# 实验3.8 ADC0808信号采集实验

姓名：孟麟芝 学号：201800121050 实验时间：2020.6.25

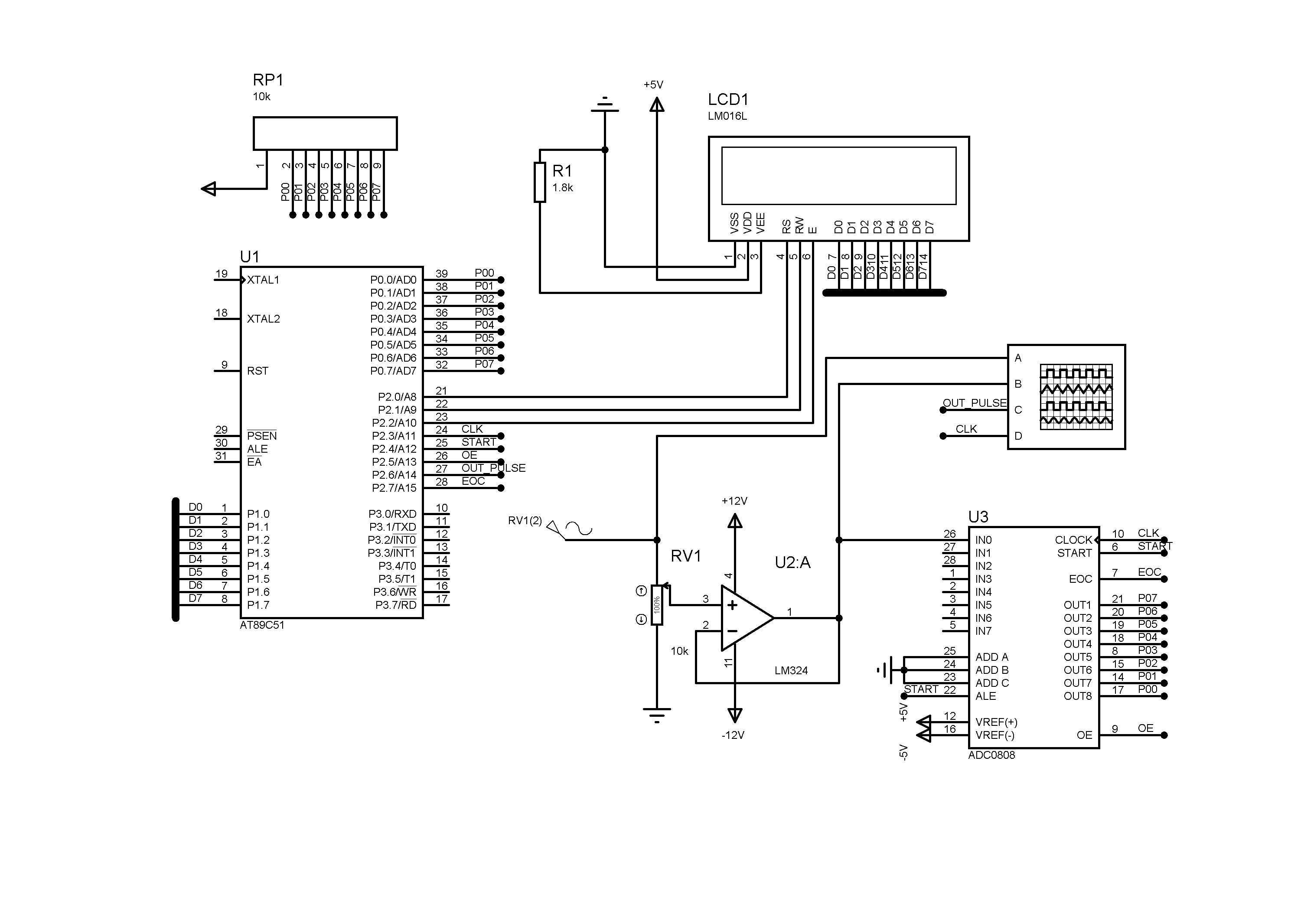
## 实验要求

（1）编写源程序并进行注释，叙述实验原理，画出流程图

（2）记录实验过程

（3）记录Proteus仿真结果

## 实验电路及功能



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **选件编号** | **元件名称** | **参数** | **所在元件库类名** | **子类名** | **生产厂家** |
| U1 | AT89C51 |  | Microprocessor ICs | 8051 Family | ATMEL |
| 微处理器 | 8051 家族 |
| LCD1 | LM016L |  | Optoelectronic 光电器件 | Alpha Numeric LCDs |  |
| 字符数字 LCD |
| U3 | ADC0808 |  | Data Converters | A/D Converters |  |
| 数据转换器 | A/D 转换器 |
| U2 | LM324 |  | Operational Amplifiers | Quad |  |
| 运算放大器 | 四 |
|  |  |  |  |  |  |
| RP1 | RESPACK- | 10k- | Resistors | Resistor Packs |  |
| 8 | 8 | 电阻 | 排阻 |
| RV1 | POT-HG | 10k | Resistors | Variable |  |
| 电阻 | 可变的 |
| R1 | RES | 1.8k | Resistors | Generic |  |
| 电阻 | 一般的 |

实验功能为利用LCD1602和AD0808实现简单的交流信号过零检测与频率分析。要求信号幅度变化时(满量程的5%-95%) ,不影响检测的结果。频率检测的结果通过LCD1602的第一行显示出来，信号过零时，能够通过P2.6输出一个脉冲宽度为5us的脉冲信号。

