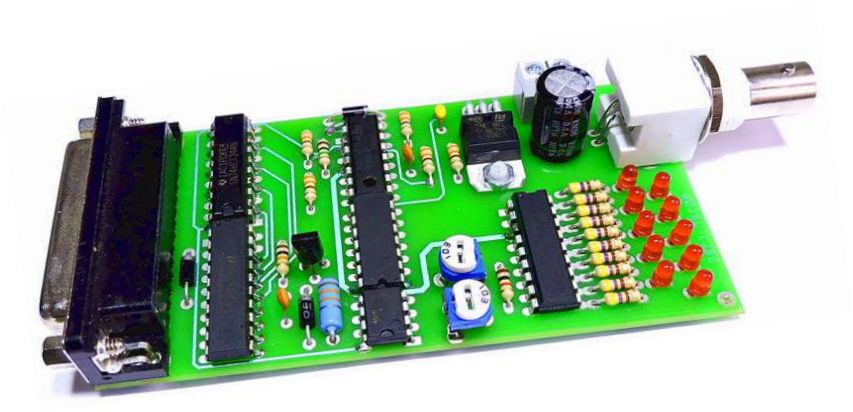
EPSICOM

Ready Prototyping

Colectia Proto Lab-Service **EP 0268**



Cuprins

PC-Based Oscilloscope
Osciloscop Pe Portul Paralel

- Avantaj Pret/Calitate
- Livrare rapida
- Design Industrial
- Proiecte Modificabile
- Adaptabile cu alte module
- Module usor de asamblat
- Idei Interesante

Idei pentru afaceri

Hobby & Proiecte Educationale

www.epsicom.com/kits.php
a division of **EPSICO Manufacturing**

Observarea, măsurarea, înregistrarea, memorarea și prelucrarea semnalelor lent variabile în domeniul industrial, medical, agricol, casnic,… în laboratoare ca placă de achiziție multi-canal.

