Journal of Neuroscience Methods, 19 (1987) 61–68
Elsevier

NSM 00646

A Turbo Pascal program for on line spike data acquisition and analysis using a standard serial port

Enrique Soto and Rosario Vega

Departamento de Ciencias Fisiológicas, Instituto de Ciencias, Universidad Autónoma de Puebla, Puebla (México)

(Received 15 April 1986)
(Revised 31 April 1986)
(Accepted 20 August 1986)

Key words: Data acquisition; Interval histogram; Frequency histogram; Microcomputer; IBM PC; Spike train; Vestibular system

A Turbo Pascal program for data acquisition and analysis is presented. The program detects incoming data as TTL pulses through the serial port of an IBM-PC and compatible computers and displays the instantaneous frequency plot and the interval histogram on-line. Novel features of this Pascal program are: (1) no special hardware requirements and (2) on-line analysis of more than one source at a time. No hardware additions to the supplied computer and easy operation make this program a valuable tool, directly useable in an IBM-PC and compatible computers.

There are biological processes in which specific sequences of events are the relevant data. Neurophysiologists very often deal with these kind of processes in which little or no basic importance may be given to the particular wave form of the signal. One such process is the neuronal activity recorded as action potential trains (spikes). In this case the information transmitted by a neuron is assumed to be determined by the specified distribution of spikes over time (Moore et al., 1966; Glaser and Ruchkin, 1976).

Usually, the data obtained from neuronal recording are displayed as frequency (rate) or interval histograms. Neurophysiologists commonly use special purpose computers for this kind of analysis and very often the task is accomplished by means of analog to digital converters or of some kind of special electronic timer-counters (Cohen and Pfaff, 1984).

Our main interest was to develop a method for event counting and interval analysis in our Columbia PC compatible computer, which depends exclusively on the supplied computer hardware, without having to develop electronic interfaces or analog to digital converters.

Correspondence: E. Soto, Departamento de Ciencias Fisiológicas, ICUAP, Universidad Autónoma de Puebla, Apartado Postal 406, Puebla, Pue, México.

The program presented here allows the computer to accept TTL (0.2–0.8 V logical zero; 2.0–5.0 V logical one) data stream directly through its "standard" serial port, and to analyze their time distribution, plotting the rate and the interval histogram as data arrive. The program accepts data from more than one source at the same time, although there is a tradeoff between temporal resolution and number of data sources. Our program allows data acquisition from two sources with a time resolution of 1 ms and on-line display of rate plot and interval histogram of up to 20 min. In addition the system also includes the option of storing data on disk and obtaining elemental statistics.

Material and Methods *

The basic computer set-up required for the acquisition program is an IBM PC compatible computer with a 64 K memory, one disk drive and a free input port (serial port). In addition to the basic computer hardware the Turbo Pascal compiler is needed.

The system imposes only one condition on the incoming data: the signal must be a TTL compatible signal.

In the IBM-PC and compatible computers, the serial communication port is an RS-232 interface, in which the TTL signal from the experiment may be directed through the handshaking circuits of the interface. In our system we connect a double threshold window discriminator to pins 1–5 (above window) and 1–22 (within window) of the serial port (normally this port is intended for modem connection; pin 1 is ground, pin 5 is clear to send, and pin 22 is the ring call). The computer will identify the presence of TTL pulses as a logical one. Then, the value of the port registers will depend on the presence of pulses in each pin. Turbo Pascal gives the user direct access to the computer ports with the possibility of assigning the value of the port to a variable. In the IBM-PC and compatible computers serial port one is at address hexadecimal 3FE (Norton, 1985), the Pascal sentence x: = port[$3FE] will assign the value of serial port status register to "x". Port value will be 0 if no pulse, 17 if there is a pulse in pin 5; 64 if there is a pulse in pin 22 and 81 if there are pulses in both pins 5 and 22. Programming is straightforward: port value reading and subsequent assessment of whether there is a pulse in some pin of the interface (procedure capture data and capturefreqDouble of Appendix 1).