## 实验原理

LCD1602的控制方法按3.7节所示方法进行，ADC0808的控制方法按3.8.1所示方法进行。这里主要是过零点的检测方法如何实现。不能采用判断所采集到的数据是否为0的方法来实现，因为你的采集时刻不一定能够严格对准过零时刻。但是，我们注意到在0点的两边信号的极性是发生变化的，我们可以利用这一特点来实现过零检测。正弦波每个周期有两个过零点，因此，1s内过零次数除以2就是信号的频率。

因此，在程序中可以这样实现。当每次采集到一个新的数据之后都要看一下这个数据是正数还是负数。当这个数大于128时是正数，当它小于128时是负数。判断当前数据的正负极性和上一个数据的正负极性是否一致，如果不一致，则说明经过了一次过零点，将其记录入次数计数器。

ADC0808的CLK仍然用定时器T1来实现，可以将其设置为50kHz（硬件实现时可以更高，软件仿真再高将难以实现）。利用定时器T0实现50ms定时，并配合软件实现1s钟定时。采用12M晶振时，T0采用方式1，则处置应为（TH0=0x3C，TL0=0xB0）。

但是，由于中断处理函数需要一定的响应时间，因此这个参数只是理论计算结果，要根据实测情况稍作调整。

同样T1理论计算值和实际输出值可能也会有一定的差距，也要进行调整。

## 实验过程

（1）根据上述实验内容，参考1.2.2，在Proteus环境下建立图3.22所示原理图。

（2）根据实验原理画出流程图，并编写源程序。

（3）运行Keil5 C51开发环境，按照1.1.3节介绍的方法建立工程。

（4）按照1.2.2第（6）节介绍的方法将C语言源程序加入工程ADC0808\_self.uV2，并设置工程属性，将其晶振频率设置为12MHz，选择输出可执行文件，仿真方式为选择硬仿真，并选择其中的“PROTEUS VSM MONITOR 51 DRIVER”仿真器。

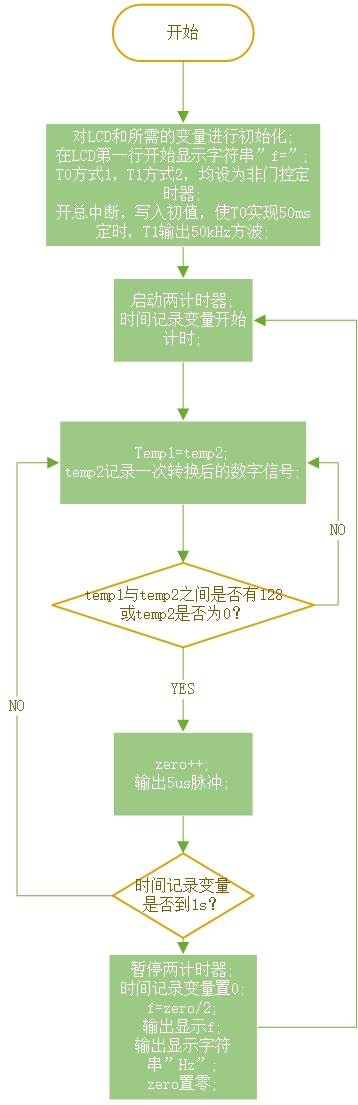
（5）构造（Build）工程。如果输入有误进行修改，直至构造正确，生成可执行程序为止。

