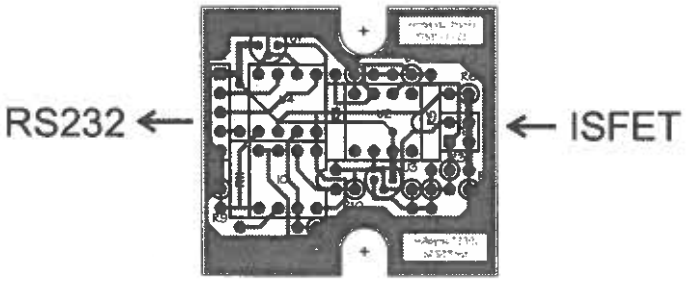
ISFET To RS232 converter

Geert Langereis
March 1997



Description

The ISFET to RS232 converter uses a simple, low power version of the ISFET amplifier which output is being sampled using a 12 bits AD converter with serial output. The serial output can directly be clocked into a computer using the standard RS232 port. Because the whole system uses about 8 mA only, its power supply can be acquired from the same computer port. For convenience, the ISFET to RS232 converter is realized into a standard RS232 connector package.

Specifications

AD converter		
Resolution	12	bits
	1.22	mV
Range	0 .. 5	Volt
ISFET amplifier		
Potential of reference electrode versus AD converter window *)	3.65	Volt
Total system		
Supply current	8	mA
Resolution (at 59 mV/pH)	0.02	pH
Costs of materials	≈ 60	NLG
Software		
Operating system	DOS	
Used language	Pascal	
Displayed precision	0.1	pH
Calibration	1 or 2	Points
Sample rate (on 40 MHz 80386 PC)	≈10	Hz

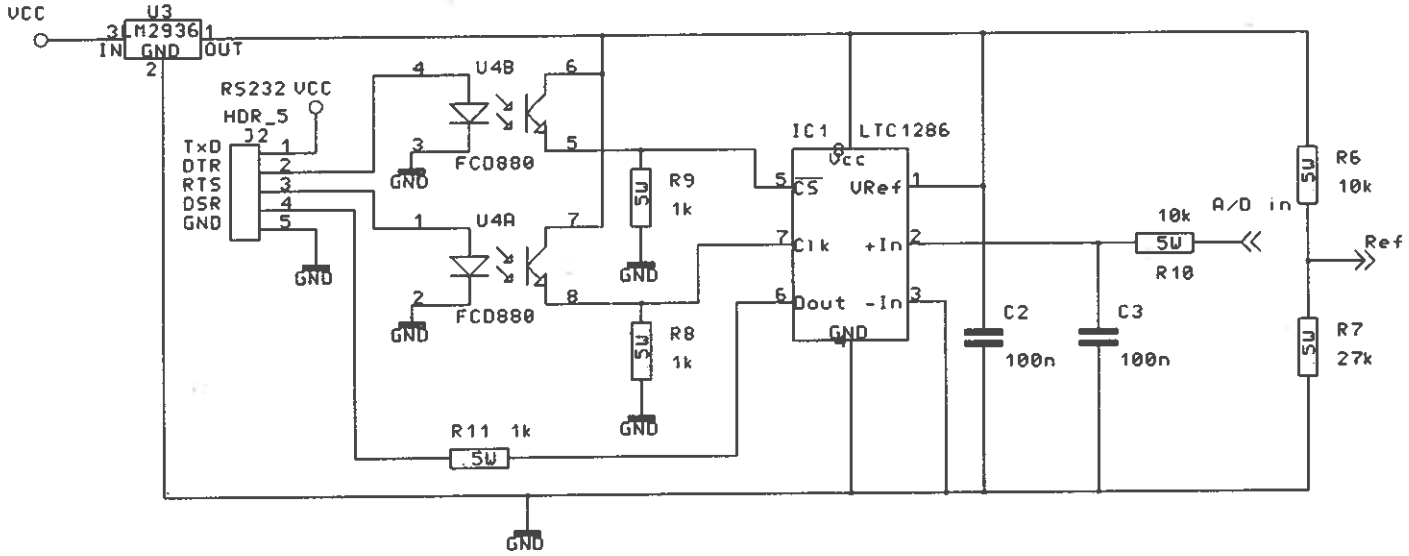
*) Because the reference electrode is not connected to the computer ground and the circuitry is not electrically insulated from the computer, the use with other electrochemical equipment in the same sample solution might cause problems.