Accurate timing is achieved by programming the 8253 clock chip of the PC so that it ticks once each millisecond (Jourdain, 1986). Port addressing rate should be defined on the basis of the duration of incoming data. In our case the window

---

* Apple II+ is a trademark of Apple Computers; Columbia and Columbia 1600-IV, are trademarks of Columbia Data Products, Inc.; Commodore 64 is a trademark of Commodore Business Machines; IBM PC, is a trademark of International Business Machines, Inc.; Lotus 1–2–3 is a trademark of Lotus Development Corp.; MS DOS is a trademark of Microsoft Corporation; Texas TI 99/4A is a trademark of Texas Instruments; Turbo Pascal is a trademark of Borland International.

discriminator signals are 1.1 ms, thus reading the port each millisecond suffices to prevent any loss of data.

The program described here allows the user to select from the start menu the desired analysis of data (frequency histogram from one source, frequency histogram from two sources, interval histogram of one source and both frequency histogram and interval histogram from one source). The user may also select the bin width, the analysis period and one of two trigger modes: the first signal in pin 5 channel or a keyboard command. The data acquisition procedure continues until it reaches the selected number of bins (maximum 300), or until the user presses the "F1" key. Pressing any other key causes a mark to appear indicating the onset of experimental manipulation. Histograms are displayed on-line. When data acquisition finishes elemental statistics of data (mean and standard deviation) and protocol details are also displayed. Finally, the raw data may be stored on disk (Lotus 1–2–3 compatible archive) for subsequent processing.

Results and Conclusion

In our laboratory we have been using this program for data acquisition and analysis of unitary recordings from primary afferent nerve fibers of the vestibular system of the axolotl (*Ambystoma mexicanum*). In this preparation we have studied the effect of excitatory amino acids on spontaneous discharge of these fibers. Fig. 1 shows the frequency and interval histogram from a typical experiment in which the spontaneous afferent discharge is initially recorded; then (mark), D-aspartic acid (10^{-3} M) is microperfused in the vicinity of the sacculi. The drug produces a clear excitatory effect on this fiber.

Although this program was developed for the Columbia PC compatible computer, it runs in the IBM PC, and its fundamentals are easily adapted to practically

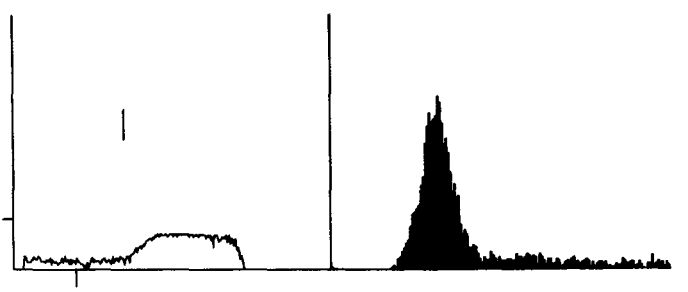

Fig. 1. Frequency and interval histogram from a primary afferent fiber of the macula sacculi of the axolotl (*Ambystoma mexicanum*). Upper mark in frequency histogram indicates the moment of application of D-aspartic acid (10^{-3} M). Excitatory effect of D-aspartic acid is clearly appreciated from the rate plot interval histogram of the whole epoch is shown at right. Bin width is of 1 ms. Vertical calibration applies to both records representing twenty spikes per second, and twenty events for bin size. Horizontal calibration applies also to both frequency and interval histograms. It represents 1 min for frequency histogram and 60/bin ms for interval histogram.

any commercially available computer. Indeed, in our laboratory, we have been able to implement this program for counting of behavioral events on the Commodore 64, and the Texas TI 99/4A computer. However the time resolution of these home computers is not suitable for neuronal train spike analysis.

