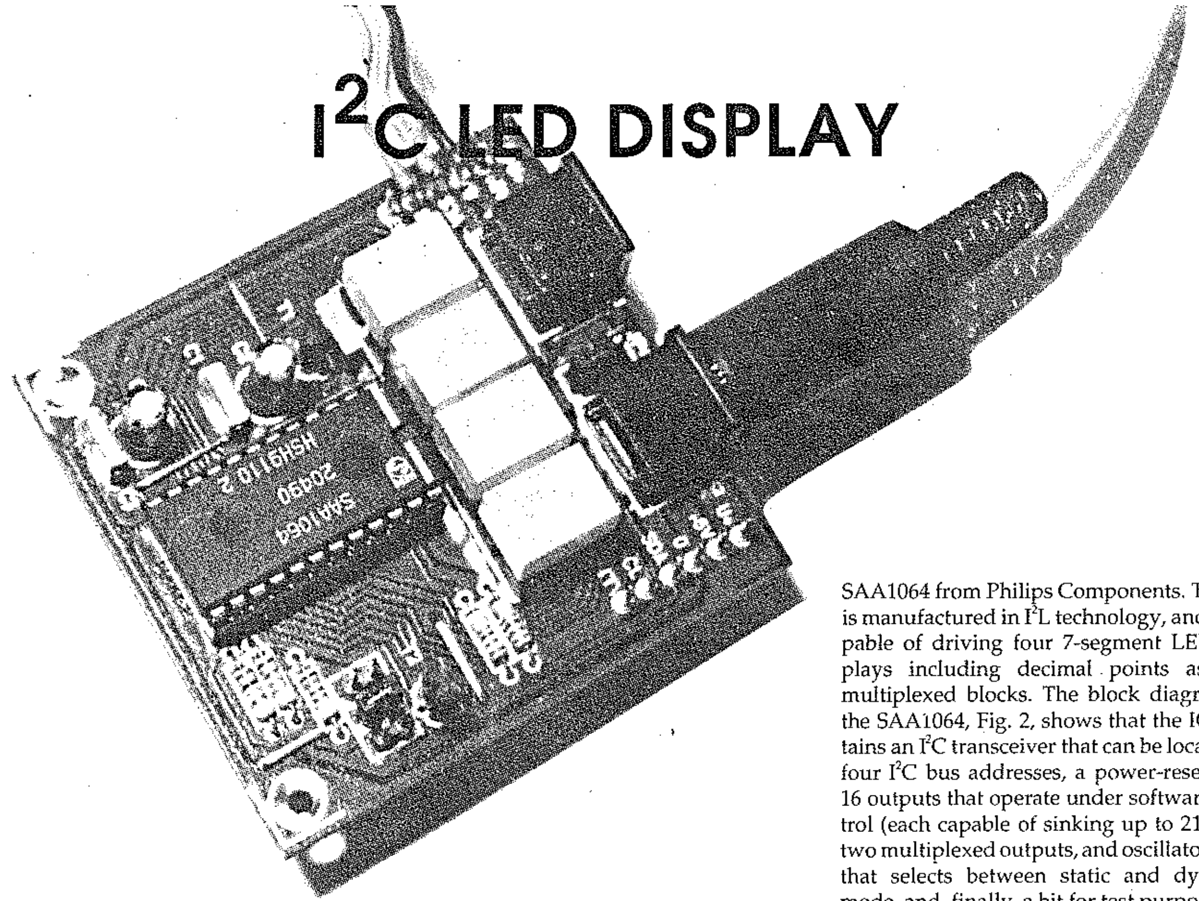
I^{2}C LED DISPLAY



Our earlier publications on I^{2}C-compatible circuits having met with a great deal of interest, we now move on to a four-digit seven-segment LED display unit that can be used in many applications as a bright and easy to drive readout.

Design by J. Ruffell

SO far it has appeared a matter of course that the readout function in a computerized measurement system is invariably provided by the screen of the PC. The present 4-digit LED display unit may, however, prove an attractive alternative in many cases. It allows a PC fitted with an I^{2}C controller card (Ref. 1), or any other I^{2}C controller, to indicate, for instance, measured data in bright green, red or yellow numbers on an LED display. In other words, it is no longer necessary to reserve a part of the PC's screen for displaying measured values. This means that the PC can continue to run its main program, while a background program takes care of outputting measurement data to the LED display at regular intervals.