（6）为AT89C51 设置可执行程序。

（7）运行程序，观察计算结果，并验证其是否正确。

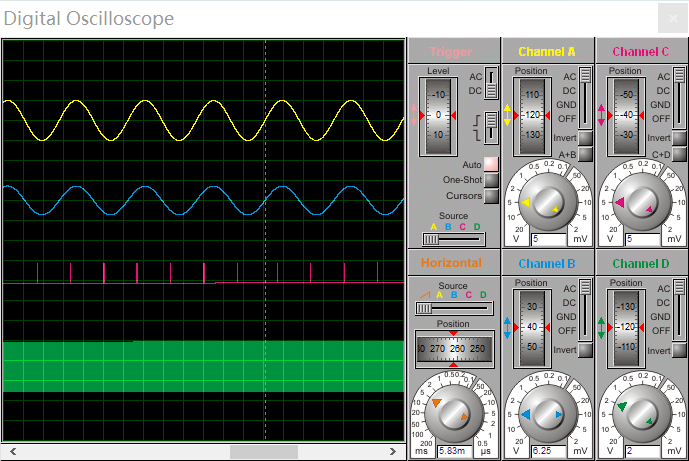
（8）改变RV1的抽头位置，从而改变输入信号的幅值，观察计算结果是否正确。

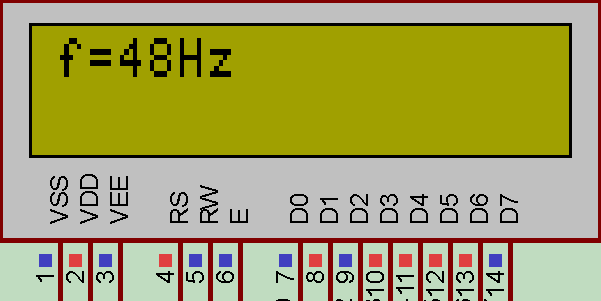
（9）更改信号发射器的频率，再次验证其功能是否正确。（注意：因为是软仿真，所以信号采集的速度受到限制，因此所输入的交流信号频率也不能太高，可以在200Hz以内尝试）。