The Pascal program is easily understandable and can be adapted to many computers and experimental conditions. The experienced user can modify the acquisition procedure to include more lines of data input and output through the port. The timer chip can also be programmed to modify timing. It is worth noting that our approach will allow the computer to accept data through any of the computer ports (joystick, parallel, second serial), by only changing port address and values resulting from port reading. [When working on the revised version of this manuscript we found out that a very similar approach had been developed for the Apple II + computer by Gras and Horner (1985).]

Acknowledgements

The authors wish to thank Javier Birwe and Julio Glockner for critical reading of the manuscript and Gertrudis Payas for the English version. This research has been partially supported by PRONAES Grant 84010196 from the Secretaria de Educación Pública (SEP) and a 1985 grant from the Consejo Estatal de Ciencia y Tecnología (CECyT), Puebla, México.

References

Cohen, M.S. and Pfaff, D.W. (1984) On-line data acquisition system using an Apple computer: ISI and PST histograms, Brain Res. Bull. 13: 205–223.

Glaser, E.M. and Ruchkin, D.S. (1976) Principles of Neurobiological Signal Analysis, Academic Press, London.

Gras, H. and Horner, M. (1985) A program for time control and measurement in electrophysiological experiments with an Apple II microcomputer, J. Electrophysiol. Technol. 12: 197–208.

Jourdain, R. (1986) Programmer's Problem Solver for the IBM PC, XT,&AT, Brady Communications Company, U.S.A.

Moore, G.P., Perkel, D.H. and Segundo, J.P. (1966) Statistical analysis and functional interpretation of neuronal spike data, Annu. Rev. Physiol., 28: 493–523.

Norton, P (1985) Programmer's guide to the IBM PC, Microsoft Press, U.S.A.

