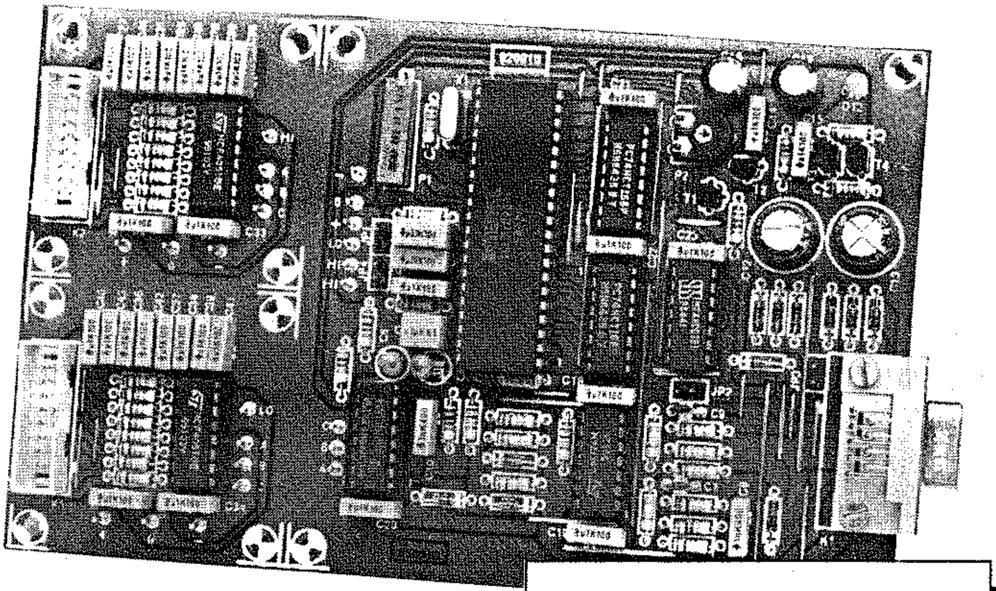
AD232 CONVERTER



The well-known ICL7106 A-D converter IC has been with us in many generations of digital multimeter. A special version of it, the ICL7109, offers the same accuracy, comes at a low price also, but sports a parallel interface instead of a 7-segment driver. The ICL7109 requires only a handful of components to build a versatile measuring box with an RS232 interface that is easy to manage thanks to some hardware tricks. Software for IBM PCs and compatibles is available to control this unit, which is basically a multiplexed A-D converter with an RS232 interface.

MAIN SPECIFICATIONS

- For every PC with a serial port
- 16-channel input multiplexer
- 12-bit A-D converter
- Powerful graphics-based menu-driven control software
- Based on inexpensive ICs
- Powered by PC

Design and software by Ing. B.C. Zschocke and A. Arnold

THE concept of the circuit is illustrated in the block diagram in Fig. 1. At the left are three input blocks with low-pass filter characteristics, marked HI', HI and LO. The HI and LO blocks represent eight input channels, each of which can be selected individually by the block marked 'select'. The part behind the input selection circuitry allows switching between the HI' (direct) input and the switchable inputs.

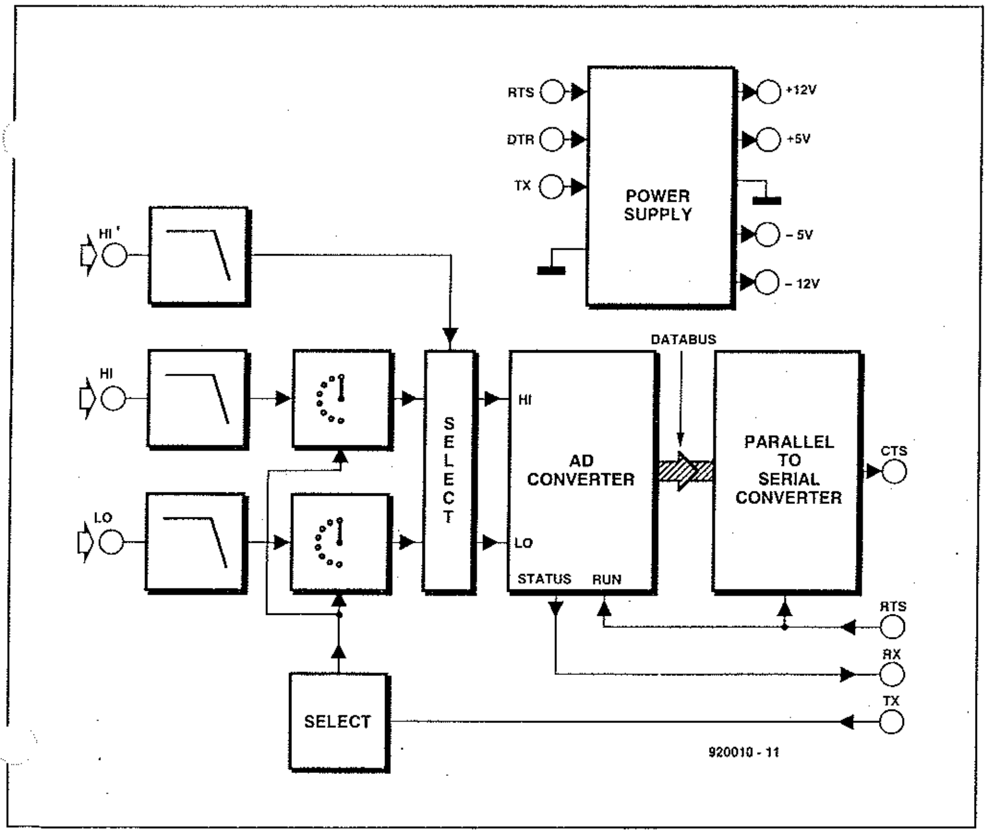


Fig. 1. Block diagram of the RS232-controlled A-D converter.