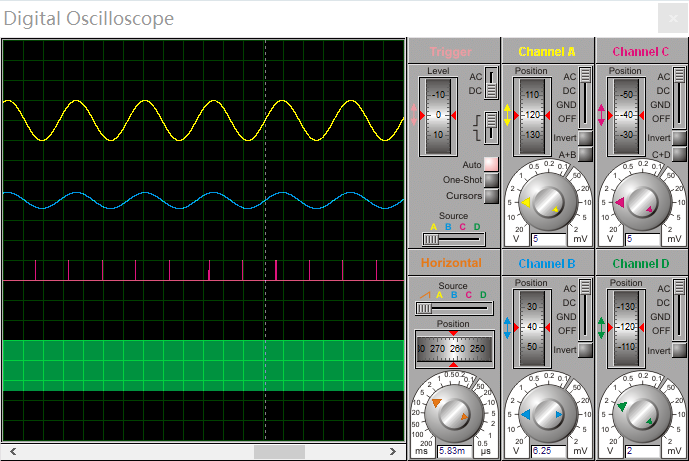
## 实验结果

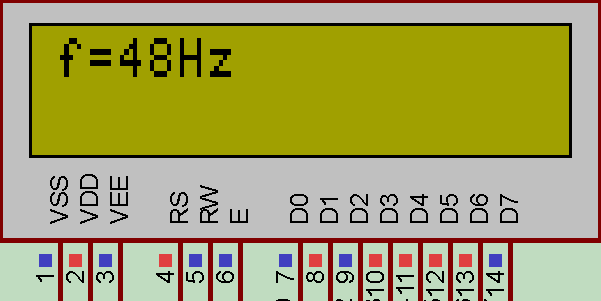
（1）设置频率为50Hz，信号幅值为满量程的90%



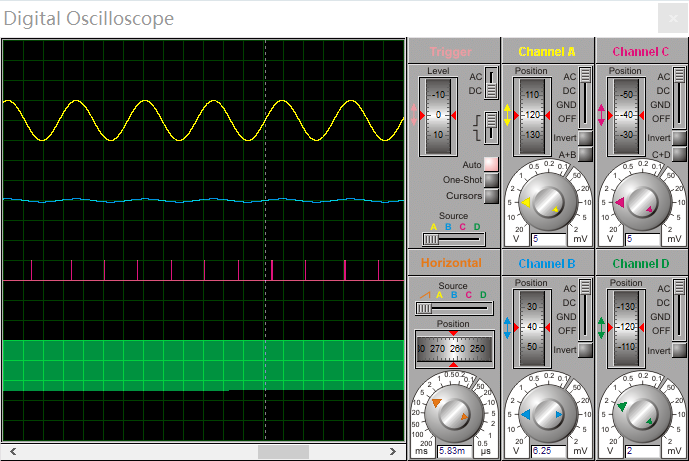


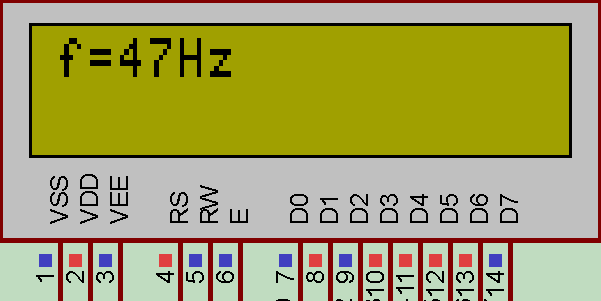
（2）设置频率为50Hz，信号幅值为满量程的50%





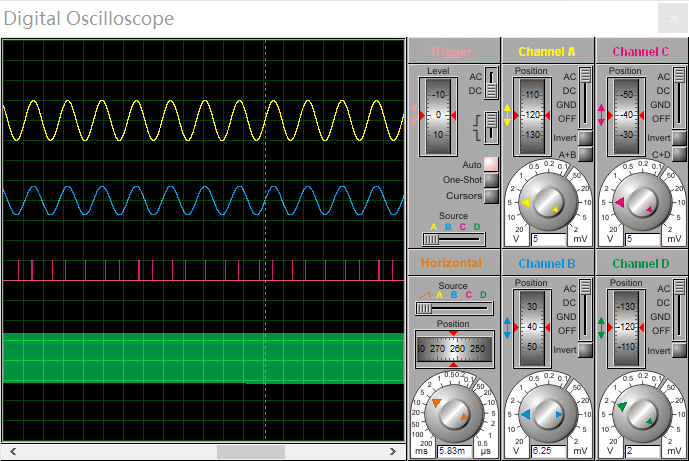
（3）设置频率为50Hz，信号幅值为满量程的10%

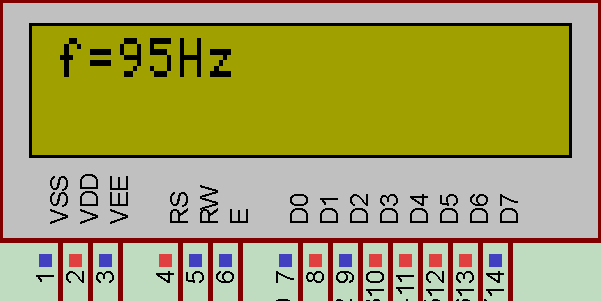




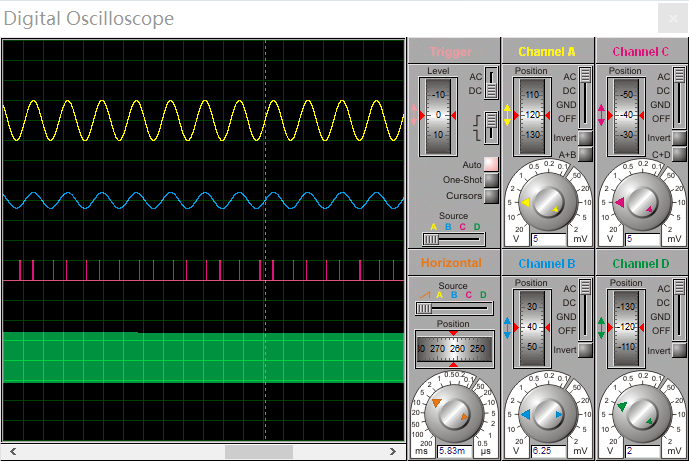
由上述结果可见，该条件下有一定的误差，约在3Hz左右

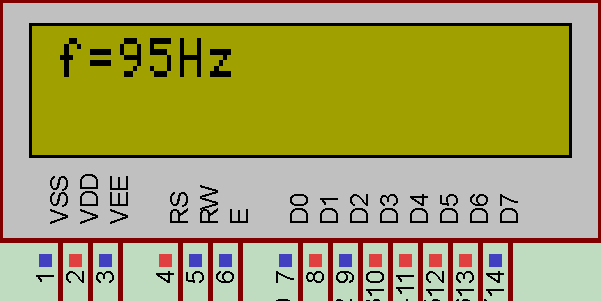
（4）设置频率为100Hz，信号幅值为满量程的90%



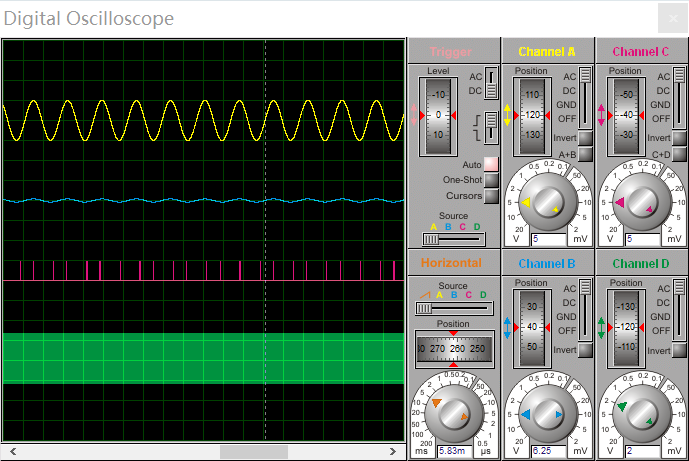


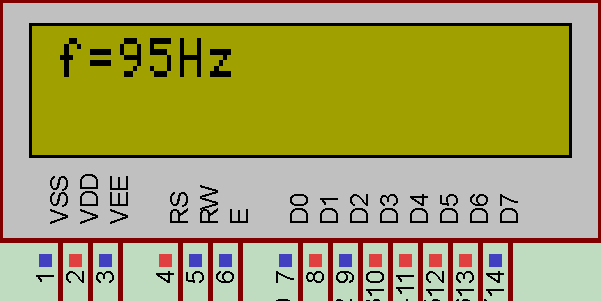
（5）设置频率为100Hz，信号幅值为满量程的50%





（6）设置频率为100Hz，信号幅值为满量程的10%





由上述结果可见，该条件下有一定的误差，约在5Hz左右，这些误差可能是由定时时间不够精确，或取样频率较低，使一些零点被遗漏导致的

## 实验源程序

#include<reg51.h>

#include<intrins.h>

#define uchar unsigned char

#define uint unsigned int

uchar code line1[] = "f=";

uchar code line2[] = "Hz";

uchar data disf[] = { 0xf,0xf,0xf,0xf,0xf,0xf };

//定义LCD控制引脚