APPENDIX 1.

```
PROGRAM freq_INTERVAL;          { July, 1986 }

TYPE datarray=ARRAY[0..300] OF Integer;
     named=string[12];
VAR
  time_tick:Integer ABSOLUTE $0040:$006C;{points to tick count of BIOS timer}
  nBin,nEvent,experiment,bin,i,interval_n,time_init,time_end:Integer;
  data_above,data_between,his_a,zdata:datarray;
  option,continue,any,trigger:Char;
  archive:Text;
```

```
PROCEDURE Menu;                          {select action and beginning parameters}
BEGIN IF any <> '1'                {decide if change of action has been selected}
  THEN BEGIN
         REPEAT
    ClrScr;TextMode(2);GoToXY(3,6);
    WriteLn('1 : BIN RATE '); GoToXY(3,9);
    WriteLn('2 : INTERVAL HISTOGRAM ');GoToXY(3,12);
    WriteLn('3 : BIN RATE AND INTERVAL HISTOGRAM ');GoToXY(3,15);
    WriteLn('4 : Double BIN RATE ');
    GoToXY(3,18); Read(Kbd,option);
         UNTIL (option>='1') AND (option <='4');
       END;
 REPEAT
    ClrScr; GoToXY(3,10);
    WriteLn(' Experiment number ');
    Read(experiment); GoToXY(3,12);
    IF option<>'2'                                    {if no interval histogram}
    THEN BEGIN
           WriteLn(' bin width in sec ');Read(bin);
           GoToXY(3,14); WriteLn(' Processing time ( in bins ) ');
           Read(nBin);  nEvent:=30000;
         END
    ELSE BEGIN
           Writeln(' total number of events ? ');
           Read(nEvent); bin:=1; nBin:=1;
         END;
    GoToXY(3,16);
    WriteLn(' Trigger mode: Keypressed - K; External - E ');
    Read(trigger);
  UNTIL (bin>=1) AND ((nBin>=1) AND (nBin<=300));
END;

PROCEDURE Avrg_Ed(N,locate:integer;hf:boolean);{calculates basic statistics}
VAR i,j,h:Integer;
    iReal,dataReal,average,sum,deviation,sqr_dat:Real;
BEGIN
  sum:=0;sqr_dat:=0;deviation:=0;average:=0;j:=1;

 FOR i:=2 to 300 DO
   BEGIN
     h:=0; dataReal:=zdata[i];
     IF hf = true THEN                              { if interval histogram }
     BEGIN
       j:=i;
       WHILE h < zdata[i] DO
       BEGIN
         iReal:=i; h:=h+1;
         sqr_dat:=sqr_dat+Sqr(iReal);
       END;


    END
    ELSE sqr_dat:=sqr_dat+Sqr(dataReal);
    sum:=sum+dataReal*j;
  END;
  gotoxy(45,3);
if N=0 THEN write(' no data ')
       ELSE BEGIN
deviation:=Sqrt((sqr_dat-(Sqr(sum)/n))/(n-1));
average:=sum/N;
gotoxy(locate,25);
Write('Average  ',Round(average));
Write('   SD  ',Round(deviation));
Write('  N ',N);
            END;
read(kbd,any);
  END;

PROCEDURE datafile(which:named;locate:integer);                {disk store}
VAR save:Char;name_arch:named;
BEGIN
```

```
  Port[$43]:=54;                                        {restores normal timing}
  Port[$40]:=Lo(0);
  Port[$40]:=Hi(0);
  gotoxy(locate,2);
  Write('Store data ',which,' ? y / n');
  Readln(save);
  IF (save='Y') OR (save='y') THEN
  BEGIN  gotoxy(locate,4);
    Write('Name archive '); Read(name_arch);
    Assign(archive,name_arch); Rewrite(archive);
    FOR i:=1 TO 300 DO
    WriteLn(archive,zdata[i]:4);
    Close(archive)
  END;
END;

PROCEDURE graph_freq(v,paint:Integer);                  {axis and calibration}
BEGIN
  IF (i>=nBin) THEN Write('  DONE    ')
  ELSE Read(kbd,continue);
  IF (continue=#59) OR (i>=nBin) THEN

  BEGIN             {if F1 have been pressed or processing time is complete}
    nBin:=i;
    FOR i:=1 TO nBin DO
      BEGIN
        Draw(i+9,180-data_above[i-1],i+10,180-data_above[i],paint);
        Draw(319+i,V*180-data_between[i-1],320+i,V*180-data_between[i],paint);
      END;
    Draw(10,170,3,170,paint);                           { calibration in y }
    Draw(60 DIV bin,180,60 DIV bin,187,paint);          { calibration in x }
    Draw(10,180,310,180,3);Draw(10,180,10,90,paint);                {axis}
    Draw(320,180,620,180,v);Draw(320,180,320,90,v);
    IF paint=3 THEN BEGIN         { option 1 or 2 or 4 have been selected }
      zdata:=data_above;
      Avrg_Ed(nBin,5,false);                            {call statistics}
      datafile('above',3);                      {call disk store of raw data}
                    END;

     IF V=1 THEN BEGIN                         { option 4 have been selected }
       zdata:=data_between;
       Avrg_Ed(nBin,45,false);
       datafile('between',43);
                 END;
     IF (option='2') OR (option='3') THEN
     BEGIN
       Draw(320,180,620,180,1);Draw(320,180,320,90,1);
       zdata:=his_a;
       Avrg_Ed(interval_n,45,true);
       datafile('interval',43);continue:=#59;
     END;
   END
   ELSE Draw(i+10,70,i+10,90,1);
 END;

 PROCEDURE capture_data(paintint,paintfreq:Integer);