The A-D (analogue-to-digital) converter behind the input circuitry supplies its digital output data to a parallel-to-serial converter via a parallel bus. Unusually, the output information of the format converter is connected the CTS (clear to send) handshaking line of the RS232 port, rather than the RxD (received data) line. This arrangement results in a much simpler control of the measuring box than would be possible if the RxD line were used.

The circuit is powered by the RS232 port on the PC. The power supply block shown in the diagram provides a regulated output voltage of ±5 V for the converter, and ±12 V (approximately) for the serial interface.

Connection problems

The serial (RS232) interface is used here because the parallel (Centronics) interface can not furnish enough current to power the RS232 A-D converter. Furthermore, in most cases it is easier to find a free serial port on a PC than a free Centronics port, while the use of longer cables is also a boon. On the down side, a serial link is much slower than a parallel link, so that relatively slow ADCs can be used only. Fortunately, this is not always a disadvantage, because slow but very accurate ADCs are inexpensive and widely available. Also, it is not always necessary to sample slowly varying measuring quantities at kHz rates.

The ICL7109 ADC used here differs from the ubiquitous ICL7106 by the digital interface only. The input circuitry is almost identical for both ICs.

Interface parameters can be a real problem when implementing an RS232 link and keeping to the standards. For instance, the link will work correctly only if the transmitter and the receiver are set to the same data format (number of data bits and start/stop bits), but also to the same data rate (in bits per second, or baud). Many of you will know the difficulties and frustrations in setting up a serial link, with obstacles such as DIP switches, configuration files, MODE commands, and long series of parameters.

Here, the RS232 link is used in a non-standard way to avoid some of the problems mentioned above. Data is conveyed via CTS, and clocked by another handshaking line, RTS (ready to send). Four good reasons can be given for this choice:

- The control software determines when and how fast data is conveyed. It is not necessary to configure the serial port. This is particularly attractive when the interface is used for several different peripherals.
- Conveying data via a 'standard' RS232 link requires either special interfacing circuits with internal or external oscillators, or complex discrete alternatives.
- The AD232 board can be linked to interfaces that are not fully RS232 compatible, or directly to a microcontroller.
- The component count is quite low.

Details

The circuit diagram is given in Fig. 2. A quadruple opamp, IC6, forms the RS232 interface. Circuits IC4, IC7 and IC8 form a simple input multiplexer. IC1 is the ADC proper. The two 74HCT166s, IC2 and IC3, form a 16-bit shift register for the parallel-to-serial conversion. The four XOR gates in the 74HC02 package, IC5, control the timing of the A-D conversion, and generate a clock signal for the shift registers.

The power supply takes its input voltage from the RS232 interface in the PC, and provides the stabilized ±5 V rails for the converter circuitry. The analogue-to-digital conversion starts when a level transition is detected at the RTS input. Opamp IC6a makes the RTS signal TTL compatible, and its output signal causes bistable IC5c-IC5d to toggle. The high level at pin 10 of IC5 triggers the A-D conversion in IC1. When the conversion result is available, the ICL7109 supplies a $\overline{\text{LOAD}}$ signal to the shift register, IC2, via gates IC5a and IC5b. First, the high (most significant) byte is loaded into IC2, then the low (least significant) byte into IC3. The signals $\overline{\text{HBEN}}$ (high byte enable) and $\overline{\text{LBEN}}$ (low byte enable) are treated in an unusual manner here. When low, these signals switch the associated register from shifting to loading. Because $\overline{\text{HBEN}}$ and $\overline{\text{LBEN}}$ can never be low at the same time, and the clock inputs of the registers are interconnected, this results in the shift registers being in opposite mode all the time, i.e., when one loads, the other shifts. This has no significance for the loading of the high byte, while the high byte is shifted one position in IC3 during loading. Bits 7 and 8 in the high byte, which are not used by the converter, are made permanently high. This is done to bring about a level change on the CTS line after the shifting, when the low byte is loaded. This change can be used as an interrupt request in the PC, or as a 'conversion ready' status signal when the CTS line is 'polled' (i.e., continuously monitored).

Before the software can read the content of the shift registers with the aid of 16 clock pulses on the RTS line, it is necessary to wait 1.5 clock cycles (about 30 μs) for the $\overline{\text{LBEN}}$

