateMachine**(NECMachineStatue stateNow, uint8\_t bit){

**switch**(stateNow){

**case**(*LeadZeros*):

**if**(0==bit) **return** *LeadZeros*;

**else** **if**(1==bit){ ui8LeadOnesCounter++; **return** *LeadOnes*;}

**break**;

**case**(*LeadOnes*):

**if**(1==bit){ ui8LeadOnesCounter++; **return** *LeadOnes*; }

**else** **if**(0==bit)

**if**(ui8LeadOnesCounter>5) **return** *WaitOne*;

**else** **if**(ui8LeadOnesCounter<=5) **return** *RepeatSignal*;

**break**;

**case**(*WaitOne*):

**if**(0==bit) **return** *WaitOne*;

**else** **if**(1==bit) { ui8CountOnesCounter++; **return** *CountOnes*; }

**break**;

**case**(*CountOnes*):

**if**(1==bit) {

ui8CountOnesCounter++;

//高电平超过3个，则判断接收状态

**if**(ui8CountOnesCounter>=4)

**if**(32==ui8EncodedBitCounter) **return** *TransmitSuccess*;

**else** **return** *TransmitError*;

**return** *CountOnes*;

}

//收到低电平确认信号

**else** **if**(0==bit){

//收到0，1~2个高电平为0

**if**( (ui8CountOnesCounter>=1)&&(ui8CountOnesCounter<=2) ) NECReceiveBit(0);

//收到1，3~4个高电平为1

**else** **if**( (ui8CountOnesCounter>=3)&&(ui8CountOnesCounter<=4) ) NECReceiveBit(1);

//等待下次接收

**return** *WaitOne*;

}

**break**;

**default**:

**return** *LeadZeros*;

}

**return** *LeadZeros*;

}

## 红外命令响应状态机



图 5 红外命令响应状态机

状态机代码：

NECCommand **NECCommandMenu**(NECCommand necCommandMenuStatus, NECCommand necCommand)

{

uint8\_t comToNum = (uint8\_t)necCommand;

**switch**(necCommandMenuStatus){

**case**(*ChannelMinus*):

// if( (Repeat==necCommand)||(ChannelMinus==necCommand) ){//Repeat或者相同

**if**(*ChannelMinus*==necCommand){

**if**(ddfsAPPState.waveChoose>0) ddfsAPPState.waveChoose--;

**return** *ChannelMinus*;

}

**else**

**return** *WaitCommand*;

**case**(*ChannelPlus*):

// if( (Repeat==necCommand)||(ChannelPlus==necCommand) ){//Repeat或者相同

**if**(*ChannelPlus* == necCommand){

**if**(ddfsAPPState.waveChoose<3) ddfsAPPState.waveChoose++;

**return** *ChannelPlus*;

}

**else**

**return** *WaitCommand*;

//数字态

**case**(*Num0*):

**case**(*Num1*):

**case**(*Num2*):

**case**(*Num3*):

**case**(*Num4*):

**case**(*Num5*):

**case**(*Num6*):

**case**(*Num7*):

**case**(*Num8*):

**case**(*Num9*):

//不考虑Repeat，防止误操作

**if**( (0<=comToNum)&&(comToNum<=9) ){

freqToSet = 10\*freqToSet + comToNum;

seg\_display\_num(freqToSet);

//返回数字态

**return** *Num9*;

}

**else** **if**(*Equal*==necCommand){

ddfsAPPState.freq = freqToSet;

freqToSet = 0;

seg\_display\_num(ddfsAPPState.freq);

//返回等待态

**return** *WaitCommand*;

}

**else**

//Repeat 不响应

**return** *Num9*;

**default**: **return** *WaitCommand*;

}

}

NECCommand **NECCommandExecute**(NECCommand necCommand)

{

//首次执行命令

**switch**(necCommand){

**case**(*ChannelMinus*):

**if**(ddfsAPPState.waveChoose>0) ddfsAPPState.waveChoose--;

**return** *ChannelMinus*;

**case**(*ChannelPlus*):

**if**(ddfsAPPState.waveChoose<3) ddfsAPPState.waveChoose++;

**return** *ChannelPlus*;

**case**(*Num0*):

**case**(*Num1*):

**case**(*Num2*):

**case**(*Num3*):

**case**(*Num4*):

**case**(*Num5*):

**case**(*Num6*):

**case**(*Num7*):

**case**(*Num8*):

**case**(*Num9*):

//首次执行则归零

freqToSet=0;

freqToSet=10\*freqToSet+(int32\_t)necCommand;

seg\_display\_num(freqToSet);

**return** *Num9*;

**default**: **return** *WaitCommand*;

}

}

中断处理代码，包括了对解码状态机的调用和对命令响应状态机的调用：

**void** **NECTimerIntHandler**()

{

uint8\_t bit;

NECCommand necCommand;

**TimerIntClear**(*NEC\_TIMER*, *TIMER\_TIMA\_TIMEOUT*);

//读使能打开

**if**(ui8TransmitEnable)

{ //调试，读取560us间隔码

//读取PE0口的状态

ui8RawBitCounter++;

bit = **GPIOPinRead**(*INFRA\_RED\_PORT*, *INFRA\_RED\_PIN*);

**if**(bit) bit=1; **else** bit=0;

**if**(ui8RawBitCounter<=64){

ui64InfraRawCode[0]<<=1;

ui64InfraRawCode[0]+=bit;

}

**else**{

ui64InfraRawCode[1]<<=1;

ui64InfraRawCode[1]+=bit;

}

**if**(ui8RawBitCounter==*MAX\_RAW\_BITS*)

{

ui8TransmitEnable=0;

ui8RawBitCounter=0;

}

//进入状态机，NEC解码

machineState = StateMachine(machineState, bit);

// if( (TransmitSuccess==machineState) || (OverfullLowLevel==machineState) || (NotEnoughHighLevel==machineState) || (OverfullLeadingOnes==machineState) || (RepeatSignal==machineState)){

**if**( (*TransmitSuccess*==machineState) || (*RepeatSignal*==machineState) ||(*TransmitError*==machineState) ){

//关闭三色LED

**GPIOPinWrite**(*GPIO\_PORTF\_BASE*,*GPIO\_PIN\_1* | *GPIO\_PIN\_2* | *GPIO\_PIN\_3*,0x00);

//传输结束，保留数据

NECReceiveReset(*true*);

**if**(*TransmitSuccess*==machineState){

//响应相应的命令

necCommand = NECCommandExtract(ui32NECEncoded);

//命令没有问题

**if**( (necCommand!=*AddressError*)&&(necCommand!=*ValidateError*)&&(necCommand!=*UnknownCommand*) ){

//首次进入命令

**if**(*WaitCommand*==necCommandMenuStatus){

necCommandMenuStatus = NECCommandExecute(necCommand);

}

**else**{

necCommandMenuStatus = NECCommandMenu(necCommandMenuStatus, necCommand);

//如果返回WaitCommand则说明是不同的命令，需要再次执行

**if**(*WaitCommand*==necCommandMenuStatus)

necCommandMenuStatus = NECCommandMenu(necCommandMenuStatus, necCommand);

}

}

}

**else** **if**(*RepeatSignal*==machineState)

necCommandMenuStatus = NECCommandMenu(necCommandMenuStatus, *Repeat*);

}

}

# 参考文献

[1] 红外遥控器的原理，<http://www.cnblogs.com/heiyue/p/3478439.html>