using the Junior Computer as a voltmeter

With the addition of a small circuit and the aid of the accompanying program the Junior Computer can be used as a digital voltmeter . . . and a good one at that! The voltmeter has 3¾ digits and an automatic polarity indicator, even though the program is less than 180 bytes long.

G. Sullivan

Voltmeters are always useful as far as anybody interested in electronics is concerned. Junior Computer owners can use this straightforward circuit and program to convert their computer into an excellent digital voltmeter.

The basis of the circuit is an A/D converter from Intersil. This IC has binary outputs and will convert the input signal level into the BCD code. Since the IC has tri-state outputs, it is suitable for use with microprocessors.

The circuit

The Intersil ICL 7109 contains a complete voltmeter circuit for a 3¾ digit display including automatic polarity indication. This means the whole circuit can be very straightforward.

Figure 1 shows the circuit diagram. The voltmeter IC contains a 12 bit A/D converter with tri-state outputs. The outputs B1 . . . B8 are displayed in two bytes. Which byte is displayed depends on the address decoder IC2. The low order byte contains the eight least significant bits and the high order byte the four most significant bits plus an overflow bit and a polarity bit.

Using the given specifications, the

1

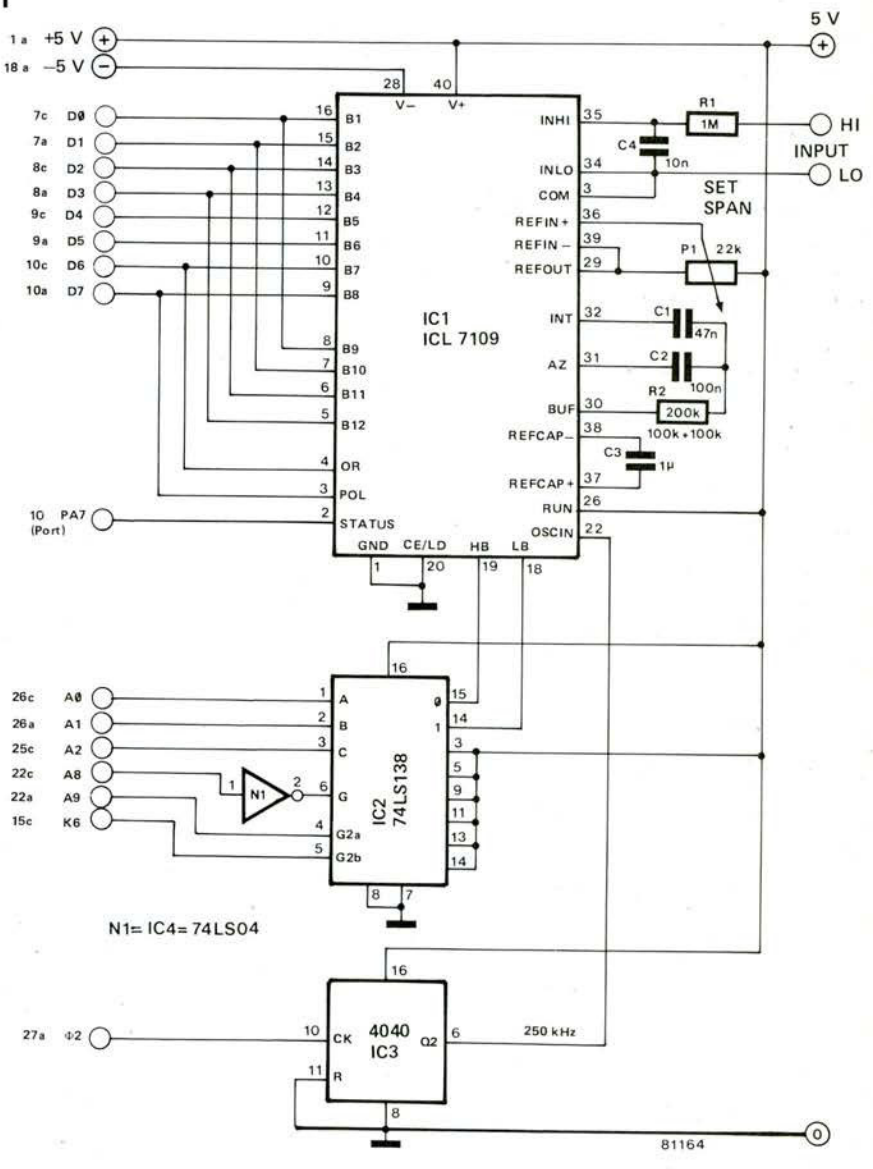

Figure 1. The digital voltmeter circuit diagram which enables the Junior Computer to measure DC voltages. IC1 takes care of the whole process, converting an analogue input signal into a 12 bit code for the computer.

input voltage for 'full scale deflection' will be 4.096 V. The conversion speed is about 30 per second. The input range can be varied by changing the value of R2 and by modifying the reference voltage which is set by P1.
Thus,

$$R2 = \frac{U_{full\ scale}}{20\ \mu A} \text{ and}$$

$$U_{full\ scale} = 2 \cdot U_{ref}.$$

The value of C1 and C2 is determined by the oscillator frequency used, where

$$C1 = \frac{2048 \cdot period \cdot 20\ \mu A}{3.5\ V}$$

and C2 = 2 · C1.

The 250 kHz frequency that is used for the converter is derived from the microprocessor clock. For this IC3 is connected as a divide-by-four. If required, the converter may be run at a different frequency by choosing a different output of IC3. This will also alter the number of conversions per second. Because of the converter's high input impedance, input attenuators can be included fairly easily to provide several voltage ranges.

The program

The program which enables the JC to be used as a digital voltmeter is shown in table 1. This will read in the two bytes produced by the converter, after which a binary-to-decimal conversion takes place and the result is then shown on the display. In the case of a negative input signal the status of the polarity flag is detected and a minus sign is shown on the display. If the maximum input voltage of the converter is exceeded, the display will indicate OL (overload) together with the polarity.
In the circuit diagram in figure 1 the converter is connected in the 'free run mode'. This means it will start with the next conversion as soon as an analog-to-digital conversion has been completed. This is fine under normal circumstances, but it may prove necessary to detect the actual moment a conversion has ended and then read in the data to avoid this happening while the data is changing.
This can be done by connecting the status output of IC1 to PA7 of the port connector and by using the negative-going edge of this output to enable an interrupt (IRQ) at the end of a conversion. The interrupt routine can then read in and store the two bytes before the following conversion comes to an end. An example of this interrupt routine is provided in table 2.
Once the two programs have been loaded beginning with address Ø2ØØ, the