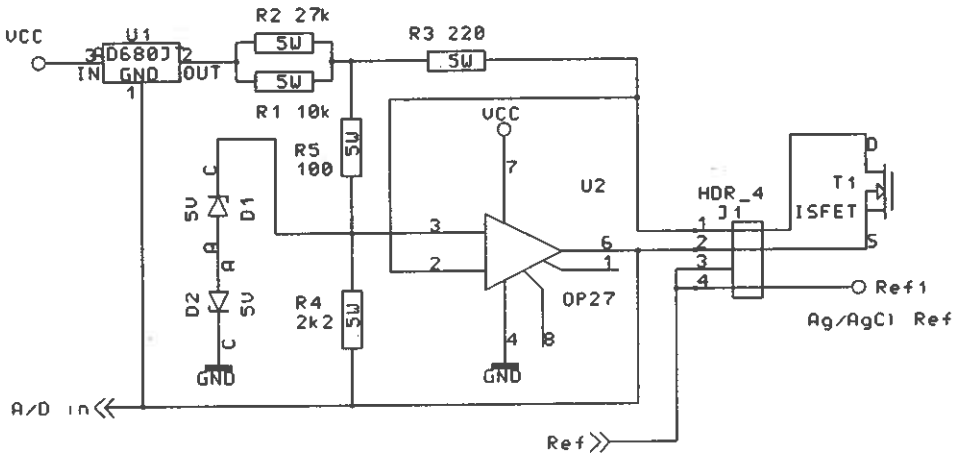
Acknowledgment

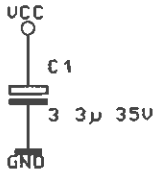
Some helpful tips concerning the electronic circuitry were given by Ad Sprenkels

Appendices

- Schematic of prototype
- Source listing of Turbo Pascal program
- First page of data sheet of TLC1286 AD converter