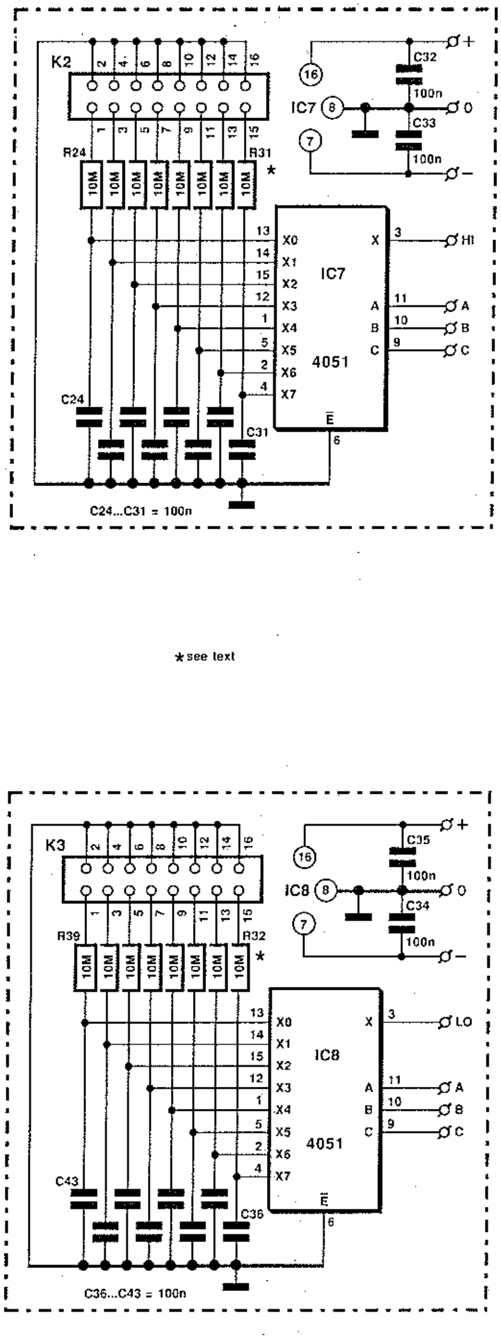
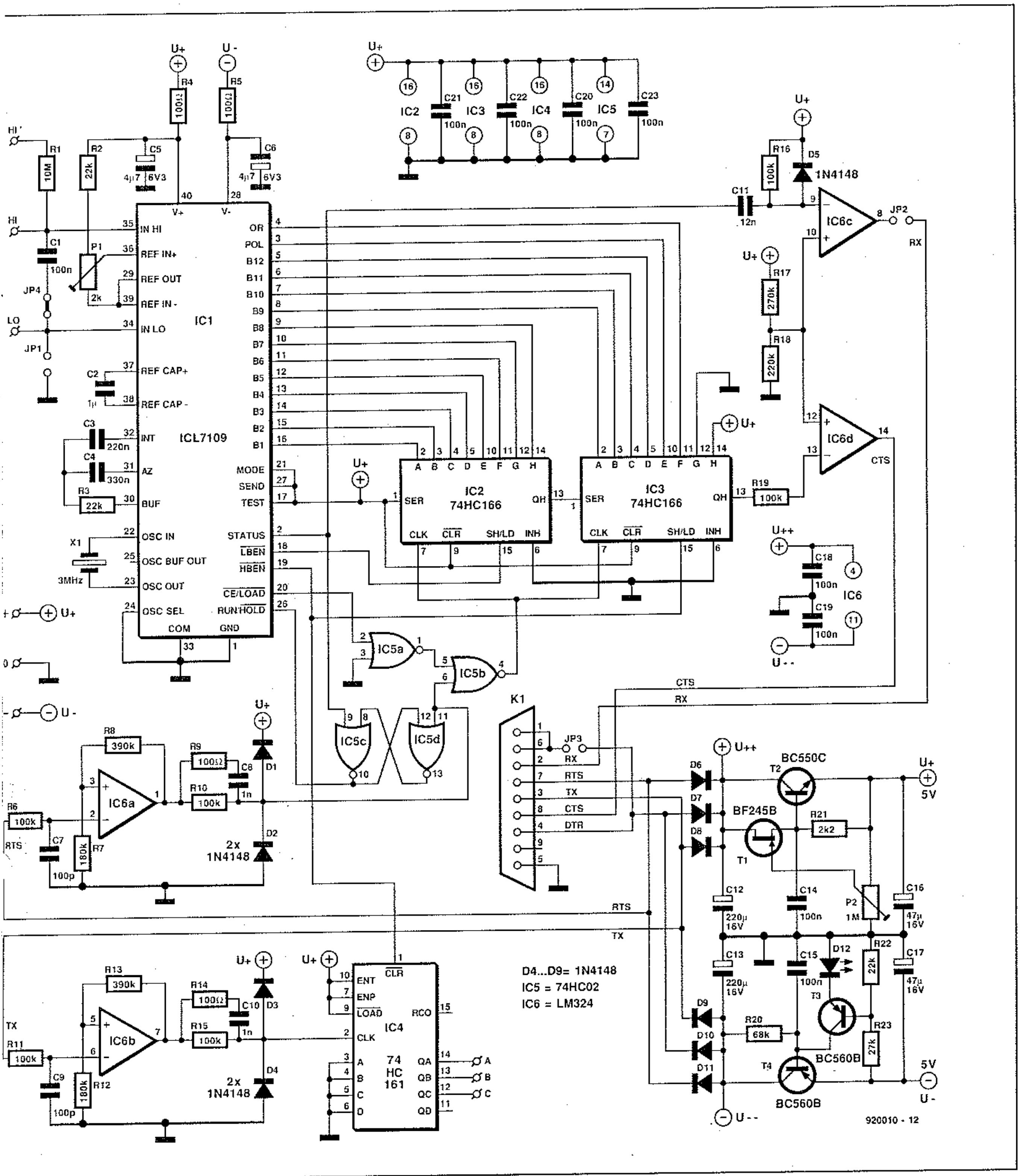


Fig. 2. Circuit diagram of the measurement system. The two input multiplexers are optional, and



al, and nstructed on separate printed circuit boards. Note that the entire circuit is powered via the RS232 link to the PC.

signal to return to high. This time can be used, for instance, to switch the multiplexer to the right channel.