sbit LCD\_RS = P2^0;

sbit LCD\_RW = P2^1;

sbit LCD\_EN = P2^2;

//定义ADC控制引脚

sbit CLK = P2^3;

sbit START = P2^4;

sbit OE = P2^5;

sbit OUT\_PULSE = P2^6;//5us脉冲

sbit EOC = P2^7;

sbit BF = P1^7;//用于检测忙子程序

uchar flag = 0;//用于防止检测忙死循环

uchar j, time1s;

uchar i = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*延时子程序\*\*\*\*\*\*\*\*\*\*\*\*/

void delays(uint ms)//参数为多少即为多少毫秒

{

uint t;

while (ms--)for (t = 0; t < 120; t++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*LCD 忙检查子程序\*\*\*\*\*\*\*\*\*\*\*\*/

bit lcd\_busy()

{

bit result;

LCD\_RS = 0;

LCD\_RW = 1;

LCD\_EN = 1;

delays(5);

result = BF; //返回数据最高位BF代表是否忙

LCD\_EN = 0;

flag++;

return result;

}

/\*\*\*\*\*\*\*\*\*\*\*\*写命令子程序\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_wcmd(uchar cmd)

{

{while (lcd\_busy() & (flag <= 10)); }//防止忙检测进入死循环

LCD\_RS = 0;

LCD\_RW = 0;

LCD\_EN = 0;

delays(5);

P1 = cmd;

delays(5);

LCD\_EN = 1;

delays(5);

