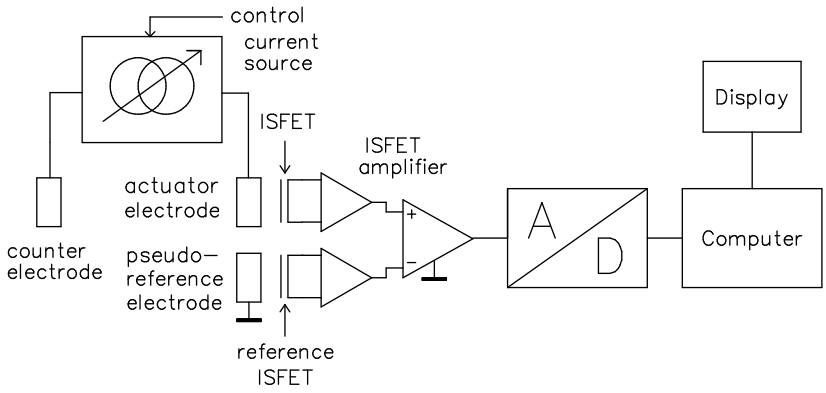
Re-design of the sensor electronics of the coulometric sensor /actuator system

Geert Langereis
march 1995

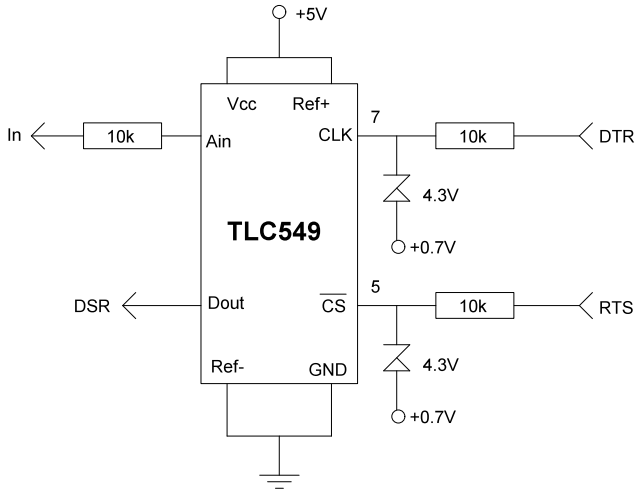


Contents:

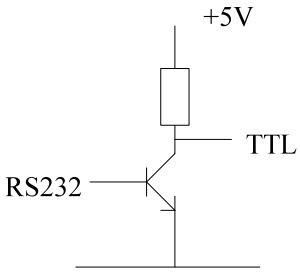
1. The TLC549 AD converter with serial output

The TLC549 is an 8 bit AD converter with a serial output (Datasheet I) and a conversion time of 17µs. The TLC548 is a faster version and the TLC1540 a 10 bit equivalent (21µs).

The connection to the RS232 port can be very simple:



In this set-up the RS232 port is not used as it should be, but the system works on three handshake-lines. The RS232 signals may vary between -12V and +12V so zener-rectifiers are necessary to convert this signals to TTL levels. Notice that the forward potential drop across silicon based diodes is about 0.7, this would result in
a -0.7 volt signal at the CS and CLK inputs of the TLC459 with negative RS232 signals. This is too much according to the specifications (max -0.3 volt) so the diodes must be connected to +0.7 volt. In the experimental set-up this was done by the use of a 190Ω/1kΩ potential devider between 0 and 5 volt. Better RS232 to TTL-level converters are with a transistor:



The connections with the computer port is as follows (the direction is relative to the computer):

	GND	DTR	RTS	DSR
9 - pins sub-D connector	5	4	7	6 *)
25 - pins sub-D connector	7	20	4	6
Direction	-	Out	Out	In

*) The DSR signal is not always on the 9 - pins sub D connector

The handshake pins can directly be accessed in Turbo Pascal:

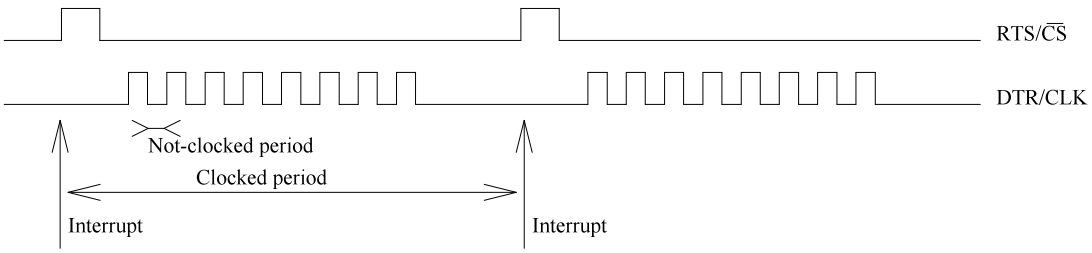
Action:	Pascal command:
RTS off, DTR off	Port [BA+4] := 0;
RTS off, DTR on	Port [BA+4] := 1;
RTS on, DTR off	Port [BA+4] := 2;
RTS on, DTR on	Port [BA+4] := 3;
Read DSR	DSR := ((port [BA+6] **and** 32) = 32)

with BA is the base adress of the serial port and is 3F8H for COM1.

The measure range is here set to 0..5 V by connecting Ref+ and Ref- to ground and Vcc. From the specifications it can be seen that this are the maximal allowed boundaries.

2. An example of a pascal program for controlling the TLC549

This program uses the timer interrupt to synchronize the serial port (RS232). After 300 points are being sampled an oscilloscope image is drawn. The program assumes that the full scale is from 0 to 5 volt. The complete listing is given in appendix B, here is a short functional description.
No second timer is available for timing the sampling routine (according to the specifications of the TLC549 there are some ristrictions to the control signals). The sampling routine may last as long as the time between two sample interrupts. On the slowest possible computer the sampling routine must be fitted to this period.



The used global constants and variables are:

constants
BA = $3F8 : Adress of COM1 port
variables
samp : Stores 300 sampled bytes
counter : Counts how many samples are taken
oldvector : Stores the pointer to the old timer interrupt;
samplefreq: The sample frequency (1kHz by default or else the value after the /f parameter on the command line);
grid : toggles grid on/of (off by default, true if /g parameter is present on the command line);

In pseudo-pascal the sample method is:

program TLC549;

```
    procedure sample;
    begin
        give CS signal to TLC549;
        give 8 CLK signals and read 8 Dout values from TLC549;
        increase counter;
    end;

begin
    Re-direct timer interrupt to procedure "sample" and set the required preload value;
    Wait until the desired number of samples is taken;
    Restore timer interrupt;
    Draw a graph;
end.
```

Description of the procedures:

procedure EGAVGA_driver; **external**;
Is necessary to link the graphics driver which is stored in an object file. The *.OBJ file can be made from the EGAVGA.BGI file using BINOBJ.EXE which is given with Turbo Pascal and Borland Pascal. Command line example:

BINOBJ egavga.bgi egavga.obj EGAVGA_driver

procedure Read_commandline;
Interpretes the command line. There are two options: the /g option sets the grid on, the /f #S option sets the sample frequency to the value #S.

procedure str_to_int(stri:string;**var** intg:integer);
Converts string "stri" to an integer value "intg"

Procedure Init_scoop_beeld;
Initializes the graphics driver, draws the screen including graph and scaling.

Procedure eoi; **inline** ($B0/$20/$E6/$20);
Takes care of the handling of the old interrupt $08 (with interrupt $1C this is probably not necessary).

procedure sample; **interrupt**;
Samples on every interrupt $08 a value from the AD-converter. This is done by using two lines out and one line in:

	Pascal commando	TLC549 pin	RS232 line
Out	port[BA+4] := 2	/CS	RTS
Out	port[BA+4] := 1	CLK	DTR
In	port[BA+6] **and** 32	Dout	DSR

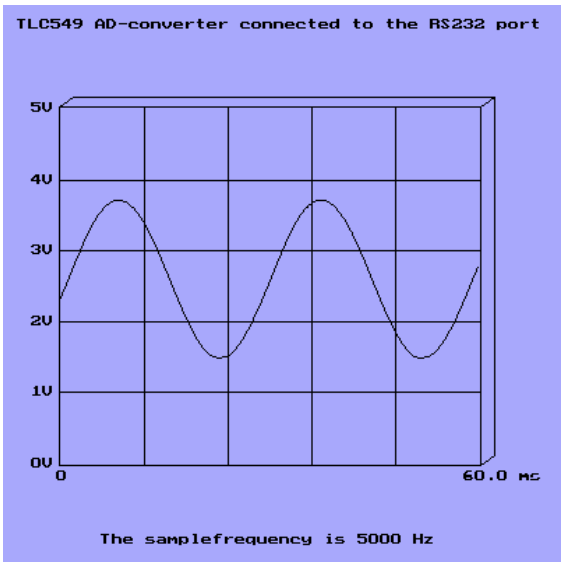
procedure init_interrupt;
Loads the timer register with the accurate preload value to obtain the desired timing (sample frequency). After that the timer interrupt $08 is connected to the own procedure "sample" and the old interrupt vector is stored in the variable "oldvector".