Table 1.

```
LINE  LOC   OBJECT      SOURCE

0001  0000               ;DIGITAL VOLTMETER PROGRAM
0002  0000               ;FOR INTERSIL ICL 7109.
0003  0000               ;
0004  0000               ;AUTHOR G.SULLIVAN
0005  0000               ;
0006  0000                       * = $00D0
0007  00D0               ACUM   *=*+2
0008  00D2               SUM    *=*+1
0009  00D3               DELAY  *=*+1
0010  00D4               TMPX   *=*+1
0011  00D5               ;
0012  00D5               ;DEFINE A/D CONVERTER.
0013  00D5                       * = $1800
0014  1800               HB     *=*+1            ;HIGH NIBLE + FLAGS
0015  1801               LB     *=*+1            ;LOW BYTE
0016  1802               ;
0017  1802               ;DEFINE PIA
0018  1802                       * = $1A80
0019  1A80               PRA    *=*+1            ;DATA A REG.
0020  1A81               DDRA   *=*+1            ;A DIRECTION REG.
0021  1A82               PRB    *=*+1            ;DATA B REG.
0022  1A83               DDRB   *=*+1            ;B DIRECTION REG.
0023  1A84               ;
0024  1A84               SCAND1 =$1DCC           ;DISPLAY 1 BYTE
0025  1A84               ;
0026  1A84                       * = $0200
0027  0200               ;
0028  0200               ;MAIN DISPLAY ROUTINE
0029  0200  2C 00 18     MAIN   BIT HB           ;TEST OVERANGE BIT
0030  0203  70 06                BVS OL
0031  0205  20 24 02             JSR DISVLT      ;SHOW VOLTS
0032  0208  4C 12 02             JMP NOL
0033  020B  A2 0C        OL     LDX #$0C         ;DISPLAY OL MESSAGE
0034  020D  A0 03                LDY #$03
0035  020F  20 49 02             JSR DISTAT
0036  0212  AD 00 18     NOL    LDA HB           ;TEST POLARITY BIT
0037  0215  30 07                BMI NOT
0038  0217  A2 08                LDX #$08        ;DISPLAY MINUS
0039  0219  A0 00                LDY #$00
0040  021B  20 49 02             JSR DISTAT
0041  021E  20 73 02     NOT    JSR HEXBCD       ;CONV. BINARY TO BCD
0042  0221  4C 00 02             JMP MAIN
0043  0224               ;
0044  0224               ;VOLT DISPLAY SUBROUTINE
0045  0224  A9 7F        DISVLT LDA #$7F         ;SET PIA TO OUTPUT
0046  0226  8D 81 1A             STA DDRA
0047  0229  A2 0C                LDX #$0C        ;ADDRESS OF FIRST BYTE
0048  022B  A0 FF                LDY #$FF
0049  022D  C8          LOOP   INY
0050  022E  B9 D0 00             LDA ACUM,Y      ;GET BYTE
0051  0231  20 CC 1D             JSR SCAND1      ;LIGHT DISPLAY
0052  0234  E0 14                CPX #$14        ;TEST IF TWO BYTES YET
0053  0236  D0 F5                BNE LOOP
0054  0238  AD 00 18             LDA HB          ;TEST POLARITY
0055  023B  30 07                BMI NXT         ;SHOW - IF NEG.
0056  023D  A2 08                LDX #$08
0057  023F  A0 00                LDY #$00
0058  0241  20 49 02             JSR DISTAT
0059  0244  E6 D3        NXT    INC DELAY       ;DELAY ON DISPLAY
0060  0246  D0 DC                BNE DISVLT
0061  0248  60                   RTS
0062  0249               ;
0063  0249               ;DISPLAY '-','OL'
0064  0249  A9 7F        DISTAT LDA #$7F         ;SET PIA TO OUTPUT