 VAR bin_count,bin1,interval,above,data,datant,fin:Integer;
 BEGIN
   bin_count:=0;interval:=0;datant:=0;fin:=1;bin1:=bin*1000;
   IF option='2' THEN BEGIN fin:=0; i:=nBin; END;
   REPEAT i:=i+1;bin_count:=0;above:=0;      { resets variables for new bin }
     WHILE (bin_count<bin1) AND (above<nEvent) DO
       BEGIN
         time_init:=time_tick;              { read ROM-BIOS timer tick count }
         data:=Port[$3FE];                { read serial port status register }
         interval:=interval+1;                      { increment counter one }
         IF interval>299 THEN interval:=300;   { maximum value for interval }
         IF (Abs(data-datant)>10) AND (data>60) THEN
         BEGIN {have port changed and is bigger than 60 then spike in pin 5 }
```

```
          above:=above+1;                                  { count for bin rate }
          his_a[interval]:=his_a[interval]+1; {array for interval histogram}
          Plot(interval+320,180-his_a[interval],paintint);
          interval:=0;                                  { resets interval count }
        END;
        bin_count:=bin_count+fin;          { increment counter two, for bin }
        datant:=data;                    { stores port value for comparation }
        REPEAT time_end:=time_tick;                          { time delay }
        UNTIL time_end<>time_init;
      END;                                                  { end of while }
    interval_n:=interval_n+above;          { total number of events counted }
    data_above[i]:=above;                                  { bin rate array }
    Plot(i+10,180-above,paintfreq)                { graph bin rate }
  UNTIL KeyPressed OR (i>=nBin);
  graph_freq(0,paintfreq);
END;                                                    { end of procedure }

PROCEDURE capture_freqDouble;
VAR bin1,bin_count,data,datant,absdat,above,between:Integer;
BEGIN
  bin_count:=0;bin1:=bin*1000; datant:=0;
  REPEAT
    bin_count:=0;above:=0;between:=0;i:=i+1;
    WHILE bin_count<bin1 DO
      BEGIN
        time_init:=time_tick;              { read ROM-BIOS timer tick count }
        data:=Port[$3FE];                { read serial port status register }
        absdat:=Abs(data-datant);       { actual value minus previous value }
        IF absdat>10 THEN                          { has port value changed }

        BEGIN
          IF data>70 THEN                  { is port value greater than 70 ?}
            BEGIN above:=above+1; between:=between+1; END  { spike in both }
          ELSE BEGIN
            IF data>35                     { is port value greater than 35 ?}
            THEN  above:=above+1                { spike in pin 5 data source }
            ELSE IF data>10 THEN           { is port value greater than 10 ?}
               between:=between+1;            { spike in pin 22 data source }
        END;     END;
        bin_count:=bin_count+1;datant:=data;       { increment counter one  }
        datant:=data;                                      { store port value }
        REPEAT time_end:=time_tick;     { time delay, up to one millisecond }
        UNTIL time_end<>time_init;
      END;                                                  { end of while }
    data_above[i]:=above; data_between[i]:=between;
    Plot(I+10,180-above,3);
    Plot(I+320,180-between,3);
  UNTIL KeyPressed OR (i>=nBin);
  graph_freq(1,3);
END;

PROCEDURE Capture;
BEGIN
  FOR I:=0 TO 320 DO                                    .
    BEGIN                                             { zeroing arrays }
      data_above[i]:=0;data_between[i]:=0;
      his_a[i]:=0;
    END;
  HiRes;
  GoToXY(3,1);Write(' exp ',experiment);
  IF option = '2' THEN Write('  events  ',nEvent)
                  ELSE Write('  bin  ',bin);
  IF (trigger='E') OR (trigger='e') THEN
    REPEAT  i:= Port[$3FE];                      {waiting for signal in pin 22}
    UNTIL   ((i>12) AND (i<30));
  i:=0;interval_n:=0;
  REPEAT
    CASE option OF
      '1':capture_data(0,3);
      '2':capture_data(3,0);
      '3':capture_data(3,3);
      '4':capture_freqDouble;
    END;
  UNTIL (continue=#59) OR (i>=nBin);
END;
```

```
BEGIN
  REPEAT
    Menu;
    Port[$43]:=54;                    { write to 8253 timer chip control port }
    Port[$40]:=Lo(1193);              { program 8253 to tick each millisecond }
    Port[$40]:=Hi(1193);
    capture;
    gotoxy(5,10);
    WriteLn('  continue same = 1; change = 2; finish = Esc  ');
    Read(Kbd,any);
  UNTIL any=#27;
  ClrScr;
END.
```