Rev: 3	Eng: Geert Langereis
Project: ISFET to RS232 converter	
Company: University of Tuente Adress: P O Box 217 City: 7500 AE Enschede Country: The Netherlands	
Date: 18 March 1997	Page: 1 of: 1

```
program ISFET;

uses dos,crt;
const
  COM1=$3F8;  COM2=$2F8;
  fullscale=5;
  Calibration_filename='ISFET.DAT';

  (* Location on screen        *)
  org_X=10;   org_Y=3;

  (* Defaults                  *)
  default_pH_Slope=-59;
  default_pH_Offset=3.500;
  default_COM=COM2;
  default_wait=100;

  (* pH references             *)
  pH_ref1=1.68;
  pH_ref2=7.00;

  (* Colors *)
  textcol=white;   backcol=blue;
  textcol2=white;  backcol2=lightgray;
  LED_OK=lightgreen;
  LED_OL=lightred;

var
  pH_Offset,pH_Slope:real; (* ISFET calibration data    *)
  BA:word;                 (* Base address of COM port  *)
  Show      : byte;          (* Show pH or Volts    *)

(************ Routines for sampling ************)

var wait:integer;

function AD:word;
var m,n,i :integer;
    value :word;
begin
(* Initiate data transfer by a /CS pulse *)
  port[BA+4]:=2; {/CS aan (RTS)
  for m:=1 to wait do;
  port[BA+4]:=0; {/CS uit (RTS)
  for m:=1 to wait do;
  value:=0;
  (* Read three dummy bits: see data sheet *)
  for n:=1 to 3 do
  begin
    port[BA+4]:=1; {clock aan (DTR)}
    for m:=1 to wait do;
    port[BA+4]:=0; {clock uit (DTR)}
    for m:=1 to wait do;
  end;
  (* Transfer of 12 bits *)
  i:=4*1024;
  for n:=1 to 12 do
  begin
    port[BA+4]:=1; {clock aan (DTR)}
    for m:=1 to wait do;
    i:=i div 2;
    if (port[BA+6] and 32) = 32 then value:=value+i;
    port[BA+4]:=0; {clock uit (DTR)}
    for m:=1 to wait do;
  end;
  AD1:=value;
end;
```

```
(* ****** Routines for the graphical user interface ********)

procedure Show_Help;
begin
  clrscr;
  highVideo; Writeln('       ISFET Amplifier for RS232');
  normvideo; writeln('              Version 1.0');
             Writeln('         Geert Langereis 1997 ');writeln;
  highvideo; writeln(' /?   : Show help ');
             writeln(' /C x : Use serial port');
  normvideo; writeln('         x = 1 : COM1');
             writeln('         x = 2 : COM2 [default] ');
  highvideo; writeln(' /S x : Set ISFET slope');
  normvideo; writeln('         0.0 < x < 100.0 [mV/dec]');
             writeln('         The default is read from file ISFET.DAT');
  highvideo; writeln(' /O x : Set ISFET OFFSET slope');
  normvideo; writeln('         -2.000 < x < 2.000 [Volt]');
             writeln('         The default is read from file ISFET.DAT');
  highvideo; writeln(' /B x : Bus timing');
  normvideo; writeln('         0 < x < 1000 : use loop');
             writeln('         The default is /B 100');
             writeln;
end;

procedure CursorOff;
var
  Regs    : registers;
  CSL,CEL : byte;
begin
  Regs.AH:=$0F;
  intr($10,Regs);
  Regs.AH:=$03;
  intr($10,Regs);
  CSL:=REGS.CH;
  CEL:=REGS.CL;
  with Regs do
  begin  AH:=$01; CH:=$20;  CL:=$20; end;
  intr($10,Regs);
end;

procedure CursorOn;
var
  Regs    : registers;
  CSL,CEL : byte;
begin
  with Regs do
  begin AH:=$01; CH:=CSL; CL:=CEL; end;
  intr($10,Regs);
end;

Procedure Init_scherm;
begin
  textcolor(textcol);
  textbackground(backcol);
  gotoxy(org_X,org_Y);   write('
  gotoxy(org_X,org_Y+1); write('
  gotoxy(org_X,org_Y+2); write('
  gotoxy(org_X,org_Y+3); write('
  gotoxy(org_X,org_Y+4); write('
  gotoxy(org_X,org_Y+5); write('
  gotoxy(org_X,org_Y+6); write('
  gotoxy(org_X,org_Y+7); write('
  gotoxy(org_X,org_Y+8); write('
  gotoxy(org_X,org_Y+9); write('
  gotoxy(org_X,org_Y+10);write('
  gotoxy(org_X,org_Y+11);write('  1: Single point calibration   | a: Show AD-converter output
  gotoxy(org_X,org_Y+12);write('  2: Two points calibration     | p: Show pH
