PC-BASED OSCILLOSCOPE

M.M. VIJAI ANAND

This circuit conditions different signals of frequency below 1 kHz and displays their waveforms on the PC's screen. The hardware is used to condition the input waveform and convert it to the digital format for interfacing to the PC. The software for acquiring the data into the PC and displaying the same on its screen is written in Turbo C.

The input waveform (limited to 5V peak-to-peak) is first applied to a full-wave rectifier comprising op-amps A1 and A2 of quad op-amp LM324 (IC4) and a zero-crossing detector built around LM3914 dot/bar display driver (IC8) simultaneously.

The full-wave rectifier rectifies the input signal such that the negative half cycle of the input signal is available in the positive side itself, so both the half cycles are read as positive when it is given as input to the ADC. During positive half cycle, diode D3 is on and diode D4 is off, and op-amps A1 and A2 act as inverters. Thus the output is a replica of the input. During the negative half cycle, diode D3 is off and diode D4 is on. With R2 = R3 = R4 = R5 = R6 = R = 330 ohms, the voltage (V) at output pin 1 of op-amp A1 is related to the input voltage (Vi) as follows:

$$Vi/R + V/(2R) + V/R = 0$$

$$V = -(2/3)Vi$$

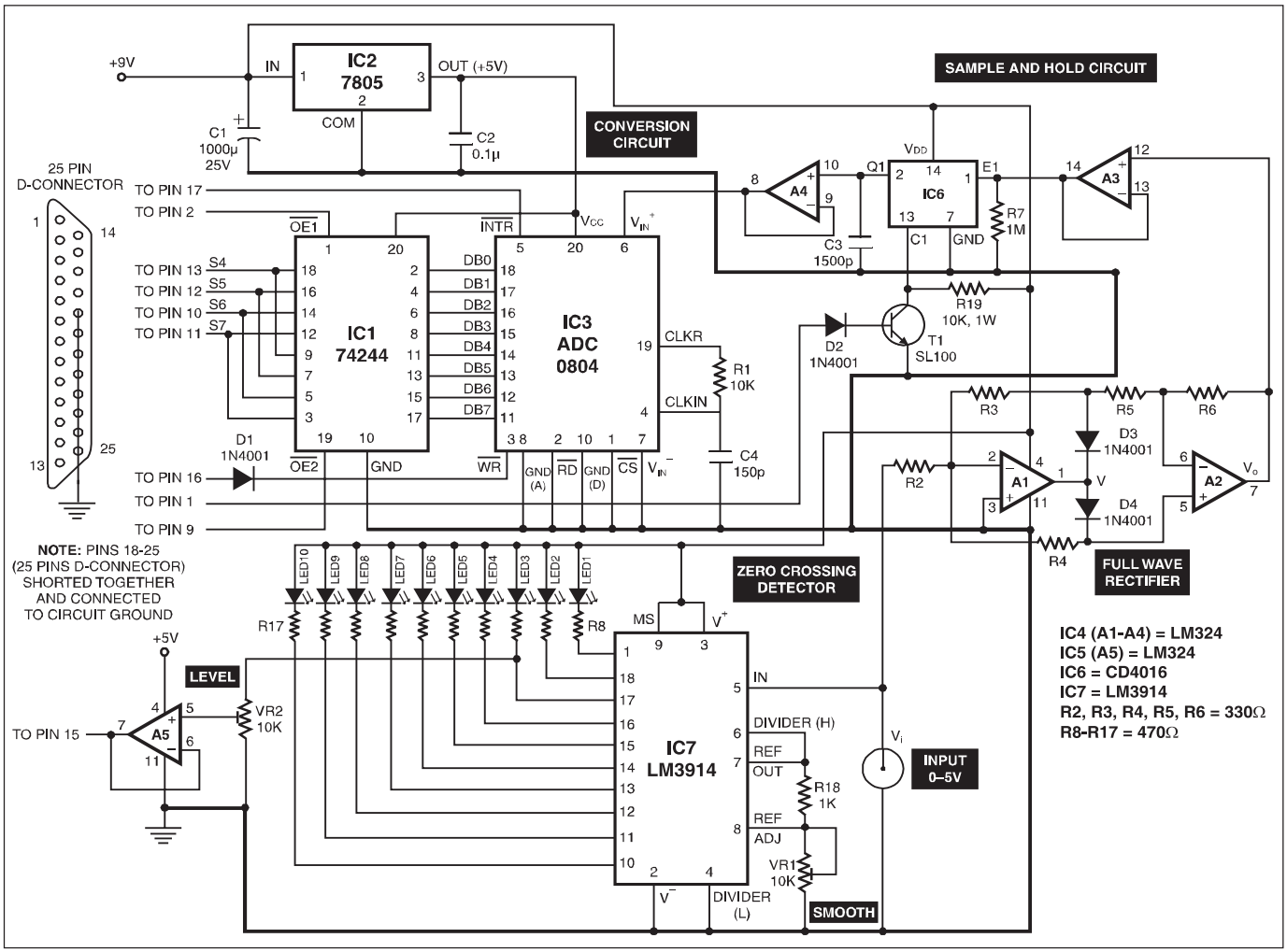
The final output voltage (Vo) at pin 7 of op-amp A2 is given by the following relationship:

$$Vo = (1 + R/2R)(-2Vi/3) = -Vi$$

As Vi is negative, the output voltage is positive.

The zero-crossing detector detects whether the cycle is positive or negative. It is the most critical part of the circuit and if it operates improperly, the symmetry of the analogue signal displayed in the PC monitor gets affected. At the zero-crossing instant when the input signal transits to negative side, the zero-crossing detector informs the PC by taking pin 15 of 25-pin 'D' connector of the parallel port high. The input at pin 15 of 'D' connector goes low when the input signal transits to positive side. The zero-crossing detector communicates with the PC through bit D3 of the status port 379Hex.

The zero-crossing detector has been realised using LM3914 IC. You may adjust VR1 such that the last LED (LED10) goes off when the input signal transits negative side of the input waveform. The LM3914 itself rectifies the input signal and allows



PARTS LIST

Semiconductors:

IC1	- 74244
IC2	- 7805
IC3	- ADC 0804
IC4, IC5	- LM324
IC7	- CD4016
IC8	- LM3914
T1	- Transistor SL100
D1-D4	- 1N4001 switching diode
L1-L10	- Red LED

Resistors (all ¼-watt, ±5% carbon, unless stated otherwise):

R1	- 10kilo-ohm
R2-R6	- 330-ohm
R7	- 1Mega-ohm
R8-R17	- 470-ohm
R18	- 1kilo-ohm
R19	- 10kilo-ohm/1watt
VR1, VR2	- 10kilo-ohm Preset

Capacitors:

C1	- 1000µF, 25V electrolytic
C2	- 0.1µF, ceramic
C3	- 1500PF, ceramic
C4	- 150PF, ceramic

Miscellaneous:

25 PIN D-Connector Female
2 PIN SIP CONNECTOR
PCB

only positive half of the cycle.

The output from the full-wave rectifier is applied to the input of a sample-and-hold circuit comprising op-amps A3 and A4 of the LM324 (IC5), capacitor C3, transistor T1 (SL100), and analogue switch IC6 (CD4016). This circuit samples the input signal, i.e. it divides the waveform into a number of voltages or points and inputs each voltage level (with a delay) to the ADC for conversion into the digital format. Op-amps A3 and A4, along with a switch from IC CD4016 and a 1500pF capacitor with sampling time of 20 µs, are used as voltage followers/buffers.

When the base of transistor T1 is made low via strobe pin 1 (bit Do of I/O port 37A) of 25-pin D connector of the parallel port, the transistor stops conducting and the voltage at its collector goes high. The high voltage at the collector of transistor T1 closes the switch inside CD4016. As a consequence, the analogue input signal is applied to the capacitor, which charges towards the signal voltage.

When the switch is subsequently opened by applying a logic-high voltage from pin 1 of 'D' connector to the base of transistor T1, the capacitor retains the voltage with a loss of about 20 mV/sec and this voltage is given to input pin 6 of the ADC0804 (IC3) via buffer A4 for conversion to the digital format. When the number of sampling points in the input signal waveform is increased, the reconstructed waveform becomes more accurate.

The ADC0804 is compatible with microprocessors. It is a 20-pin IC that works with 5V supply. It converts the analogue input voltage to 8-bit digital output. The data bus is tristate buffered. With eight bits, the resolution is 5V/255 = 19.6 mV.

The inbuilt clock generator circuit produces a frequency of about 640 kHz with R1 = 10 kilo-ohms and C4 = 150 pF, which are the externally connected timing components. The conversion time obtained is approximately 100 µs. The functions of other pins are given below:

Pin 1 ($\overline{CS}$): This is active-low chip-select pin.