0065  024B  8D 81 1A             STA DDRA
0066  024E  B9 6D 02             LDA CHART,Y     ;GET MESSAGE BYTE
0067  0251  30 14                BMI ENDD
0068  0253  8D 80 1A             STA PRA         ;LIGHT SEGMENTS
0069  0256  8E 82 1A             STX PRB         ;SELECT DIGIT
0070  0259  86 D4                STX TMPX
0071  025B  A2 FF                LDX #$FF
0072  025D  CA          DLY2   DEX              ;DELAY
0073  025E  D0 FD                BNE DLY2
0074  0260  A6 D4                LDX TMPX
0075  0262  E8                   INX
0076  0263  E8                   INX             ;NEXT DIGIT
0077  0264  C8                   INY
0078  0265  D0 E2                BNE DISTAT
0079  0267  A9 00        ENDD   LDA #$00         ;DISPLAY OFF
0080  0269  8D 82 1A             STA PRB
0081  026C  60                   RTS
0082  026D  7F          CHART  .BYTE $7F,$3F,$80
0082  026E  3F
0082  026F  80
0083  0270  40                   .BYTE $40,$47,$80
0083  0271  47
```

```
0083  0272  80
0084  0273
0085  0273                ;CONVERT BINARY TO BCD
0086  0273  A9 00        HEXBCD LDA #$00
0087  0275  85 D0               STA ACUM        ;CLEAR ACCUMULATOR
0088  0277  85 D1               STA ACUM+1
0089  0279  AE 01 18            LDX LB
0090  027C  F0 0B               BEQ HIGH
0091  027E  A0 01        LOOP1  LDY #$01        ;CONVERT LOW BYTE
0092  0280  98                  TYA
0093  0281  85 D2               STA SUM
0094  0283  20 9E 02            JSR ADD
0095  0286  CA                  DEX
0096  0287  D0 F5               BNE LOOP1
0097  0289  AD 00 18     HIGH   LDA HB          ;CONVERT HI BYTE
0098  028C  29 0F               AND #$0F        ;REMOVE FLAGS
0099  028E  F0 0D               BEQ LAST
0100  0290  AA                  TAX
0101  0291  A9 08               LDA #$08
0102  0293  85 D2               STA SUM
0103  0295  A0 20        LOOP2  LDY #$20
0104  0297  20 9E 02            JSR ADD
0105  029A  CA                  DEX
0106  029B  D0 F8               BNE LOOP2
0107  029D  60          LAST   RTS
0108  029E  F8          ADD    SED             ;ADD 1 OR 256 TO ACUM
0109  029F  18          LOOP3  CLC
0110  02A0  A5 D2               LDA SUM
0111  02A2  65 D1               ADC ACUM+1
0112  02A4  85 D1               STA ACUM+1
0113  02A6  A9 00               LDA #$00
0114  02A8  65 D0               ADC ACUM
0115  02AA  85 D0               STA ACUM
0116  02AC  88                  DEY
0117  02AD  D0 F0               BNE LOOP3
0118  02AF  D8                  CLD
0119  02B0  60                  RTS
0120  02B1                     .END

ERRORS    0000

SYMBOL TABLE

 ACUM      00D0     ADD       029E     CHART     026D     DDRA      1A81
 DDRB      1A83     DELAY     00D3     DISTAT    0249     DISVLT    0224
 DLY2      025D     ENDD      0267     HB        1800     HEXBCD    0273
 HIGH      0289     LAST      029D     LB        1801     LOOP      022D
 LOOP1     027E     LOOP2     0295     LOOP3     029F     MAIN      0200
 NOL       0212     NOT       021E     NXT       0244     OL        020B
 PRA       1A80     PRB       1A82     SCAND1    1DCC     SUM       00D2
 TMPX      00D4

END OF ASSEMBLY
```