  gotoxy(org_X,org_Y+13);write('  s: Save calibration data      | q: Exit
  gotoxy(org_X,org_Y+14);write('
end;
```

```
Procedure Init_scherm2;
begin
  textcolor(textcol2);
  textbackground(backcol2);
  gotoxy(org_X,org_Y);    write('┌──────────────────────────────────────────────┬───┬───┬───┐');
  gotoxy(org_X,org_Y+1);  write('│                                              └───┴───┴───┤');
  gotoxy(org_X,org_Y+2);  write('│                                                          │');
  gotoxy(org_X,org_Y+3);  write('│                                                          │');
  gotoxy(org_X,org_Y+4);  write('│                                                          │');
  gotoxy(org_X,org_Y+5);  write('│                                                          │');
  gotoxy(org_X,org_Y+6);  write('│                                                          │');
  gotoxy(org_X,org_Y+7);  write('│                                                          │');
  gotoxy(org_X,org_Y+8);  write('│                                                          │');
  gotoxy(org_X,org_Y+9);  write('│                                                          │');
  gotoxy(org_X,org_Y+10);write('│    AD-Converter:        V    Slope:        mV/pH         │');
  gotoxy(org_X,org_Y+11);write('│    pH:                       Offset:       V             │');
  gotoxy(org_X,org_Y+12);write('├─────────────────────────────┬────────────────────────────┤');
  gotoxy(org_X,org_Y+13);write('│ a: Accept                   │ c: Cancel                  │');
  gotoxy(org_X,org_Y+14);write('└─────────────────────────────┴────────────────────────────┘');
end;

procedure writechar(x,y:word;i:word);
var j:word;
begin
  case i of
  0:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('█    █');
      gotoxy(x,y+3);write('█    █');
      gotoxy(x,y+4);write('█    █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  1:begin
      gotoxy(x,y);  write('      ');
      gotoxy(x,y+1);write('   ▀█ ');
      gotoxy(x,y+2);write('    █ ');
      gotoxy(x,y+3);write('    █ ');
      gotoxy(x,y+4);write('    █ ');
      gotoxy(x,y+5);write('    █ ');
      gotoxy(x,y+6);write('    ▀ ');
    end;
  2:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('     █');
      gotoxy(x,y+3);write('   ▄▀ ');
      gotoxy(x,y+4);write(' ▄▀   ');
      gotoxy(x,y+5);write('█     ');
      gotoxy(x,y+6);write('▀▀▀▀▀▀');
    end;
  3:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('     █');
      gotoxy(x,y+3);write('   ██ ');
      gotoxy(x,y+4);write('     █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  4:begin
      gotoxy(x,y);  write('█    █');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('█    █');
      gotoxy(x,y+3);write('▀▀▀▀▀█');
      gotoxy(x,y+4);write('     █');
      gotoxy(x,y+5);write('     █');
      gotoxy(x,y+6);write('     ▀');
    end;
  5:begin
      gotoxy(x,y);  write('█▀▀▀▀▀');
      gotoxy(x,y+1);write('█     ');
      gotoxy(x,y+2);write('█     ');
      gotoxy(x,y+3);write(' ▀▀▀▄ ');
      gotoxy(x,y+4);write('     █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  6:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('█     ');
      gotoxy(x,y+3);write('█▀▀▀▀▄');
      gotoxy(x,y+4);write('█    █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  7:begin
      gotoxy(x,y);  write('▀▀▀▀▀█');
      gotoxy(x,y+1);write('     █');
      gotoxy(x,y+2);write('    █ ');
      gotoxy(x,y+3);write('   █  ');
      gotoxy(x,y+4);write('  █   ');
      gotoxy(x,y+5);write('  █   ');
      gotoxy(x,y+6);write('  ▀   ');
    end;
  8:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('█    █');
