010 Analogeingang für PCs

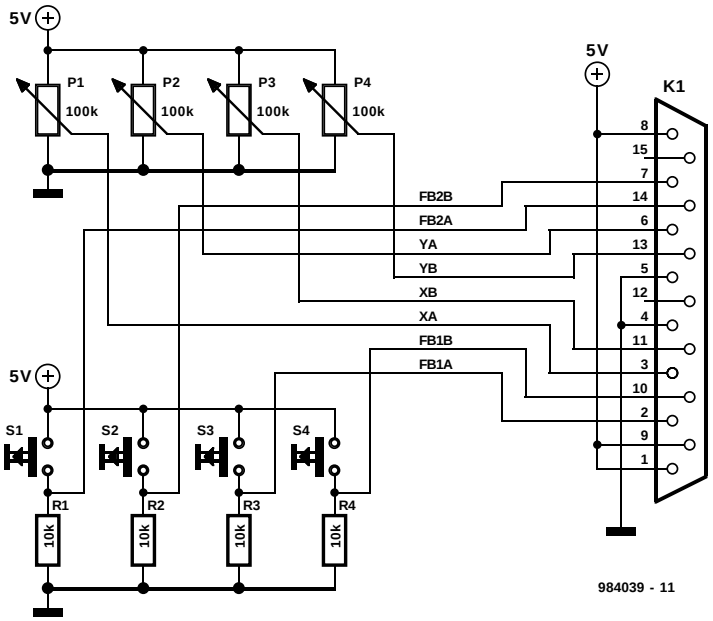
Von G. Scheibe

Die meisten modernen PCs besitzen einen 15poligen Gameport-Anschluß, der vier digitale Eingänge für die Feuerknöpfe und vier analoge Eingänge für die Potis des (analogen) Joysticks. Die Joystick-Eingänge sind mit monostabilen Multivibratoren verbunden, deren On-Zeit von der Stellung der externen 100-kΩ-Potis abhängt, wie dies Bild 1 angedeutet. Von dieser Tatsache kann man Gebrauch machen, indem man statt Potis zum Beispiel Widerstandssensoren wie NTC, PTC oder LDR einsetzt und so den Gameport in ein Meßinterface verwandelt.

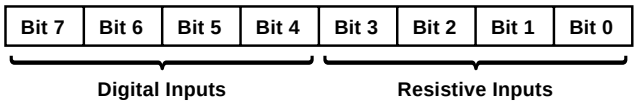
Die Software, die den Gameport liest, kann sehr einfach gehalten werden. Unter Adresse 201_{HEX} ist ein Byte zu finden, dessen vier MSBs (bit 7...4) den Status der vier Feuerknöpfe angeben, während die vier LSBs (bit 3...0) während der Monozeit des jeweiligen Monoflops High sind. Die Software muß nun mit einer schnellen Schleife die Zeit messen, in der ein Bit High ist. Aus der Impulslänge ist der analoge Wert abzuleiten. Je schneller die Schleife arbeitet, desto genauer ist die Messung. Das Listing zeigt ein PASCAL-Programm, mit dem vier analoge Pegel ermittelt werden können.