Another possibility is to use the display as a time and/or date indication. In a PC, the parameters needed for this application are easily copied from the system software or the PC's internal real-time clock. Another possible application that comes to mind is to use the display to bring to your attention the status of a certain measurement program that runs in the background.

The present display is, of course, software-compatible with the I^{2}C device driver published earlier (Ref. 2), and its control is, therefore, 'food for programmers'. Those of you who work with microcontrollers will also find that the I^{2}C display unit is readily used and fairly simple to control. Today, an increasing number of microcontrollers is available with an on-chip I^{2}C interface, which makes connecting the present display unit a piece of cake.

The circuit

The circuit shown in Fig. 1 is best described with three words: compact, simple and inexpensive. Apart from one IC, two transistors and, of course, four 7-segment LED displays, only a handful of passive parts is required to complete the circuit. As with the other circuits in our I^{2}C series, the communication with the outside world is via two miniature 6-way DIN sockets. A length of 6-way cable is all that is required to convey the two serial signals, ground, the supply voltage and an interrupt signal (the +5 V supply voltage is carried via two wires).

The heart of the circuit is formed by a four-digit I^{2}C-compatible LED driver Type SAA1064 from Philips Components. This IC is manufactured in I^{2}L technology, and is capable of driving four 7-segment LED displays including decimal points as two multiplexed blocks. The block diagram of the SAA1064, Fig. 2, shows that the IC contains an I^{2}C transceiver that can be located at four I^{2}C bus addresses, a power-reset flag, 16 outputs that operate under software control (each capable of sinking up to 21 mA), two multiplexed outputs, and oscillator, a bit that selects between static and dynamic mode, and, finally, a bit for test purposes.

As could be expected of an I^{2}C application circuit, the control of the display driver is a matter of sending the right command to a previously determined address, which is the 'location' of the IC in the I^{2}C network. Here, the address of the SAA1064 can be set with the aid of jumpers. Remarkably, only one IC pin is used to select one of four possible addresses in the system. Resistors R3, R4 and R5 form a voltage divider which supplies the address selection voltage to the ADR pin of the SAA1064. The voltage level, i.e., the IC address, is determined by three jumpers. The four voltage levels that can be set are 0 V, 3⁄8Vcc, 5⁄8Vcc and Vcc. Each of these levels corresponds to one of the four combinations of the two address bits, A0 and A1.

The base address of the SAA1064 is

0 1 1 1 0 A1 A0 x

To select an address, fit the jumpers as shown in Table 1. All other bits in the I^{2}C ad-

Table 1.

A0	A1	JP1	JP2	JP3
0	0	0	1	0
1	0	0	0	0
0	1	0	0	1
1	1	1	0	0

0 = open
1 = jumper fitted

dress are 'burnt' in the IC hardware, and can not be changed. As usual with I²C compatible ICs, the 'x' at the end of the address is a bit that selects between a read (x=1) or a write (x=0) operation. The 'read' addresses are 70H, 72H, 74H and 76H. The 'write' addresses are 71H, 73H, 75H and 77H.

Transistors T1 and T2 are required to multiplex the four common-anode displays pair-wise. They function as switches, with the hardware in the SAA1064 determining the current flow through the display segments. Software control allows the segment current to be set between 3 mA and 21 mA in steps of 3 mA. In dynamic mode, the segments light about half the time (48.2% typ. min.). The circuit configuration used here results in a multiplex frequency of about 150 Hz, which can be increased to about 800 Hz or 1,500 Hz by reducing C5 to 820 pF or 390 pF respectively.

The multiplex duty factor results in an average segment current that is about half the programmed current. Since the brightness of the Type HD1105O LED displays (manufacturer: Siemens) is sufficient at a segment current of 4.5 mA, a segment current of 9 mA is programmed.