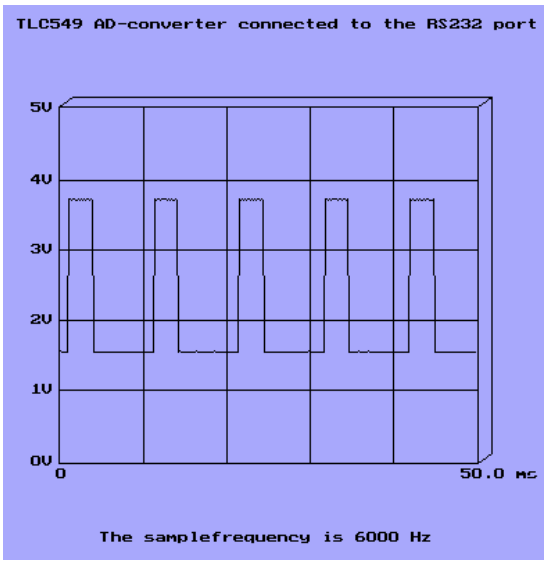
procedure interrupt_done;
Returns the interrupt vector to the original interrupt procedure.

procedure draw_grid;
Draws the grid lines.

procedure Scoop_beeld;
Draws the signal ("samp" variable) in the graph and decides if the grid should be drawn.

Some experiments were performed using a 80286 computer with a clock frequency of 12MHz. No wait states between the rise and fall of the CS and CLK signals were necessary and the program was reliable to a sample frequency of 8kHz (in the sensor/actuator system a sample frequency of 1kHz will be high enough). This was checked using an oscilloscope. The limit was dependent on the computer and was the result of the generation of a second interrupt while the previous interrupt routine was not yet finished. The AD-converter could follow the fastest possible frequency.



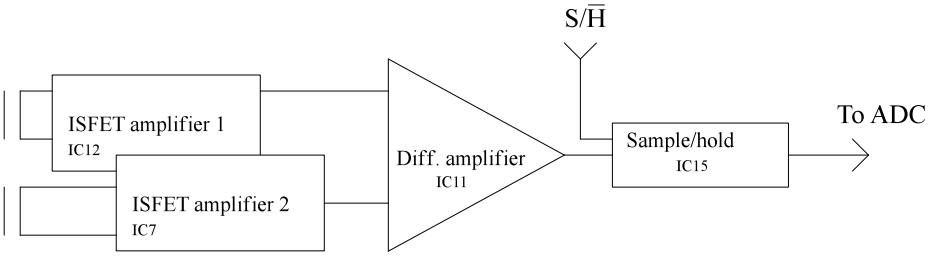


At the left a 35 Hz sine signal sampled with a samplefrequency of 5 kHz. The right figure shows a pulse signal with a pulse period of 10 ms and a pulse duration of 4 ms. Both pictures were made with the program listed in appendix B and grabbed with the GRAB.COM program from WordPerfect 5.1.