(984039)rg

1



2




```
{######################## Analog Game Port ###################################
# Example how to use an analog game port as analog input                     #
# Copyright 1998 Segment B.V., Beek, The Netherlands                         #
##############################################################################}

program gametest;

uses crt;

const g_port = $201;          {game port's base address}
      max = 550;              {holds maximum value}
      offset = 50;            {holds minimum value}
      nr = 5;                 {number of samples to average}
{######################## Measurement function ############################}
function measure (var Value: integer; Input, Nr: integer):boolean;
{ 'Value' contains the result of the measurement
  'Input' selects the input channel
  'Nr' determines the number of samples used for averaging}

var i, counter, game : integer;
    bitgame : boolean;
    dummy : longint;

begin
     if ((nr > 100) or (input>4) or (input<1)) then
     begin
          gotoxy(5,20);
          writeln('Error!! Wrong parameter in measurement function');
          measure:=false;
     end
     else
```

```
     begin
          value:=0;
          dummy:=0;
          if input=4 then input:=8;
          if input=3 then input:=4;
          for i:=0 to (Nr-1) do
          begin
               counter:=0;
               bitgame:=true;
               port[g_port]:=0;
               while bitgame do
               begin
                    game:=port[g_port];
                    bitgame:=(((game and input)=input) and
                                   (counter<((max*2)+offset)));

                    counter:=counter+1;
               end;
               counter:=counter-offset;
               dummy:=(dummy + counter);
               delay(1);
          end;
          value:=trunc (dummy/Nr);
          if ((value> (max * 2)) or (value<0)) then
          begin
               if value<0 then value:=-9999
               else value:=9999;
               measure:=false;
          end
          else measure:=true;
     end;
end; {function}

{######################### Input fire button status ##########################}
procedure buttons(var key1, key2, key3, key4: boolean);

{returns boolean values to show fire button status}
{keyx := true if button pressed}
var game : integer;

begin
     game := port[g_port];
     key1:=((game and 128)<>128);
     key2:=((game and 64)<>64);
     key3:=((game and 32)<>32);
     key4:=((game and 16)<>16);
end; {procedure}

{######################### Main Program ######################################}
var connect, i : integer;
    value1 : integer;
    returnm, t1, t2, t3, t4 : boolean;
    e : char;
    value2 : real;

begin
     ClrScr;
     gotoxy(5,3);
     writeln('Analog game port input');
     gotoxy(5,23);
     writeln('Press "e" to interrupt this program');
     gotoxy(1,8);
     writeln ('     INPUT 1:');
     writeln ('     INPUT 2:');
     writeln ('     INPUT 3:');
     writeln ('     INPUT 4:');
     gotoxy(5,15);
     write('digital inputs:');
     gotoxy(12,17);
     write('1:');
     gotoxy(22,17);
     write('2:');
     gotoxy(32,17);
     write('3:');
```

```
        gotoxy(42,17);
        write('4:');
        while e<>'e' do
        begin
             if keypressed then e:=readkey
             else
             begin
                  for i:=0 to 3 do
                  begin
                       returnm := measure(value1, (i+1), Nr);
                       if returnm then
                       begin
                            gotoxy(30,(8+i));
                            write('        ');
                            gotoxy(30,(8+i));
                            write(value1);
                            gotoxy(35,(8+i));
                            write(' number of program loops ');
                       end
                       else
                       begin
                            gotoxy(30,(8+i));
                            if value1<0 then
                               write('----  negative overflow               ')
                            else
                               write('++++  positive overflow               ');
                       end;
                   end;
                   buttons(t1,t2,t3,t4);
                   gotoxy(15,17);
                   if t1 = true then
                      write(' ON')
                   else
                       write('OFF');
                   gotoxy(25,17);
                   if t2 = true then
                      write(' ON')
                   else
                       write('OFF');
                   gotoxy(35,17);
                   if t3 = true then
                      write(' ON')
                   else
                       write('OFF');
                   gotoxy(45,17);
                   if t4 = true then
                      write(' ON')
                   else
                       write('OFF');
             end; {else}
        end; {while}
        ClrScr;
    end.
```

011 Safety Key

Von Peter Lay

Der Safety Key ist ein elektronischer Schlüssel in Form einer diskreten Chipkarte. Die "Chipkarte" besteht dabei aus einer kleinen Platine, auf der sich ein Chip befindet. Wie im Schaltbild zu sehen ist, handelt es sich um ein ganz einfaches Standard-CMOS-IC, nämlich um einen 4001 (IC1), der auf der Platine eine ebenso einfache festverdrahtete Logikschaltung realisiert. Eingänge und Ausgänge dieser Logikschaltung sind auf eine Steckverbindung geführt. Vorzugsweise verwendet man für IC1 eine SMD-Ausführung und für den Steckverbinder ebenfalls eine kompakte Ausführung, so daß die Platine möglichst klein ausfällt. Wenn man die Schaltung in Kunstharz einbettet, erhält man einen kompakten und robusten elektronischen Schlüssel, den man am Schlüsselbund befestigen kann.
So weit der Schlüssel, zu dem