The function of the bits in the datastream is shown in Table 1. The first clock pulse sets bistable IC5c-IC5d. This causes the RUN input to return to logic high, whereupon the A-D converter starts a new conversion cycle. The bistable is reset by a high level on the status line of IC1 (pin 2). The high level at the serial input of the low-byte shift register ensures that the output returns to high (–12 V on the CTS line) after the value has been read.

The ICL7109

The ICL7109 is a low-cost 12-bit dual-slope A-D converter with additional polarity and overflow outputs. Figs. 3 and 4 shed light on the operation of the digital part of the converter. First, Fig. 4: the upper signal represents the voltage at the capacitor (C3; the component numbers in brackets refer to the actual circuit). Below, some signals of the ICL7109, the internal clock, the clock signal of the output latch, the status output and the RUN/$\overline{\text{Hold}}$ input.

The conversion is started when a high level appears at the RUN input. During the

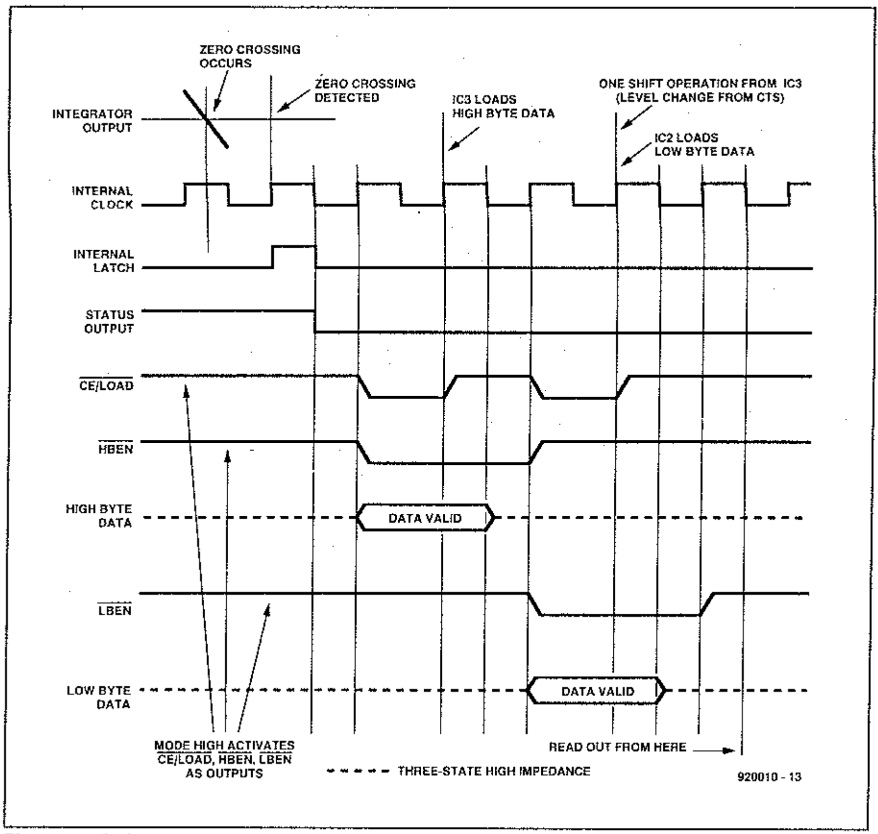


Fig. 3. Timing diagram of the data and command transfer protocol used.

Table 1. Overview of bit functions in dataword (16 clock pulses)

Clock on RTS	Function
1	Overrun (OR)
2	Polarity (POL)
3	B12
4	B11
5	B10
6	B9
7	High
8	B8
9	B7
10	B6
11	B5
12	B4
13	B3
14	B2
15	B1
16	High

conversion, the status output is held logic high. First, the integrator capacitor (C3) is charged with a current, I_c:

$$I_c = (U_{HI} - U_{LO})/R3.$$

This takes 2,048 internal clock cycles (oscillator frequency divided by 58). Subsequently, C3 is discharged with a constant current, I_d, that is proportional to the input voltage:

$$I_d = U_{ref}/R3.$$

During the discharging, the number of internal clock pulses is counted. The first leading edge of the internal clock that occurs after the capacitor voltage drops below 0 V causes the counter state to be transferred to the output register. Next, the status output changes from high to low, and so indicates the end of the conversion. Thus, we have:

$$\text{counter state} = 2{,}048\ U_{in}/U_{ref}.$$

If the RUN input remains high during the conversion, a new conversion cycle is started after the auto-zero phase. Conversely, if RUN remains low, the new conversion starts 7 internal clock pulses after a new high level is present.

Figure 3 illustrates the handshaking of the ICL7109 with the external circuits, when the Mode and Sense inputs are held logic high. To the signals shown in Fig. 4 are added $\overline{\text{LOAD}}$, $\overline{\text{HBEN}}$, $\overline{\text{LBEN}}$ and the data lines. A low level on $\overline{\text{HBEN}}$ signals the presence at the output of the high byte of the counter state and the status signals Overrun and Polarity. The high byte can be copied on the leading edge of the $\overline{\text{LOAD}}$ signal. The subsequent transfer of the low byte is similar.

The input of the ICL7109 may be considered symmetrical within the bounds of the supply voltage, and has an input resistance of the order of a few giga-ohms. To make sure that sufficient charge reserve is available for the switches in the analogue part of the converter, the voltage source to be measured must be buffered by a low-loss capacitor. This means that capacitor C1 must be connected in parallel with the two inputs when measurements are made without the multiplexer. This is achieved by fitting jumper J4. For measurements on symmetrical voltages, the voltages (with reference to ground) at the two inputs must lie within the bounds of the supply voltage of the ICL7109. For asymmetrical measurements, the LO input can be tied to ground by fitting jumper J1. Resistor R1 prevents discharging via the measured voltage source.