3. Connecting to the coulometric sensor/actuator system

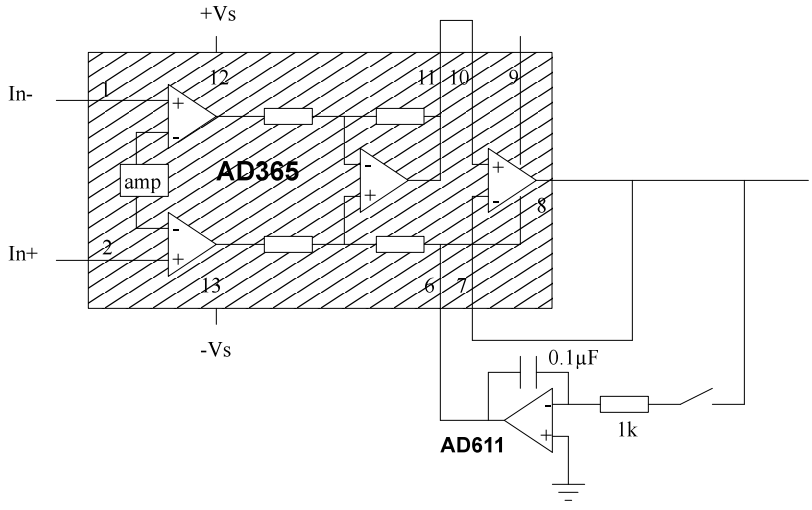
3.1. Using the AD365

In appendix C the old sensor/actuator system is drawn. The sensor system consists of three parts:



The ISFET amplifiers are very well designed, so in first instance I will not change these ones. The other circuitry has a lot of signal conversion (for example to the 0..2.55V window of the old AD converter), this is not a very elegant method. With the new AD converter the window can be set to every range we need by choosing Ref- and Ref+. In the second place there are excellent complete differential amplifier/track and hold devices (analog devices) so it is not convenient to build the sample/hold circuitry ourselves.

The AD365 is a Programmable Gain & T/H DAS Amplifier of Analog Devices (Datasheet III). The data sheet gives an auto-zero circuit which can be used in our system. Here is a simplified version:



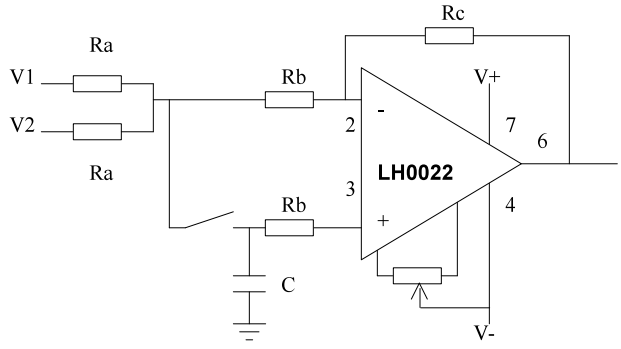
The "amp" block can be set to an amplification of 1, 10, 100 or 500 times. If the ground at the AD611 opamp is set to 2.5 volt, and the range of the output pin 8 is not larger than 2.5 volt then all possible titration curves are projected to a scale of 0..5 volt (the TLC549 input range).

A disadvantage is the extra opamp AD611. If the current through the "sense" (pin 6) input is low this opamp might be omitted however. The charge must remain on the capacitor for as long as the sampling of the whole titration curve (some seconds).

Another disadvantage is the price of the Analog Devices chip (see appendix A).

3.2. Using a modified ISFET amplifier

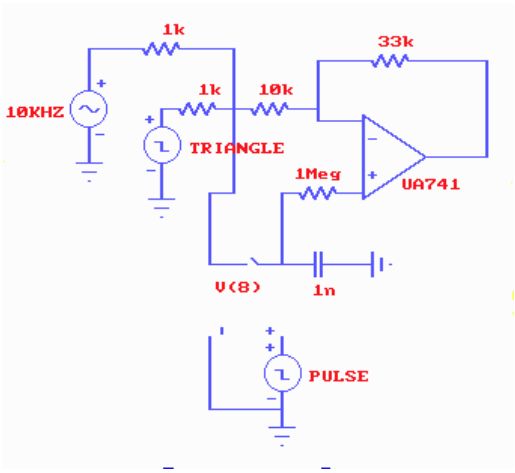
If it is possible to realize a negative version of the ISFET amplifier (one that produces a negative V_{GS} response instead of a positive), it is possible to combine the differential function with the sample and hold with one opamp:



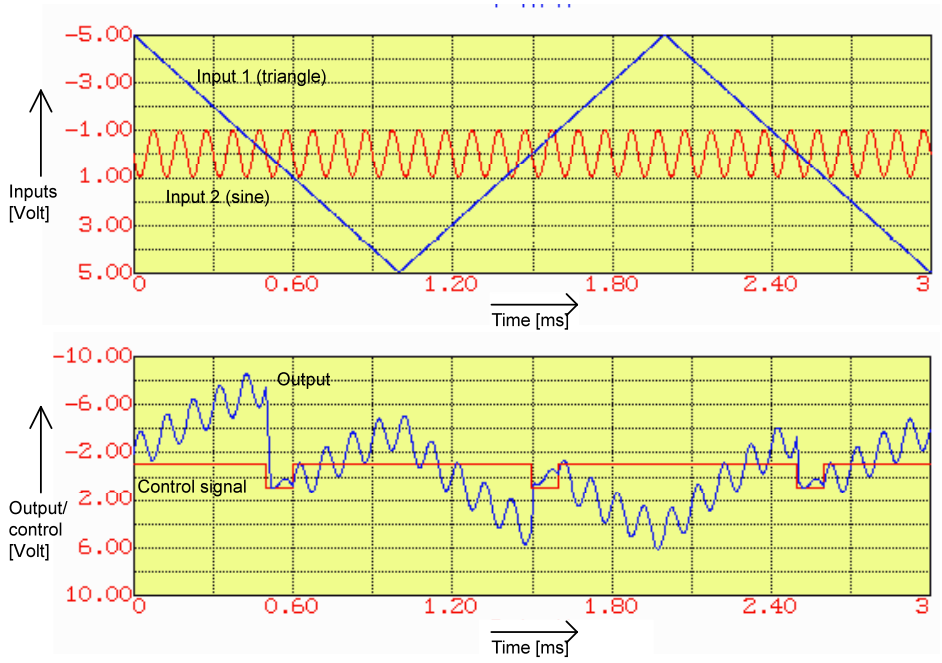
The sample and hold part in this circuitry is taken from "The National Semiconductor Cooperation Linear Applications Databook", application note AN63. The summation part is a standard method. The total output is:

$$V_{out} = V_C - \frac{R_c}{R_{b+} + R_a}\left(V_1 + V_2\right)$$

with V_C the stored voltage on the capacitor. This principle was verified by the use of an electronic simulation program (Microcap III).



The pulse source is used to control the switch. The two input signals are a relative large triangle wave and a small sine wave (top figure). The output signal of the opamp is the sum of those two signals (bottom figure). On each sample/hold pulse, the output signal is set to zero level and starts following the sum signal again.



The amplification is set to -33kΩ/(1kΩ+10kΩ) = -3. The voltage shift (for projecting the output range of this circuitry to the input range of the AD-converter) can be implemented by shifting the capacitor from ground.

3.3. Using a 10 bit AD-converter

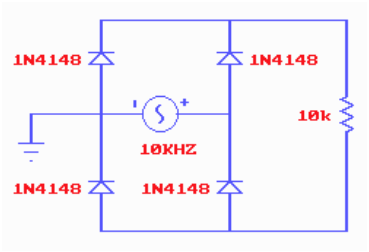
The interesting information of the differential amplifier has a range of less than 472mV on a DC signal of 5V (Diana Siemer and Wim Weultjes 1991, page 46), this is 10%. If the sample/hold part is omitted and we use the TLC1540 (10 bits version of the TLC549), the interesting signal is still stored in 8 bits and the DC signal in the other two bits. To do this, a fast computer is necessary (conversion time 21µs instead of 17µs).

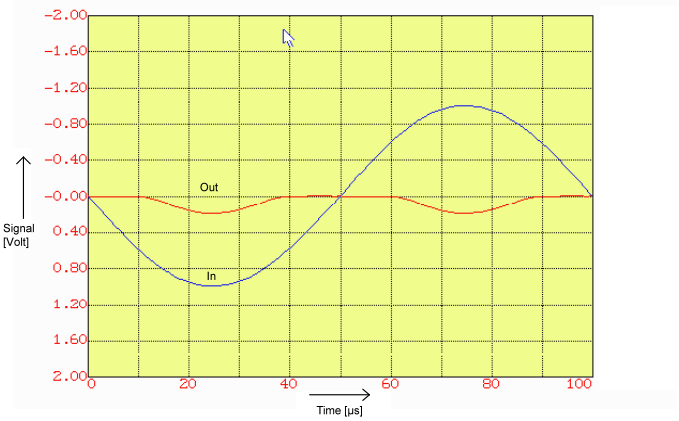
4. A rectifier circuit for using the full 8 bit resolution

The input range of the AD-converter is only half used because it is possible to do a positive or a negative titration (OH^- or H^+ generation). The simplest configuration is to put the starting point of the titration at half full-scale. By this method one bit is lost (used as a sign bit although we already know what polarity the titration will have).