      gotoxy(x,y+3);write(' ████ ');
      gotoxy(x,y+4);write('█    █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  9:begin
      gotoxy(x,y);  write(' ▄▄▄▄ ');
      gotoxy(x,y+1);write('█    █');
      gotoxy(x,y+2);write('█    █');
      gotoxy(x,y+3);write(' ▀▀▀▀█');
      gotoxy(x,y+4);write('     █');
      gotoxy(x,y+5);write('█    █');
      gotoxy(x,y+6);write(' ▀▀▀▀ ');
    end;
  10:begin
      gotoxy(x,y+2);write('█     █');
      gotoxy(x,y+3);write(' █   █ ');
      gotoxy(x,y+4);write('  █ █  ');
      gotoxy(x,y+5);write('   █   ');
      gotoxy(x,y+6);write('       ');
    end;
  11:begin
      gotoxy(x,y);  write('              ');
      gotoxy(x,y+1);write('              ');
      gotoxy(x,y+2);write('       █    █ ');
      gotoxy(x,y+3);write('▄▄▄▄▄  █    █ ');
      gotoxy(x,y+4);write('█    █ █▀▀▀▀█ ');
      gotoxy(x,y+5);write('█▀▀▀▀  █    █ ');
      gotoxy(x,y+6);write('█      ▀    ▀ ');
    end;
  12:for j:=0 to 6 do begin gotoxy(x,y+j); write('      '); end;
  13:begin
      for j:=0 to 6 do begin gotoxy(x,y+j); write('      '); end;
      gotoxy(x,y+4);write(' ▀▀▀▀ ');
    end;
  end;
end;
```

```
procedure ShowMeet(dat:word);
var d1,d2,d3,d4,ol : word;
    pH,Volt        : real;

    procedure show_LEDS;
    begin
      gotoxy(org_X+47,org_Y+1);write('   |    |   ');
      case ol of
      0:begin
          textcolor(LED_OL);
          gotoxy(org_X+47,org_Y+1);
          write('■');
          textcolor(textcol);
        end;
      1:begin
          textcolor(LED_OL);
          gotoxy(org_X+47+10,org_Y+1);
          write('■');
          textcolor(textcol);
        end;
        else begin
          textcolor(LED_OK);
          gotoxy(org_X+47+5,org_Y+1);
          write('■');
          textcolor(textcol);
        end;
     end;
    end;

    procedure show_AD;
    begin
      d1:=trunc(Volt);
      d2:=trunc((Volt-d1)*10);
      d3:=trunc((Volt-d1-0.1*d2)*100);
      d4:=trunc((Volt-d1-0.1*d2-0.01*d3)*1000);
      writechar(org_X+2,org_Y+2,12);
      writechar(org_X+10,org_Y+2,d1);
      gotoxy(org_X+17,org_Y+8);write('■');
      writechar(org_X+19,org_Y+2,d2);
      writechar(org_X+27,org_Y+2,d3);
      writechar(org_X+35,org_Y+2,d4);
      writechar(org_X+41,org_Y+2,10);
    end;

    procedure show_pH;
    begin
      case ol of
      0:begin d1:=13; d2:=9; d3:=9; end;
      1:begin d1:=1; d2:=4; d3:=0; end;
      else
        if pH<0 then d1:=13
                else if pH>=10 then d1:=1 else d1:=12;
        d4:=trunc(abs(10*pH));
        d2:=(d4 mod 100) div 10;
        d3:=d4 mod 10;
      end;
      writechar(org_X+2,org_Y+2,d1);
      writechar(org_X+10,org_Y+2,d2);
      gotoxy(org_X+17,org_Y+8);write('■');
      writechar(org_X+19,org_Y+2,d3);
      writechar(org_X+27,org_Y+2,11);
    end;

    procedure show_both;
    begin
      gotoxy(org_X+20,Org_Y+10);    write(Volt:1:3);
      gotoxy(org_X+21,Org_Y+11);    write(pH:1:1);
      gotoxy(org_X+41,Org_Y+10);    write(pH_Slope:1:1);
      gotoxy(org_X+40,Org_Y+11);    write(pH_Offset:1:3);
    end;
```

```
begin (* ShowMeet *)
  Volt:=(dat/4096)*fullscale;
  pH:=1000*(Volt-pH_Offset)/pH_slope;
  if (dat=4096-1) or (pH<=-10) then ol:=0 else
  if (dat=0) or (pH>14) then ol:=1 else ol:=2;
  show_LEDs;
  case show of
  1:show_pH;
  2:show_AD;
  else show_both;
  end;
end;

{********* Routines for the menu options **********}

procedure single_point_cal;
var command:char;
    meet:word;
begin
  Init_scherm2;
  command:='n';
  gotoxy(org_X+15,org_Y+5);write('Put probe in test buffer of pH ',pH_ref1:2:2);
  show:=0; (* Show both pH and AD *)
  repeat
    if keypressed then command:=readkey;
    meet:=AD;
    showmeet(meet);