More inputs, more channels

A multiplexer is provided for those of you who want to measure more than one voltage source. The multiplexer consists of IC7 for the selection of the Hi line, and IC8 for the se-

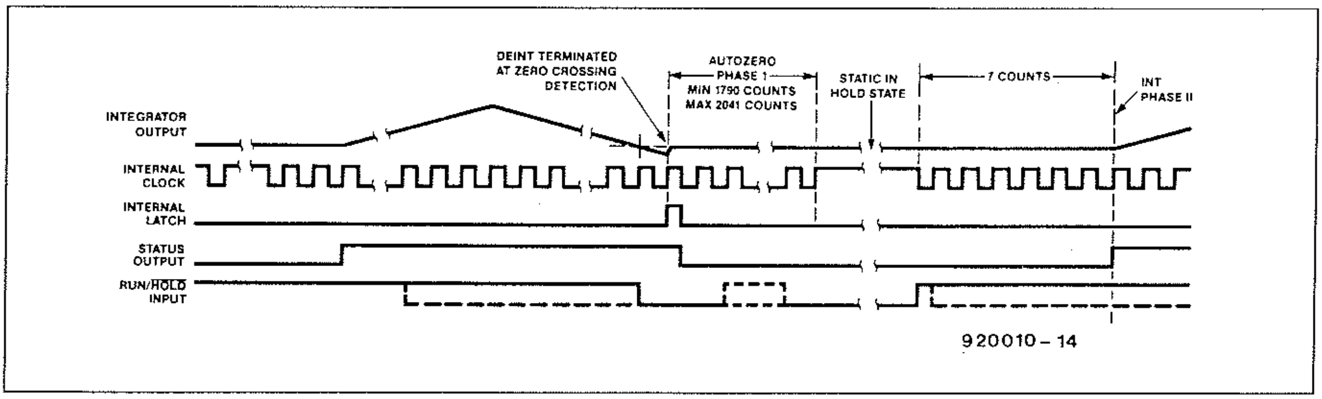


Fig. 4. ICL7109 analaogue-to-digital conversion timing.

lection of the Lo line. Both are controlled by IC4, which counts the clock pulses on the Tx line. To select a channel, the software puts certain characters on the Tx line, such that the total number of clock pulses corresponds to the desired channel (see the programming example 'setchannel' listed in Fig. 5). The set baud rate is irrelevant, but the (rarely used) parity function must be taken into account. To ensure a fixed starting configuration, the counter is automatically reset to channel 1 after each conversion.

When the multiplexer is used, capacitor C1 must be disconnected from the input (J4 is not fitted), and the ground reference must be raised (J1 is not fitted). Capacitors C24-C31 and C36-C43 then take over the function of C1.

```
Uses Crt;

CONST
   PosOver=MaxInt;          { value for pos. overflow }
   NegOver=-MaxInt;         { ditto neg. overflow }

VAR
   SIOAdr:Word;             { to be assigned with 8250 base address }
                            { e.g. COM1: SIOAdr:=$3f8 }
                            {        COM2: SIOAdr:=$2f8 }

{--------------------------------------------------------------}

{ set channel for next conversion: }

        PROCEDURE SetChannel(Channel:Byte);
CONST
   PulseBytes:ARRAY[1..5] OF Byte=(0,14,51,21,85); { -> 1 - 5 pulses to TX }

        PROCEDURE WaitTx;
BEGIN
   { wait until transmit shift register is free }
   REPEAT UNTIL Port[SIOAdr+5] AND 32<>0;
END;

BEGIN
   IF Channel>5 THEN      { 2 characters required? }
    BEGIN
     WaitTx; Port[SIOAdr]:=PulseBytes[5];
     WaitTx; Port[SIOAdr]:=PulseBytes[Channel-5];
     Waittx;
    END
   ELSE IF Channel>0 THEN { 1 character required? }
    BEGIN
     WaitTx; Port[SIOAdr]:=PulseBytes[Channel];
     WaitTx;
    END
   ELSE;                  { no pulse for Channel 0 }
END;

{Read value from shift registers. Call only when conversion is ready !!}

        FUNCTION ReadAD:Integer;
BEGIN
   Inline(
$8B/$16/>SIOAdr/    {       mov  dx,[SIOAdr]  ; DX at SIO-Port }
$83/$C2/$04/        {  add  dx,4              ; and on ModemCtrl there }
$BE/>0002/          {       mov  si,2         ; to switch address }
$33/$DB/            {       xor  bx,bx        ; clear shift accu }
$B5/$10/            {       mov  ch,16        ; 14 data +2 dummy clocks }
$B1/$04/            {       mov  cl,4         ; shift to CTS }
$EC/                {@read:in   al,dx         ; send pulse via RTS }
$24/$FD/            {       and  al,0fdh      ; RTS high }
$0C/01/             {  or    al,1             ; DTR complementary low }
$EE/                {       out  dx,al        ; set lines }
$51/                {       push cx           ; short wait loop, }
$B9/>$1F4/          {       mov  cx,500       ; to allow levels }
$E2/$FE/            {       loop $            ; to settle }
$59/                {       pop  cx           }
$0C/$02/            {       or   al,2         ; and restore }
$24/$FE/            {       and  al,0feh      ; DTR complementary high }
$EE/                {       out  dx,al        ; reset lines }
$51/                {       push cx           ; ensure }
$B9/>$1F4/          {       mov  cx,500       ; smallest pulse length }
$E2/$FE/            {       loop $            }
$59/                {       pop  cx           }
$80/$FD/$0A/        {       cmp  ch,10        ; pass dummy after 7 clocks }
$74/$0A/            {       je   @nuse        }
$03/$D6/            {       add  dx,si        ; DX to ModemStatus }
$EC/                {       in   al,dx        ; read status bits }
$D2/$E0/            {       shl  al,cl        ; shift CTS bit into carry }
$F5/                {       cmc               ; observe Inverting }
$D1/$D3/            {       rcl  bx,1         ; shift into result }
$2B/$D6/            {       sub  dx,si        ; DX back to ModemCtrl }
$FE/$CD/            {@nuse:dec  ch            ; outer loop counter }
$75/$D4/            {       jnz  @read        }
$D1/$EB/            {       shr  bx,1         ; shifted once too many }
$80/$F7/$10/        {       xor  bh,10h       ; change sign }
$F6/$C7/$20/        {       test bh,20h       ; overrun set? }
$75/$0D/            {       jnz  @Over        ; yes--> }
$F6/$C7/$10/        {       test bh,10h       ; polarity negative? }
$74/$16/            {       jz   @WErg        ; no-->result o.k. }
$80/$F7/$10/        {       xor  bh,10h       ; yes: clear Pol-Flag }
$F7/$DB/            {       neg  bx           ; ..and form 2's complement }
$EB/$0F/$90/        {       jmp  @WErg        }
$F6/$C7/$10/        {@Over:test bh,10h        ; negative overrun? }
$75/$06/            {       jnz  @ONeg        ; yes--> }
$BB/>PosOver/       {       mov  bx,PosOver   ; constant pos. overflow }
$EB/$04/$90/        {       jmp  @WErg        }
$BB/>NegOver/       {@ONeg:mov  bx,NegOver    ; constant neg. overflow }
$89/$5E/$FE);       {@WErg:mov  @result,bx    ; function result = BX }
END;

{ check if a valid result is available: }

        FUNCTION AD232_Ready:Boolean;
BEGIN
   AD232_Ready:=Port[SIOAdr+6] AND 16<>0;
END;

{ Initialisation of AD232. No timeout; endless loop created when AD232 is not found! }

        PROCEDURE AD232_Init;
VAR
   Dummy:Integer;
BEGIN
   Port[SIOAdr+4]:=2;          { only DTR low }
   Delay(100);             { allow supply voltage to stabilize }

   REPEAT UNTIL AD232_Ready; { wait for first result }
   Delay(1);
   Dummy:=ReadAD;              { discard result }
END;
```