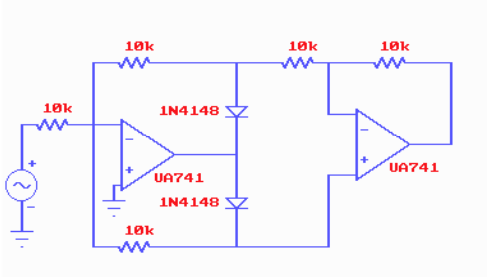
A better solution is to use a rectifier circuit and to add the direction of the titration in the software. The sample and hold circuit defines the starting point as zero and we can titrate in both directions using the full 8 bit resolution of the AD-converter.

A regular diode bridge configuration as drawn below has the problem that a dead area exists because of the threshold voltage of about 0.7 volt. The simulation (transient analysis) and figure are made with Microcap III.

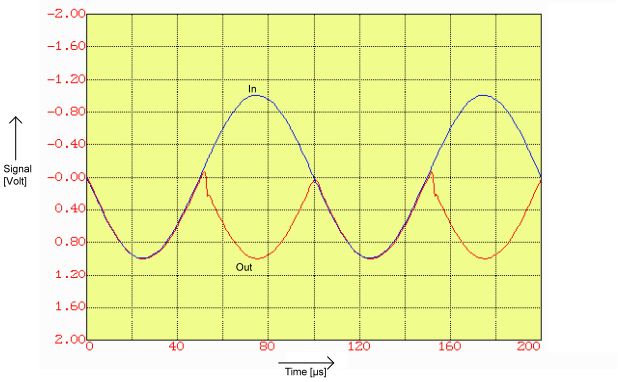


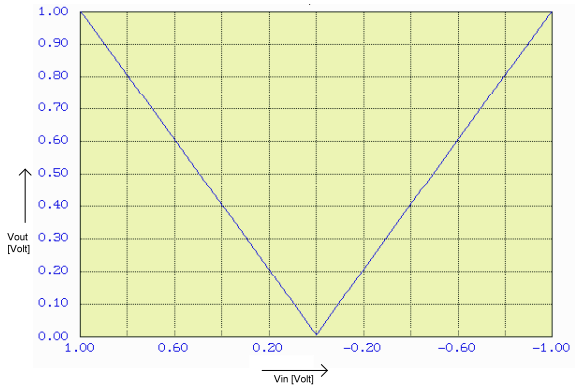


A better solution is obtained with the two-opamp configuration drawn in the next figure:



This configuration has only small diode a-linearities as shown in the transient (left) and DC (right) analysis below.





To my opinion is the noisy response in the transient analysis after a zero-crossing is due to simulation problems in Microcap. If a small a-linearity is left this is no problem in our application because this appears at the beginning of the titration curve which does not contain interesting information.

Appendix A: Some prices of components

These prices are obtained from TEXIM Electronics b.v. in Haaksbergen and are excl. btw.