    delay(100);
  until (command='c') or (command='a');
  if (command='a') and (meet<>0) and (meet<>(4096-1))
  then pH_Offset:=meet*(fullscale/4096)-pH_ref1*(pH_Slope/1000);
  Init_Scherm;
  SHow:=1;
  showmeet(AD);
end;

procedure dual_point_cal;
var command:char;
    meet,m1,m2:word;
begin
  Init_scherm2;
  command:='n';
  show:=0; (* Show both pH and AD *)
  gotoxy(org_X+15,org_Y+5);write('Put probe in test buffer of pH ',pH_ref1:2:2);
  repeat
    if keypressed then command:=readkey;
    meet:=AD;  showmeet(meet);   delay(100);
  until (command='c') or (command='a');
  if (command='a') and (meet<>0) and (meet<>(4096-1))
  then
  begin
   m1:=meet;
   Init_scherm2;
   command:='n';
   gotoxy(org_X+15,org_Y+5);write('Put probe in test buffer of pH ',pH_ref2:2:2);
   repeat
     if keypressed then command:=readkey;
     meet:=AD;     showmeet(meet);     delay(100);
   until (command='c') or (command='a');
   if (command='a') and (meet<>0) and (meet<>(4096-1))
   then begin
     m2:=meet;
     pH_Slope:=(1000/4096)*((m1-m2)*fullscale)/(pH_ref1-pH_ref2);
     pH_Offset:=m1*(fullscale/4096)-pH_ref1*(pH_Slope)/1000
   end;
  end;
  Init_Scherm;
  Show:=1;
  showmeet(AD);
end;
```

```
procedure save_data;
var F:text;  st:string;
begin
  Assign(F,calibration_filename);
  Rewrite(F);
  str(pH_Slope,st);    writeln(F,st);
  str(pH_Offset,st);   writeln(F,st);
  Close(F);
end;

{****************** Initilization ******************}

var help : boolean;

procedure UnInitialize; far;
begin
  port[BA+3]:=0; {* power_off *}
  NormVideo;     {* Selects the original text attributes *}
  clrscr;
  cursoron;
  writeln('Normal exit');
end;

procedure Initialize;

  procedure Read_Commandline;
  var st     : string;
      i      : word;
      j,code : integer;
      jj     : real;
  begin
    help:=false;
    for i:=1 to paramcount do
    begin
      st:=paramstr(i);
      if st[1]='/' then
      case st[2] of
       'c': begin inc(i); if paramstr(i)='1' then BA:=COM1 else BA:=COM2;end;
       's': begin inc(i); Val(ParamStr(i), jj, Code); if code=0 then pH_slope:=jj end;
       'o': begin inc(i); Val(ParamStr(i), jj, Code); if code=0 then pH_offset:=jj end;
       'b': begin inc(i); Val(ParamStr(i), j, Code);  if code=0 then wait:=j; end;
       '?': help:=true;
      end
      else help:=true;
    end;
    wait:=abs(wait);
  end;

  procedure Get_Params;
  var F     : text;
      error : integer;
      stri  : string;
      dummy : real;
  begin
   {* Set defaults *}
    pH_Slope:=Default_pH_Slope;
    pH_Offset:=Default_pH_Offset;
    BA:=default_COM;
    Wait:=Default_wait;
    {$I-} {* Load calibration data, if available *}
    Assign(F,calibration_filename);   Reset(F);
    if (IOResult = 0) then
    begin
      readln(F,stri);
      val(stri,dummy,error);
      if error=0 then pH_slope:=dummy;
      readln(F,stri);
      val(stri,dummy,error);
      if error=0 then pH_Offset:=dummy;
      Close(F)
    end;   {$I+}
    Read_Commandline;
  end;

begin {* Initialize *}
  Get_params;
  if Help then begin show_help; halt(1); end;
  Exitproc:=@UnInitialize;
  clrscr;
  cursoroff;
  init_scherm;
  Show:=1;  {* Show pH *}
  port[BA+3]:=64; {* Power on *}
end;

{**************** Main program *************}

procedure hand'e_event(var stop:boolean);
var command:char;
begin
  command:=readkey;
  case command of
   '1':single_point_cal;
   '2':dual_point_cal;
   's':save_data;
   'a':Show:=2;
   'p':Show:=1;
  end;
  stop:=(command='q');
end;

var
  stop : boolean;
  samp : word;

begin
  Initialize;
  stop:=false;
  while not stop do
  begin
    samp:=AD;
    if keypressed then handle_event(stop);
    showmeet(samp);
    delay(100);
  end;
end.
```