Jumper JP4 allows the display to be powered by a separate supply, which may be required when more than one display is used. The external voltage is applied via PCB terminal 'U+', and may be up to 15 V, pro-

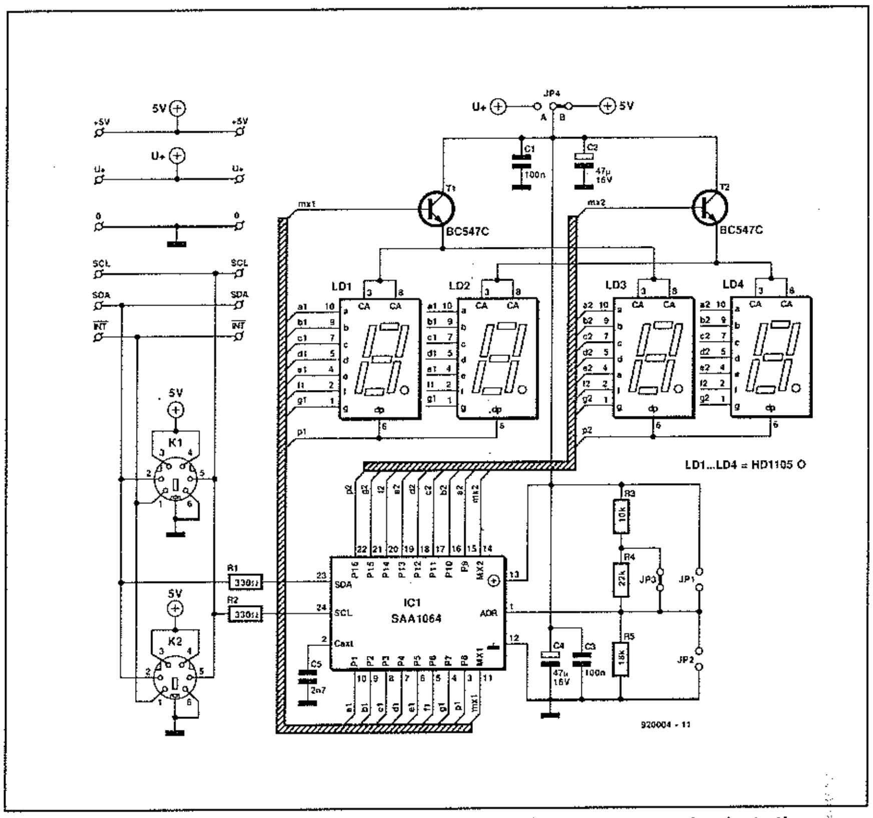


Fig. 1. The circuit for driving four 7-segment displays is very compact thanks to the power of the I²C protocol.