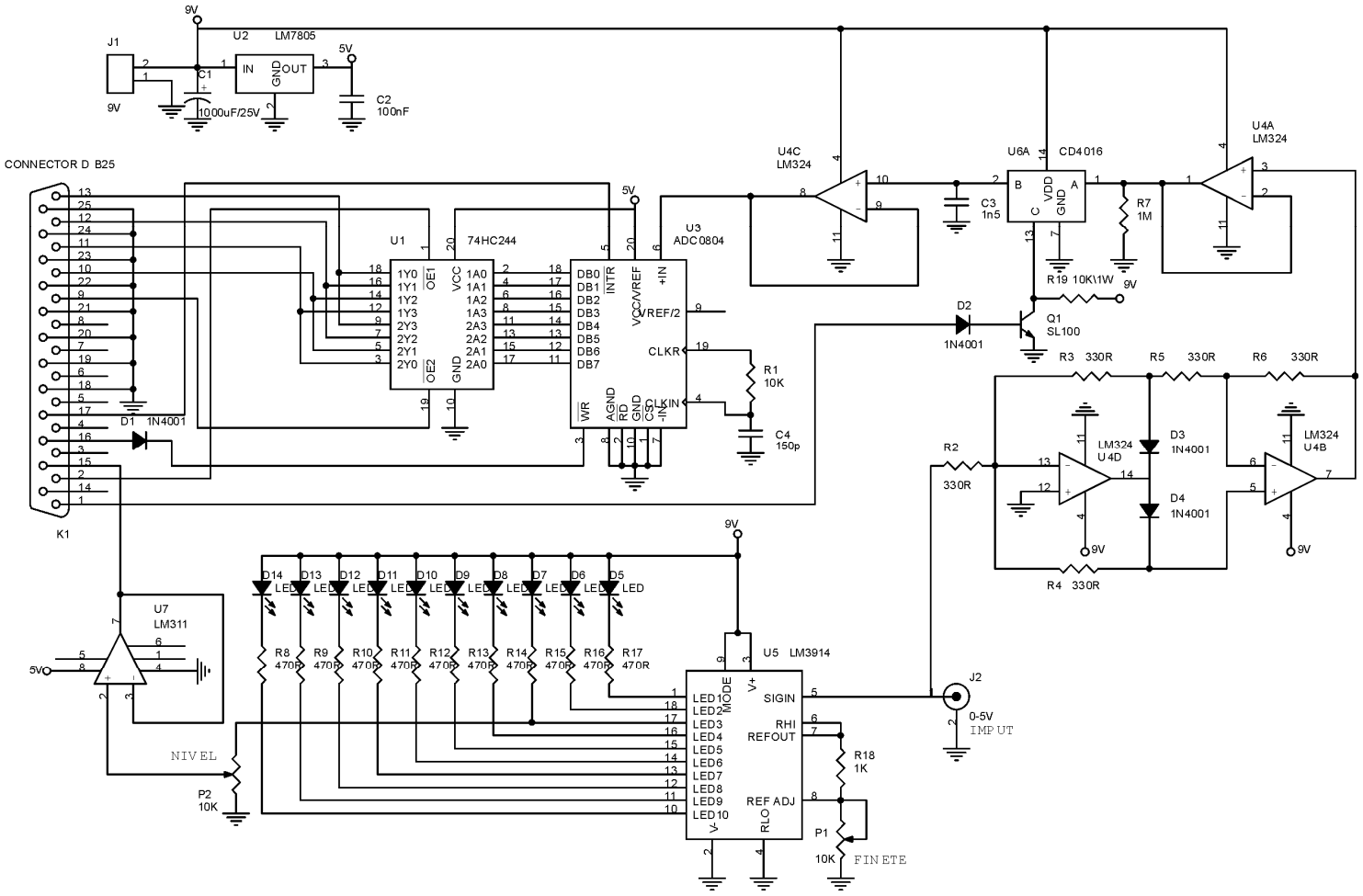
Funcționare

Semnalul de intrare este aplicat simultan atât unui redresor bialternanță cât și unui detector de trecere prin zero realizat cu U5 (display driver cu leduri). Semialternanțele sunt aplicate apoi eșantionat convertorului ADC pe 8 biți (Convertor Analog Numeric) tip AD0804. Punctul sensibil este cel de detecție a trecerii prin zero. Sunt prevăzute reglajele referință intrare și de nivel ieșire, P1 respectiv P2, iar semnalul este trimis prin pinul 15 al portului paralel către PC, zero pe semialternanța pozitivă. Din P1 se poate regla și domeniul de indicare cu leduri.

Prin pinul 1 al conectorului DB25 se primește comanda de deschidere a comutatorului analogic realizat cu U6 și se realizează eșantionarea semnalului, memorat apoi pe C3 (sample and hold), aplicat repetorului U4C și convertit cu ADC0804 (intrare pe pinul 6). Fiecare eșantion este deci convertit din semnal analogic în serii de coduri numerice. Timpul de conversie de 100µs este stabilit de valorile C4 și R1. Datele convertite numeric sunt depuse într-un buffer octal și apoi citite multiplexat de către PC în seturi de câte 4 biți, acesta trimițând semnalele de validare citire prin pinii 2 și 9 către U1 (dual buffer).

Pinul 5 al convertorului indică finalizarea conversiei printr-un "0" logic iar prin pinul 3 se se startează conversia.

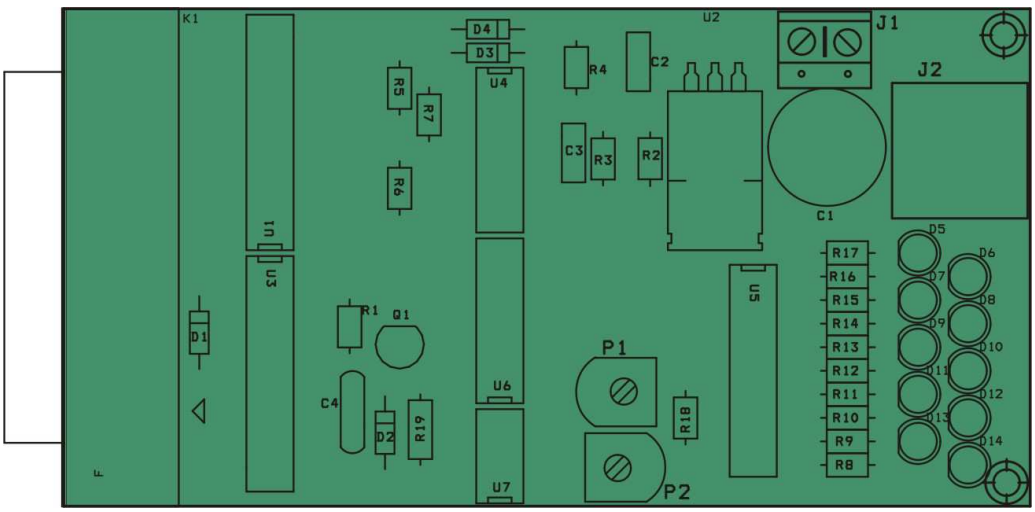
Alimentarea circuitului se face de la o sursă de 9V, tensiune ce va fi apoi stabilizată la 5V cu integratul LM7805. Softul scris în limbaj TurboC include funcții grafice de vizualizare, setari ale nivelului de tensiune și domeniului de frecvență.