LINEAR TECHNOLOGY

LTC1286/LTC1298

Micropower Sampling 12-Bit A/D Converters In SO-8 Packages

FEATURES

- **12-Bit Resolution**
- **8-Pin SOIC Plastic Package**
- **Low Cost**
- **Low Supply Current: 250µA Typ.**
- Auto Shutdown to 1nA Typ.
- Guaranteed ±3/4LSB Max DNL
- Single Supply 5V to 9V Operation
- On-Chip Sample-and-Hold
- 60µs Conversion Time
- Sampling Rates:
  - 12.5 ksps (LTC1286)
  - 11.1 ksps (LTC1298)
- I/O Compatible with SPI, Microwire, etc.
- Differential Inputs (LTC1286)
- 2-Channel MUX (LTC1298)
- 3V Versions Available: LTC1285/LTC1288

APPLICATIONS

- Battery-Operated Systems
- Remote Data Acquisition
- Battery Monitoring
- Handheld Terminal Interface
- Temperature Measurement
- Isolated Data Acquisition

DESCRIPTION

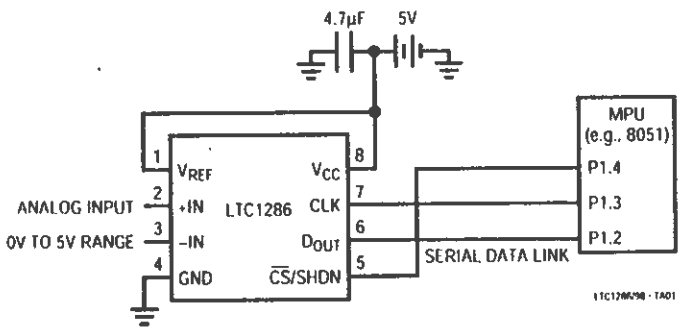
The LTC1286/LTC1298 are micropower, 12-bit, successive approximation sampling A/D converters. They typically draw only 250µA of supply current when converting and automatically power down to a typical supply current of 1nA whenever they are not performing conversions. They are packaged in 8-pin SO packages and operate on 5V to 9V supplies. These 12-bit, switched-capacitor, successive approximation ADCs include sample-and-holds. The LTC1286 has a single differential analog input. The LTC1298 offers a software selectable 2-channel MUX.

On-chip serial ports allow efficient data transfer to a wide range of microprocessors and microcontrollers over three wires. This, coupled with micropower consumption, makes remote location possible and facilitates transmitting data through isolation barriers.

These circuits can be used in ratiometric applications or with an external reference. The high impedance analog inputs and the ability to operate with reduced spans (to 1.5V full scale) allow direct connection to sensors and transducers in many applications, eliminating the need for gain stages.

TYPICAL APPLICATIONS

25µW, SO-8 Package, 12-Bit ADC
Samples at 200Hz and Runs Off a 5V Supply



Supply Current vs Sample Rate