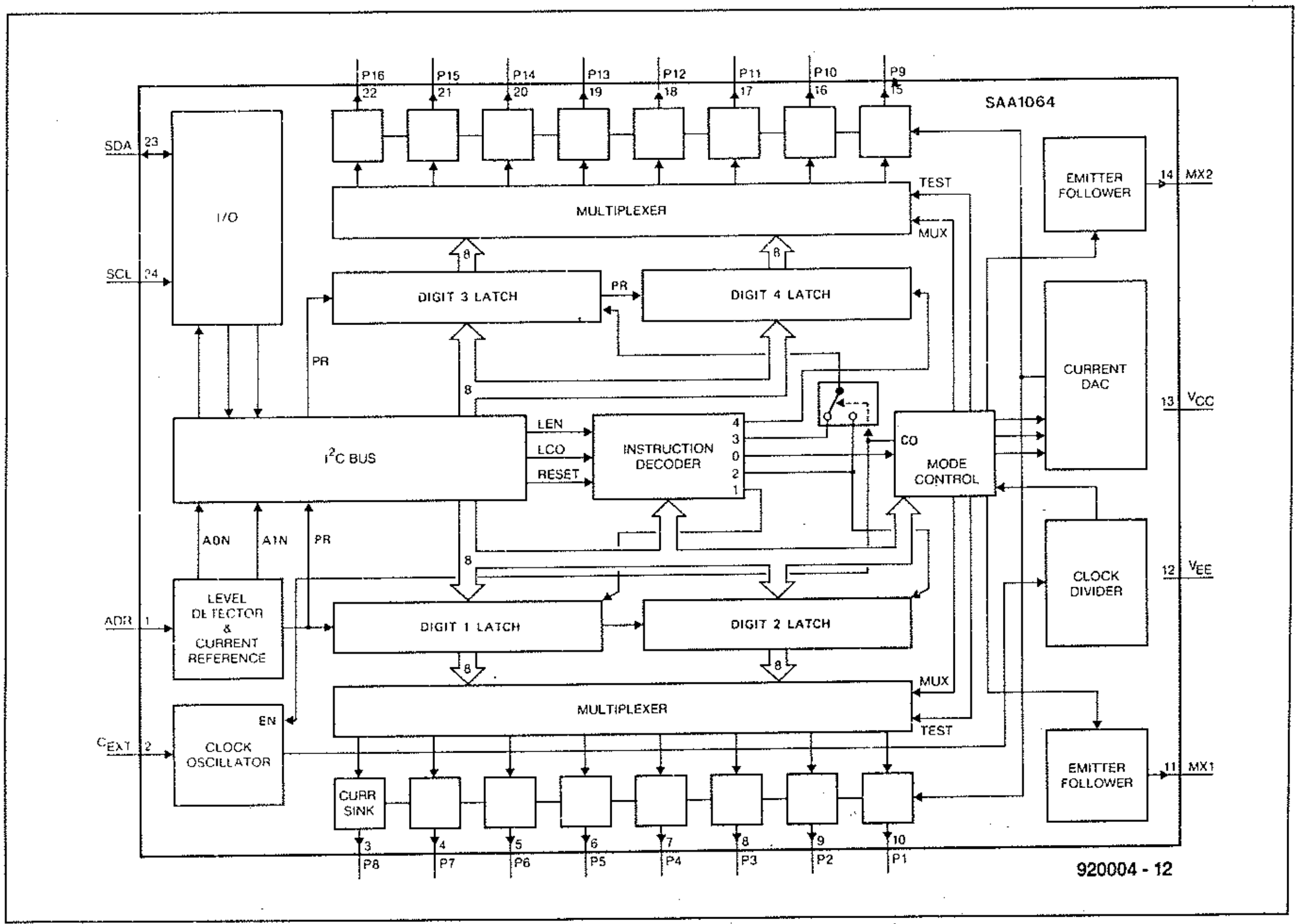


Fig. 2. Internal diagram of the SAA1064 LED display controller (courtesy Philips Components).

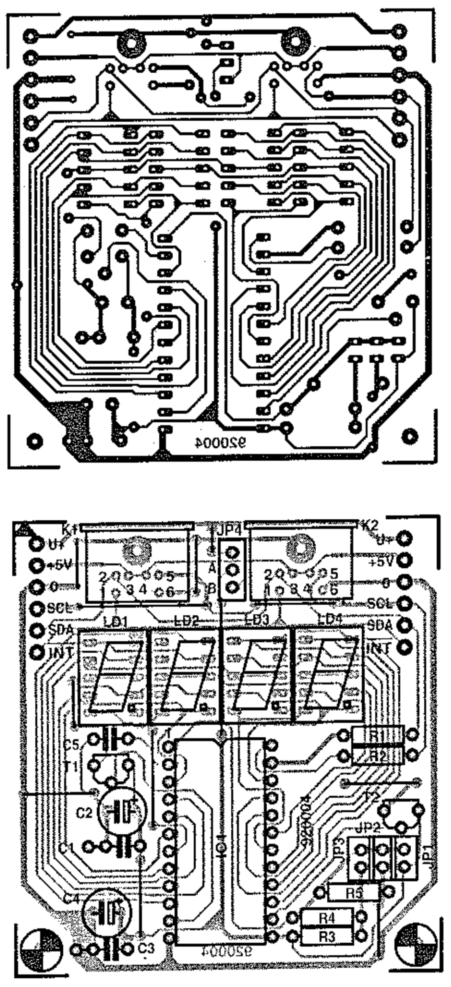
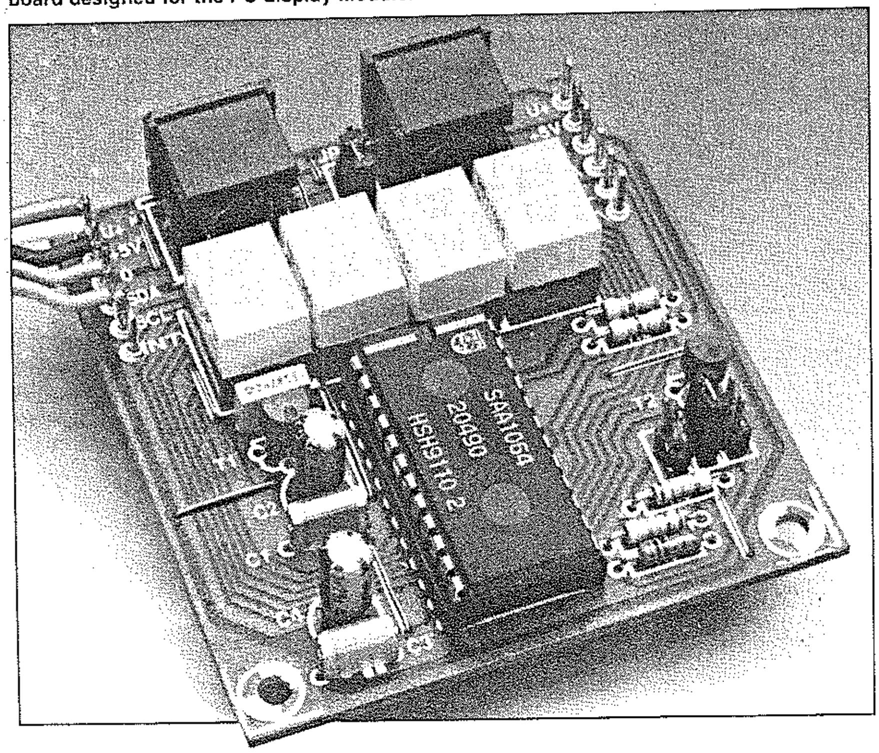