Schema electrică

Lista de componente

Nr.Crt.	Componenta	Denumire	Valoare	Cant
1	C1	Condensator POL	1000µF/25V	1
2	C2	Condensator NP	100nF	1
3	C3	Condensator NP	1,5nF	1
4	C4	Condensator NP	150pF	1
5	D1,D2,D3,D4	Diodă	1N4001	4
6	D5,D6,D7,D8,D9,D10,D11, D12,D13,D14	LED	LED	10
7	J1	Conector con2	9V	1
8	J2	Conector BNC	0-5V	1
9	K1	Conector DB25 mamă	DB25	1
10	P1,P2	Semireglabil	10KΩ	2
11	R1	Rezistență	10KΩ	1
12	Q1	Tranzistor	SL100	1
13	R2,R3,R4,R5,R6	Rezistență	330Ω	5
14	R7	Rezistență	1MΩ	1
15	R8,R9,R10,R11,R12,R13, R14,R15,R16,R17	Rezistență	470Ω	10
16	R18	Rezistență	1KΩ	1
17	R19	Rezistență	10KΩ\1W	1
18	U1	C.I.	74244	1
19	U2	C.I.	LM7805	1
20	U3	C.I.	ADC0804	1
21	U4	C.I.	LM324	1
22	U5	C.I.	LM3914	1
23	U6	C.I.	CD4016	1
24	U7	C.I.	LM311	1



Amplasarea componentelor

Acest produs se livrează în varianta asamblată sau în varianta circuit imprimat + componente în scopuri educaționale și va fi însoțit de documentația completă de asamblare pe CD.

Dacă doriţi să aflaţi mai multe despre produsele noastre, vizitaţi situl **www.epsicom.com**

Dacă aţi întâmpinat probleme cu oricare dintre produsele noastre sau dacă doriţi informaţii suplimentare, contactaţi-ne prin e-mail **office@epsicom.com**

Pentru orice întrebări, comentarii sau propuneri de afaceri nu ezitaţi să ne contactaţi pe adresa **office@epsicom.com**