Pin 2 ($\overline{RD}$): This active-low pin enables the digital output buffers. When high, the 8-bit bus will be in Hi-Z state.

Pin 3 ($\overline{WR}$): This active-low pin is used to start the conversion.

Pin 9 (Vref/2): This is optional input pin. It is used only when the input signal range is small. When pin 9 is at 2V, the range is 0-4V, i.e. twice the voltage at pin 9.

Pin 6 (V+), Pin 7(V-): The actual input is the difference in voltages applied to these pins. The analogue input can range from 0 to 5V.

In this circuit, pins 1 and 2 are always made low, so the IC and the buses are always enabled. Pin 9 is made open, as we use analogue input with 0-5V range. Pin 7 is grounded.

Pin 5 ($\overline{INTR}$): This active-low pin indicates the end of conversion. It is connected to pin 17 (bit D3 of I/O port 37A) of 'D' connector. (Note that this bit is inverted.)

The start-of-conversion command via pin 16 of 'D' connector is applied to pin 3 of the ADC0804. Since we cannot read 8-bit digital data output from ADC through the 4-bit status port at a time, we divide it in two 4-bit parts and read. Hence the ADC data output is multiplexed through two 4-bit sections of octal buffers of IC1 (74244) with the help of output-enable signals from pins 2 and 9 of 'D' connector to pins 1 and 19 ($\overline{OE1}$ and $\overline{OE2}$, respectively) of IC1. The digital data output from IC1 is interfaced to the PC via pins 13 (S4), 12 (S5), 10 (S6), and 11 (S7) of status input port 379H of 'D' connector.

The circuit uses 9V and 5V regulated DC supply voltages as shown in the circuit diagram.

A PC printer port is an inexpensive platform for implementing low-frequency data acquisition projects. Each printer port consists of data, status, and control port addresses. These addresses are in sequential order; for example, if the data port address is 0x0378, the corresponding status port address is 0x0379 and the control port address is 0x037a. The port addresses for parallel ports are summarised below:

Printer	Data port	Status port	Control port
LPT1	0x0378	0x0379	0x037a
LPT2	0x0278	0x0279	0x027a
LPT3	0x03bc	0x03bd	0x03be

(***EFY Lab note***. For details of the parallel port pins, refer 'PC-based Dial Clock with Timer' project published in June 2002 issue of EFY.)

The software, written in C programming language, is user-friendly and easy-to-understand. It gets data from the developed hardware circuit and displays it in the graphical screen with some changes.

The C program includes two user-defined functions with the main function: graphics( ) and settings( ). The settings( ) function is used to adjust the voltage and time scale. The graphics( ) function is used to display the waveform on the screen. The sample control signal is used to close the switch in the sample-and-hold circuit, so the capacitor charges towards the analogue input voltage. After the sampling is over, the switch is opened using the same signal. Then the start-of-conversion control signal is given to start the conversion. The sampling time is approximately 20 µs and the conversion time is approximately 100 µs.

After the conversion is over, the 8-bit binary data for the specific voltage sample is available in the data bus of the ADC. Since the PC accepts only 4-bit data through the status port (379H), the 8-bit data must be split into two 4-bit data, which are accepted one after another. This is done by IC 74244, which is controlled by D0 and D7 bits of the data port. Then the two 4-bit data are packed to get the final 8-bit data.

The default BGI directory path is set as 'c:\tc\bgi'. The sampling time is decided by the 'for' loop that uses the samp value. The maximum delay produced should be greater than 20 µs, which is the maximum acquisition time of the capacitor. When the sample value is increased, the number of points on the input signal decreases and therefore the accuracy decreases. The time scale may be calibrated with 50Hz sine wave as reference.

Note. Mount a 25-pin D-type female connector on the PCB. Use 25-pin 'D' male-to-male connector for connecting PCB to computer's female 25-pin LPT port 'D' connector. It is a demo PC based oscilloscope only and may not be suitable for real time application.

PROGRAM IN 'C' FOR PC OSCILLOSCOPE

```
/* PROGRAM FOR PC OSCILLOSCOPE */
/*by M.M.VIJAI ANAND B.E (E.E.E) C.I.T */
#include<dos.h>
#include<time.h>
#include<stdio.h>
#include <graphics.h>
#include<string.h>
#include<stdlib.h>
#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:\\tc\\bgi");                    //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:\\tc\\bgi");              //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();
}
```