Fig. 5. A 'bare bones' control program for the converter (Pascal with in-line assembler).

COMPONENTS LIST

Resistors:

17	10MΩ	R1;R24-R39
3	22kΩ	R2;R3;R22
4	100Ω	R4;R5;R9;R14
6	100kΩ	R6;R10;R11;R15; R16;R19
2	180kΩ	R7;R12
2	390kΩ	R8;R13
1	270kΩ	R17
1	220kΩ	R18
1	68kΩ	R20
1	2kΩ2	R21
1	27kΩ	R23
1	2kΩ multiturn preset	P1
1	1MΩ multiturn preset	P2

Capacitors:

19	100nF	C1;C14;C15; C18-C43
1	1µF	C2
1	220nF	C3
1	330nF	C4
2	4µF7 6V3 tantalum	C5;C6
2	100pF	C7;C9
2	1nF	C8;C10
1	12nF	C11
2	220µF 16V radial	C12;C13
2	47µF 16V radial	C16;C17

Semiconductors:

11	1N4148	D1-D11
1	green LED	D12
1	BF245B	T1
1	BC550C	T2
2	BC560B	T3;T4
1	ICL7109	IC1
2	74HC166	IC2;IC3
1	74HC161	IC4
2	74HC02	IC5
1	LM324	IC6
2	4051	IC7;IC8

Semiconductors:

1	9-way PCB-mount female sub-D connector	K1
2	16-way pin header	K2;K3
1	3-MHz quartz crystal	X1
1	Printed circuit board	920010
1	Control program on disk	ESS1691

The multiplexers also allow symmetrical measurements to be made within the bounds of the converter's supply voltage. When asymmetrical measurements are performed, the inputs of multiplexer IC9 must be tied to ground, for instance, by fitting jumpers on header K3.

Unfortunately, the 4051 and the high converter input resistance of the converter introduce an offset voltage that differs from channel to channel. Although the asymmetrical construction reduces the effect of the offset to a minimum, it can not be eliminated completely. For very accurate measurements, this means that the multiplexer must either not be used, or replaced by one with better specifications. For 'everyday' use, however, a correction in software of the measured voltage with the aid of a zero calibration is perfectly adequate.