Table 2.

IRQ service routine:

```
0380  48          INTS: PHA          ; save A
0381  8A                TXA          ; save X
0382  48                PHA
0383  98                TYA          ; save Y
0384  48                PHA
0385  AD 85 1A           LDA 1A 85    ; reset IRQ
0388  AD 00 18           LDA 18 00    ; read high byte
038B  85 D0              STA D0       ; store it
038D  AD 01 18           LDA 18 01    ; read low byte
0390  85 D1              STA D1       ; store it
0392  68                 PLA          ; restore all registers
0393  A8                 TAY
0394  68                 PLA
0395  AA                 TAX
0396  68                 PLA
0397  40                 RTI          ; return to main program
```

IRQ vector has to be specified:

```
1A7E  80
1A7F  03
```

Finally, the initialisation routine which is needed whether or not the interrupt routine is used:

```
0000 8D 86 1A   STA 1A 86
0003 58         CLI
0004 4C 00 02   JMP-MAIN
```

Now a reference voltage is connected to the input of the meter circuit (4 V, for instance) and P1 is adjusted so that the display indicates the value of the reference voltage. In the absence of a known voltage, any DC voltage of about 4 V can be used instead and the display can be compared to that of another, accurate meter.

These instructions make sure the PIA produces a negative-going edge at PA7 and that the interrupt-disable bit is restored once the processor is reset. When this has been done and the circuit in figure 1 is connected to the Junior Computer, the program can be started at address 0000.