LCD\_EN = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*写数据子程序\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_wdat(uchar dat)

{

{while (lcd\_busy() & (flag <= 10)); }//防止忙检测进入死循环

LCD\_RS = 1;

LCD\_RW = 0;

LCD\_EN = 0;

delays(5);

P1 = dat;

delays(5);

LCD\_EN = 1;

delays(5);

LCD\_EN = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*地址设置子函数\*\*\*\*\*\*\*\*\*\*\*\*/

void address(unsigned char x)

{

lcd\_wcmd(x + 0x80); //显示位置的确定方法规定为"80H+地址码x"

}

/\*\*\*\*\*\*\*\*\*\*\*\*LCD初始化子程序\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_init()

{

delays(5); //等待 LCD 电源稳定

lcd\_wcmd(0x38); //功能设定指令中 DL=1,N=1,F=0,8 位数据宽度,16\*2 显示，

//5\*7 点阵字符

delays(5);

lcd\_wcmd(0x0c); //显示开关控制指令中 D=1,C=0,B=1,显示开，关光标,不闪烁

delays(5);

lcd\_wcmd(0x06); //进入模式设置指令中 I/D=1,S=0,地址自动增加

delays(5);

lcd\_wcmd(0x01); //清除 LCD 显示内容,清屏指令 DB7-DB0 部分为 01H

delays(5);

}

/\*\*\*\*\*\*\*\*\*\*\*定时器T0中断服务程序\*\*\*\*\*\*\*\*\*\*\*\*/

void t0(void) interrupt 1

{

ET0 = 0;

TH0 = 0x3C;

TL0 = 0xB1;//使T0计数器实现50ms定时

i = i + 1;

if (i == 20) {

time1s = 1;

i = 0;

}//实现1s定时

ET0 = 1;

}

/\*\*\*\*\*\*\*\*\*\*\*定时器T1中断服务程序\*\*\*\*\*\*\*\*\*\*\*\*/

void t1(void) interrupt 3

{

ET1 = 0;

CLK = ~CLK;

ET1 = 1;

}

/\*\*\*\*\*\*\*\*\*\*\*AD数据采集子程序\*\*\*\*\*\*\*\*\*\*\*\*/

uint getad()

{

uint temp;

START = 0;

while (!EOC)

P0 = 0xff;//要从P0读数据必先写入0xff

OE = 1;

temp = P0;

OE = 0;

START = 1;

return(temp);

}

/\*\*\*\*\*\*\*\*\*\*\*\*频率显示子程序\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_dis(unsigned char x)

{

address(2);

for (j = 0; j < 6; j++)

{

disf[5 - j] = x % 10;

x = x / 10;

if (x == 0)break;

}

for (j = 0; j < 6; j++)

{

if (disf[j] != 0xf)lcd\_wdat(disf[j] + 48); delays(5);

}//以上两个循环将结果转十进制并显示

j = 0;

while (line2[j] != '\0')

{

lcd\_wdat(line2[j]);

j++;

};

}

/\*\*\*\*\*\*\*\*\*\*\*主程序\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

uchar temp1, temp2, f, zero;

temp1 = temp2 = j = f = OUT\_PULSE = OE = time1s = 0;

i = START = 1;

lcd\_init();

while (line1[j] != '\0')

{

lcd\_wdat(line1[j]);

delays(5);

j++;

}

TMOD = 0x21;//T0方式1，T1方式2，均为非门控定时器

EA = 1;//开总中断

TH0 = 0x3c;

TL0 = 0xb0;//使T0定时器实现50ms定时

ET0 = 1;

TR0 = 1;//启动定时器T0

TH1 = 0xfe;

TL1 = 0xfe;//T1实现100kHz定时，输出50kHz方波

ET1 = 1;

TR1 = 1;//启动定时器T1

while (1)

{

temp1 = temp2;

temp2 = getad();//转换完毕的数字信号传给temp2，上一次的数据留在temp1中

if (((temp1 < 128) && (temp2 > 128)) || ((temp1 > 128) && (temp2 < 128)) || (temp2 == 128))

{

zero++;

OUT\_PULSE = 1;

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

OUT\_PULSE = 0;//一个nop指令为1us，五个为5us

}

if (time1s == 1)

{

ET0 = ET1 = 0;

time1s = 0;

f = zero / 2;

lcd\_dis(f);

zero = 0;

ET0 = ET1 = 1;

}

}

}