TLC 549	*f*5,35
AD 365AD	*f*292,-
AD 611	*f*13,43

Appendix B: Listing TLC549 via RS232 Pascal program

```
program TLC549;
(* This program uses the timer interrupt to synchronize the serial port (RS232)   *)
(* The sample frequency is 1kHz by default but can be changed by using             *)
(* the /f option. The frequency must be between 10 Hz and 20 KHz. After 300        *)
(* points are being sampled an oscilloscope image is drawn. With the /g option    *)
(* a grid is viewed.              G.R. Langereis march 1995                        *)

uses dos,crt,graph;
const BA = $3F8;
var  samp       : array[1..300] of byte;
     counter    : integer;
     oldvector  : pointer;
     samplefreq  : integer;
     grid       : boolean;

     procedure EGAVGA_driver; external;
     {$L EGAVGA.OBJ }

     procedure Read_commandline;
     var j : integer;

          procedure str_to_int(stri:string;var intg:integer);
          (* returns the integer value of stri  *)
          (* if stri is valid and 10>stri>20000 *)
          var  i,s : longint;
               k : integer;
          begin
               s := 0; i := 1;
               for k:=length(stri) downto 1 do
               if (ord(stri[k])<58) and (ord(stri[k])>47) then
               begin
                  s := s+(ord(stri[k])-48)*i;
                  i := i*10;
               end;
               if (s<20001) and (s>9) then intg := s;
          end;

     begin
          samplefreq := 1000;  (* initial 1000 Hz *)
          grid := false;   (* initial no grid *)
          for j := 1 to paramcount do
          begin
               if ((paramstr(j)='/f') and (paramcount>j)) then
               str_to_int(paramstr(j+1),samplefreq);
               if (paramstr(j)='/g') then grid := true;
          end;
     end;

     Procedure Init_scoop_beeld;
     var  j, grDriver, grMode : Integer;
          st : string;
     begin
          if RegisterBGIdriver(@EGAVGA_driver) < 0 then WriteLn('Graphics Error');
          GrDriver := Vga;
          GrMode := VGAHi;
          InitGraph(grDriver, grMode,' ');
          setcolor(green);
```

```
        outtextxy(120,80,'TLC549 AD-converter connected to the RS232 port');
        str(samplefreq,st);
        outtextxy(180,450,'The samplefrequency is '+st+' Hz');
        bar3d(150,143,452,401,0,topon);
        outtextxy(130,140,'5V');  outtextxy(130,192,'4V');
        outtextxy(130,243,'3V');  outtextxy(130,294,'2V');
        outtextxy(130,345,'1V');  outtextxy(130,396,'0V');
        outtextxy(148,405,'0');
        if samplefreq>300
        then begin
                str(300*(1000/samplefreq):3:1,st);
                outtextxy(440,405,st+' ms');
        end
        else begin
                str(300/samplefreq:3:1,st);
                outtextxy(440,405,st+' sec');
        end;
        setfillstyle(solidfill,darkgray);
        bar(151,144,451,400);
        for j := 1 to 300 do samp[j] := 0;
        setcolor(white);
end;

procedure eoi; inline ($B0/$20/$E6/$20);

procedure sample; interrupt;
(* Sampled op ieder interrupt $08 een waarde uit de AD-converter*)
var   m,n,i : integer;
        value : byte;
begin
        eoi;
        inc(counter);
        port[BA+4] := 2; (* /CS aan (RTS) *)
        value := 0;
        i := 128;
        port[BA+4] := 0; (* /CS uit (RTS) *)
        for n := 1 to 8 do
        begin
                if (port[BA+6] and 32) = 32 then value := value+i;
                port[BA+4] := 1; (* clock on (DTR) *)
                i := i div 2;
                port[BA+4] := 0; (* clock off (DTR) *)
        end;
        samp[counter] := value;
end;

procedure init_interrupt;
(* Omleiden interrupt $08 *)
begin
        port[$21] := port[$21] or $98;
        port[$43] := 54;
        port[$40] := (Round(1.2e6/samplefreq)) mod 256;
        port[$40] := (Round(1.2e6/samplefreq)) div 256;
        getintvec($8,oldvector);
        getintvec($8,@sample);
end;

procedure interrupt_done;
(* Terugzetten interrupt $08 *)
begin
```

```
        setintvec($8,oldvector);
        port[$43] := 54;
        port[$40] := 0;
        port[$40] := 0;
        port[$21] := port[$21] and $67;
    end;

    procedure draw_grid;
    begin
        setcolor(green);
        line(151,349,451,349);   line(151,298,451,298);
        line(151,247,451,247);   line(151,196,451,196);
        line(211,144,211,400);   line(271,144,271,400);
        line(331,144,331,400);   line(391,144,391,400);
        setcolor(white);
    end;

    procedure Scoop_beeld;
    var j : integer;
    begin
        bar(151,144,451,400);
        if grid then draw_grid;
        for j := 2 to 300 do line(149+j,400-samp[j-1],150+j,400-samp[j]);
    end;

begin
    Read_commandline;
    Init_scoop_beeld;
    repeat
    counter := 0;
    init_interrupt;
    while counter<300 do;
    interrupt_done;
    Scoop_beeld;
    until keypressed;
    CloseGraph;
end.
```

Appendix C: The old Coulometric sensor/actuator system

Datasheet I: The TLC549

From:
Texas Instruments,
Linear circuits databook, vol 2 (1989)
Available at Ed Droogs' room

Datasheet II: The TLC1540

Datasheet III: The AD365