RS232 level changer

The four opamps in IC6 are used as voltage level converters. Actually, they function as inverting switches, and so provide the inverting function defined in the RS232 standard. Diodes D1-D4 and resistors R10 and R15 limit the signal to +5 V and ground. *R-C* combinations R9-C8 and R14-C10 ensure that the input capacitances formed by IC5 and IC6

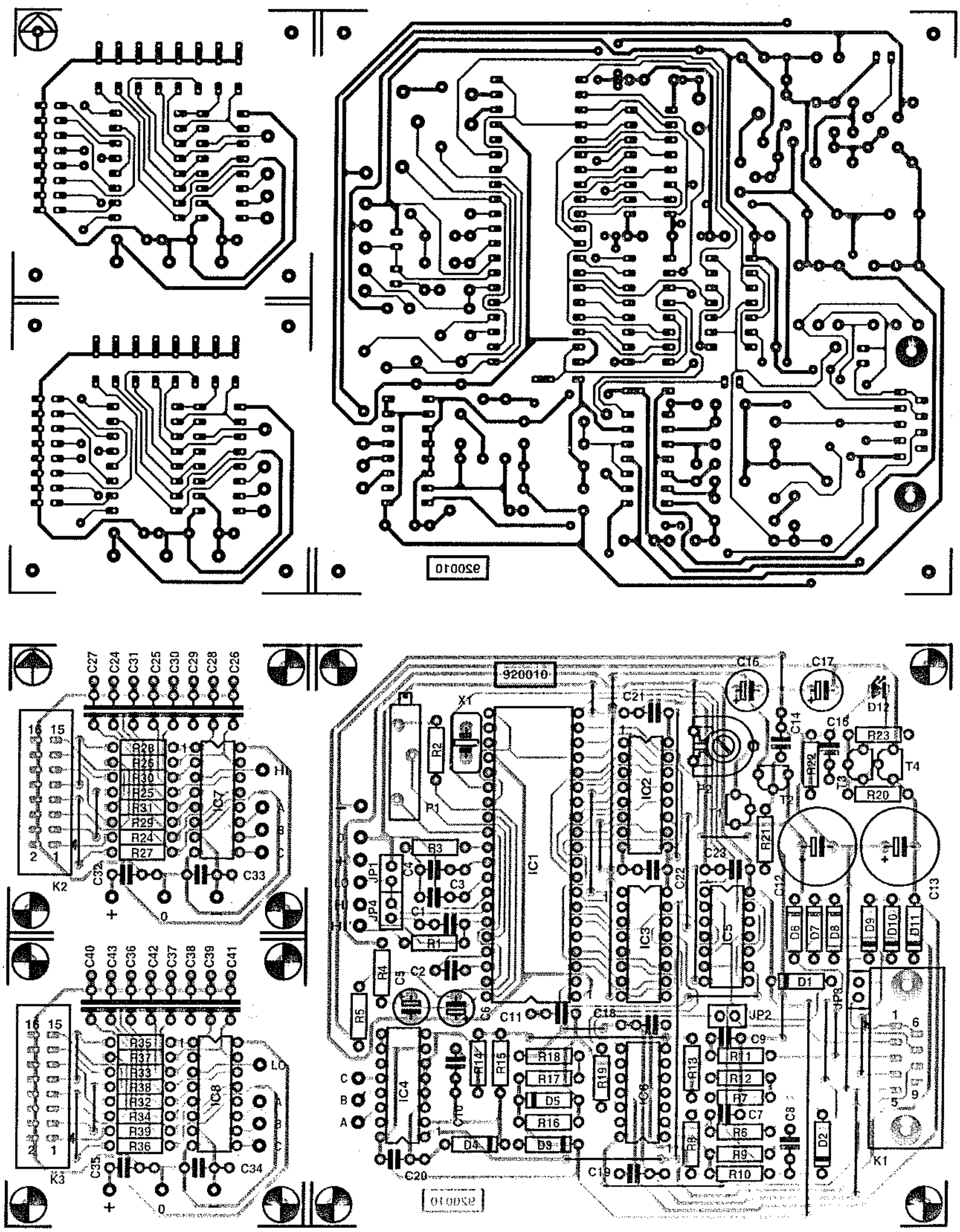
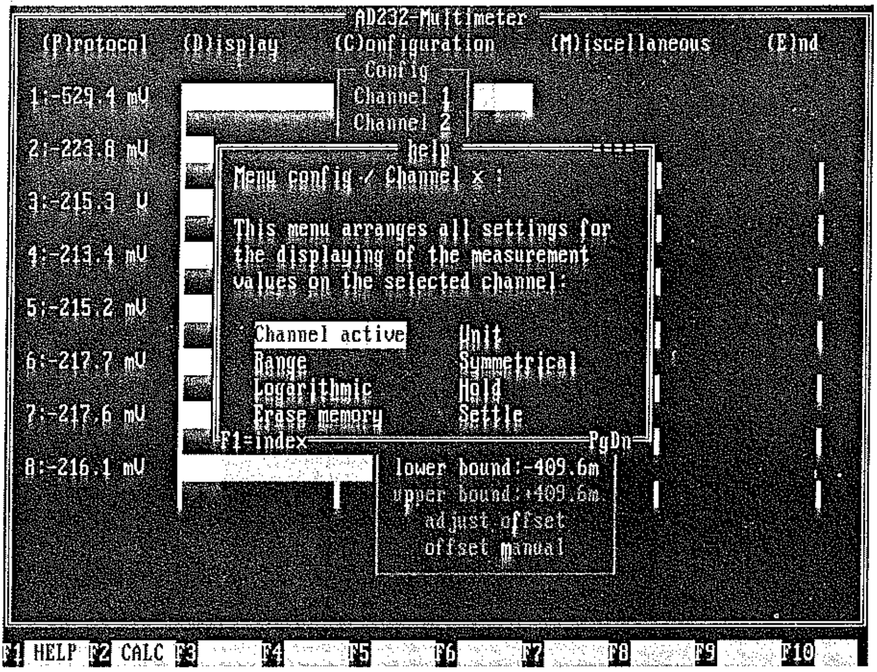