31 Sararilor Street I 200570 Craiova, Dolj, Romania I 0723.377.426, 0743.377.426

```
/* PROGRAM FOR PC OSCILLOSCOPE */
/*by M.M.VIJAI ANAND B.E (E.E.E) C.I.T */
#include
#include
#include
#include
#include
#include
#define data 0x0378
#define stat 0x0379
#define cont 0x037a


void graphics(int[],int[]); //FUNCTION TO DISPLAY GRAPH AND WAVEFORM
void settings(); //FUNCTION TO CHANGE THE SETTINGS(TIME AND VOLTAGE)
long int samp=7000; //PLEASE CHECK THESE VALUES WHEN CONVERSION IS
// NOT PROPER(+-3000)
float scale=1;
float times=1;
char again="a";
int number=800;
void main()
{
int i,j,k,a[1700],b[1700],c[1700],e[1700]; //This value 1700 is given when we want to compress the waveform
//done when we compress the time scale
long int b1;
clrscr();
settings();
while(again=="a")
{
for(i=0;i<number;i++)
{
outportb(cont,0x05^0x0b);
outportb(cont,0x04^0x0b);
e[i]=(inportb(stat)^0x80)&0x08;
for(b1=0;b1<=samp;b1++) //sampling
time is approximately 50 µsec
{}
outportb(cont,0x05^0x0b);
outportb(cont,0x01^0x0b);
outportb(cont,0x05^0x0b);
while((inportb(cont)&0x08)==0x00) //converstion time is approximately 100 µsec
{
}
outportb(data,0xf0);
a[i]=(inportb(stat)^0x80)&0xf0;
outportb(data,0x01);
b[i]=(inportb(stat)^0x80)&0xf0;
outportb(data,0xff);
}
for(i=0;i<number;i++)
{
a[i]=a[i]>>4;
c[i]=a[i]+b[i];
c[i]=c[i]*0.0196*45/scale;
}
graphics(c,e);
}
}
void graphics(int a1[],int e1[])
{
int gd=DETECT,gm,max,may,a,b,c,im,error,get=5;
char str[10],*st="-",d;
clrscr();
initgraph(&gd,&gm,"c: c gi"); //use default bgi path
error=graphresult();
if(error != grOk)
{
printf("Graphics error %s /n",grapherrormsg(error));
//reports error when
//graphics is not set
printf("PRESS ANY KEY TO EXIT");
getch();
exit(1);
}
setbkcolor(LIGHTCYAN);
setcolor(MAGENTA);
settextstyle(0,0,2);
max=getmaxx();
may=getmaxy();
may=may-20;
outtextxy(0,may,"OSCILLOSCOPE");
settextstyle(0,0,1);
setcolor(BLUE);
outtextxy(max-200,may+2,"press "a" for next sample");
```

```
setcolor(BROWN);
outtextxy(max-200,may+10,"press any key to exit");
setcolor(GREEN);
settextstyle(0,0,0);
for(a=0;a<=may;a+=get)
{line(0,a,800,a);
}
for(a=0;a<=max;a+=get)
{
line(a,0,a,may);
}
setcolor(BROWN);
setlinestyle(0,3,0);
line(max/2,0,max/2,may);
line(0,may/2,max,may/2);
setcolor(RED);
for(a=0,c=0;a<=max;a+=50,c++)
{
putpixel(a,may/2,BLUE);
itoa((a-c*30)*times/2,str,10);
outtextxy(a+3,may/2+3,str);
}
for(b=(may/2)-45,c=1;b>=0;b-=45,c++)
{
itoa((c*scale),str,10);
putpixel((max/2),b,BLUE);
outtextxy((max/2)+3,b+3,str);
}
for(b=(may/2)+45,c=1;b<=800;b+=45,c++)
{
itoa((c*scale),str,10);
strcat(st,str);
putpixel((max/2),b,BLUE);
outtextxy((max/2)+2,b+2,st);
strcpy(st,"-");
}
setcolor(MAGENTA);
outtextxy(max-80,may/2+30,"time(msec)");
settextstyle(0,1,0);
outtextxy((max/2)-10,0,"volt(s)");
setlinestyle(0,0,0);
setcolor(RED);
moveto(0,may/2);
for(b=0,c=0;b<=number;c+=1, b++)
{
if(e1[b]!=0x08)
{
lineto(c*times,((may/2)-a1[b]));
}
else
{
lineto(c*times,((may/2)+a1[b]));
}
}
again = getch();
closegraph();
restorecrtmode();
}
void settings()
{
int gd=DETECT,gm,error,max,may,b;
char c,d,e[2],m,*n;
times=1;
initgraph(&gd,&gm,"c: c gi"); //default bgi directory path
error=graphresult();
if(error != grOk)
{
printf("Graphics error %s /n",grapherrormsg(error));
printf("PRESS ANY KEY TO EXIT");
getch();
exit(1);
}
max=getmaxx();
setbkcolor(LIGHTBLUE);
settextstyle(1,0,0);
setcolor(BROWN);
outtextxy(max/2-60,20,"SETTINGS");
line(0,60,800,60);
setcolor(MAGENTA);
settextstyle(1,0,1);
outtextxy((max/4)-70,80,"Voltage Scale");
settextstyle(0,0,0);
setcolor(BROWN);
outtextxy(10,120,"DEFAULT :");
outtextxy(10,120," 1 unit = 1 volt");
setcolor(RED);
outtextxy(10,170,"TYPE "C" TO CHANGE AND "D" TO DEFAULT");
c=getch();
if(c=="c")
{
outtextxy(10,200,"TYPE 1 for 1 unit = 2 volt");
```

```
outtextxy(10,240,"TYPE 2 for 1 unit = 4 volt");
outtextxy(10,300,"TYPE 3 for user defined");
switch(getch())
{
case "1" :{ scale=2;
break;
}
case "2" :{scale = 4;
break;
}
case "3" :{ outtextxy(10,340,"TYPE VALUES FROM 1 TO 9 (minimize) or m to (magnify)");
d=getch();
if(d=="m")
{
outtextxy(10,360,"TYPE a (1 unit = 0.5 volt) or b (1 unit = 0.25 volt)");
switch(getch())
{
case "a":
{
scale=0.5;
break;
}
case "b":
{
scale=0.25;
break;
}
}
}
else
{ e[0]="0';
e[1]= "0";
e[2]=d;
scale=atoi(e);
break;
}
}
}
}
setcolor(BROWN);
outtextxy(10,380,"TYPE C TO CHANGE TIME SETTINGS");
m=getch();
if( m=="c")
{
cleardevice();
outtextxy(10,20,"X AXIS 1 unit= 10msec CHANGE TO x(10msec)");
outtextxy(10,40,"TYPE "a" IF x IS (2 to 9) ,"b" IF x IS (10 to 99) AND "c" IF x IS (.5 TO .9)");
switch(getch())
{
case "a":
outtextxy(10,60,"x value is ....");
n[0]=getch();
times=atoi(n);
itoa(times,n,10);
outtextxy(10,70,n);
break;
case "b":
outtextxy(10,60,"x value is ....");
n[0]=getch();
n[1]=getch();
times=atoi(n);
itoa(times,n,10);
outtextxy(10,70,n);
break;
case "c":
outtextxy(10,60,"x value is...");
getch();
n[0]=getch();
times=atoi(n)*0.1;
outtextxy(10,70,"scale decremented");
break;
}
number=800;
if(times<1)
{number=number/times;
}
getch();
}
closegraph();
restorecrtmode();
}</number;i++)
</number;i++)
```