Fig. 6. Single-sided printed circuit board (track layout and component mounting plan) for the AD232.



```
                         AD232-Multimeter
 (P)rotocol  (D)isplay  (C)onfiguration   (M)iscellaneous   (E)nd
                          Config
 1:-529.4 mV              Channel 1
                          Channel 2
 2:-223.8 mV                 help
              Menu config / Channel x :
 3:-215.3  V
              This menu arranges all settings for
 4:-213.4 mV  the displaying of the measurement
              values on the selected channel:
 5:-215.2 mV
               Channel active    Unit
 6:-217.7 mV   Range             Symmetrical
               Logarithmic       Hold
 7:-217.6 mV   Erase memory      Settle
              F1=index                     PgDn
 8:-216.1 mV                 lower bound:-409.6m
                             upper bound:+409.6m
                               adjust offset
                               offset manual

F1 HELP F2 CALC F3    F4    F5    F6    F7    F8    F9    F10
```

need not charge via R10 and R15, and so reduce the effects of the relatively low slew rate of the opamps. Opamp IC6c is not strictly required for the correct function of the circuit. However, fit jumper J2 if you wish to feed the 'end of conversion' signal to your PC via the RxD line (for use with an appropriate interrupt routine).

Power supply

As already mentioned, the AD232 board is powered by the RS232 port of the computer it is connected to. The minimum requirement for this to be achieved is that the PC holds one RS232 line at +12 V and another at −12 V. Since the TxD line is at −12 V when it is not active, the +12 V supply will have to be supplied by the RTS and/or the DTR line. Diodes D6-D11 and capacitors C12 and C13 add and buffer the line levels to provide two discrete regulators with their input voltages. The voltage across the capacitors is also used to power the quadruple opamp. By omitting jumper JP3, the loading of the interface can be made as small as possible, if it can not supply enough current.

The voltage regulators in the ±5 V supply are low-drop, high efficiency, types built from discrete components. The negative regulator consists of a pseudo-zener diode with a series transistor formed by T3 and a green LED, D12. Adjustment of the negative output voltage is not necessary because the A-D converter has a fairly large negative voltage range. Not so with the positive supply, where the pinch-off voltage of a FET serves as the reference. Although this voltage is reasonably stable, it has a fairly high device tolerance, and needs to be adjusted with the aid of P2. To save on parts, protection against overvoltage as a result of an incorrect adjustment is not provided. This means that the wiper of P2 must be turned to ground before calibrating the AD232 board. The +5-V voltage must be adjusted with the nominal load connected.

Construction and adjustment

The single-sided PCB designed for the circuit is shown in Fig. 6. The construction will be mostly plain sailing. The multiplexers are built as separate units, and connected to the ADC proper in accordance with the component overlay. The HI' input serves for initial tests without the multiplexer. It is important to ensure adequate screening of all signal lines—remember, the high input resistance makes the circuit sensitive to noise. Hum suppression will be optimum when a quartz crystal of 2.969600 MHz is used. Unfortunately, this is not a standard frequency, whence the use of a 3-MHz crystal here.

The adjustment of the ADC by P1 depends on the application. In principle, the fine adjustment can be done by programming. The external circuitry around the ADC is designed for a maximum input voltage of about 400 mV. For other voltages, R3 must be changed:

$$R3 = U_{max}/20\ \mu A.$$

The reference voltage, U_{ref}, should be a little higher than $U_{max}/2$, because the converter produces an overflow when $U_{max} \geq 2U_{ref}$.

Connect the AD232 board to the PC, and advance preset P2 slowly until the positive regulator supplies +5 V.

The software package

Although the routines listed in Fig. 5 can be expanded into a full-blown control program for the AD232 board, you will be pleased to know that such a program is available on a disk supplied that can be ordered through our Readers Services (order code 1691).

This program is completely menu-driven, runs in colour, accepts mouse as well as keyboard control, and is written to run on IBM PCs and compatibles, from XTs to 486-based machines. A colour video card (EGA or VGA) is not strictly required, although you will miss a lot of the presentation graphics' power when you have monochrome video only.

Installation

The program can be run from floppy disk or hard disk. When it is run from floppy disk, make sure the write protection is removed, because the program writes a configuration file on the disk. An installation proper is not required—simply copy all files to your working disk, or to the hard disk.

Running the program

The AD232 control program is started by typing MULTI from the DOS prompt. First, go the (M)iscellaneous option, and select the serial port to which the AD232 board is connected. On leaving this menu, the voltmeter should work, indicating the voltages on all eight channels by means of horizontal bars and an absolute readout. (see Fig. 7). When fewer than eight channels are used, the screen is automatically enlarged.

Options (configuration menu)

The channel settings are not limited to switching on and off. Measuring ranges, multipliers and units (mV; V, etc.) can be taken into account in the graphics readout. The bar that indicates the magnitude of the measured voltage can be asymmetrical or symmetrical, and linear or logarithmic. Use 'symmetrical' for measured quantities that can go positive and negative. The menu also includes program options for offset calibration and a smoothing (delta) factor to stabilize the display.

Protocol

Apart from being shown on the screen in the form of horizontal bars, the measured values may also be sent to a file or an output device such as a printer. All output is in straight ASCII to a simple protocol, which makes further processing by other software easy.

Extras

All screen elements (text, highlighted text, borders, background) can be displayed in a number of user-selectable colours (VGA/EGA). Irrespective of your whereabouts in the program, a scientific (UPS compatible) desktop calculator and context sensitive help are always to hand. The help texts used in the program are stored in a file called MULTI.HLP, which is generated by adding WINHELP.TXT and MULTIHLP.TXT on the diskette, as explained in README.DOC. All files and menus are in English. ■