Fig. 3. Track layout (mirror image) and component mounting plan of the printed circuit board designed for the I²C display module.



COMPONENTS LIST

Resistors:		
2	330Ω	R1;R2
1	10kΩ	R3
1	22kΩ	R4
1	18kΩ	R5
Capacitors:		
2	100nF	C1;C3
2	47µF 16V radial	C2;C4
1	2nF7	C5
Semiconductors:		
2	BC547C	T1;T2
1	SAA1064	IC1
Miscellaneous:		
2	6-way PCB-mount mini-DIN socket	K1;K2
4	HD1105O/G/Y/R (see text)	LD1-LD4
1	Printed circuit board	920004
1	Control software on disk	ESS 1671

vided the total dissipation in the SAA1064 does not exceed 1 W.

Resistors R1 and R2, finally, ensure that the I²C bus is correctly terminated.

Construction

The printed circuit board for the I²C compatible display unit is compact — see Fig. 3 and the photographs. One of the mini-DIN sockets may be omitted if the unit is the last (or the only) device on the I²C bus. Since the two sockets are connected in parallel, it makes no difference which of them is omitted. If you can not get hold of the special DIN sockets, or wish to reduce the cost of building the unit, use separate wires and the PCB terminals marked +5 V, 0 V, SDA, SCL and $\overline{\text{INT}}$ instead.

Start the construction by fitting the wire links. Next, mount the passive components, followed by the IC and the two transistors. If you solder the displays directly on to the board, make sure that they are not overheated — in general, it is better to fit the displays in IC sockets.

Set the jumpers as follows:

JP1 and JP3 open;
JP2 closed;
JP4 position 'B',

and connect the display unit to the I²C bus.

We built a few prototypes of the unit with different display colours. The suffix of the display type number indicates the colour: HD1105O: orange; HD1105G: green, HD1105R: red; HD1105Y: yellow.

Software control

As already mentioned, the display is controlled by an I²C controller board described in Ref. 1. The control software has been described in Ref. 1, and is available on disk under order number ESS 1671.

Figure 4 shows how the controller communicates with the SAA1064. Reading results in a status byte, which, among others, shows the state of the power-reset flag. This flag is set by the SAA1064 when power is applied, after which all registers contain zeroes, and the display is blank.

There are a number of ways in which we can write to the SAA1064. Writing to the device requires the relevant control register to be set to the right mode, and data to be sent to the display digits. After addressing the SAA1064, an instruction byte is sent that selects one of the eight internal registers. Which register is selected first is determined by the level of bits SA, SB and SC. The auto-increment function of the IC ensures that the next register is automatically selected for writing to. The pointer of the auto-increment function is cyclic, and changes to '0' again after '7'. The three least-significant bits of the instruction byte select the registers as follows:

b2	b1	b0	
0	0	0	control register
0	0	1	LD1 register
0	1	0	LD2 register
0	1	1	LD3 register
1	0	0	LD4 register
1	0	1	free
1	1	0	free
1	1	1	free

where b7 to b3 = 0

The structure of the control byte is as follows:

b0 = 1:	dynamic mode (multiplex digits)
b1 = 1:	enable digits 1 and 3
b2 = 1:	enable digits 2 and 4
b3 = 1:	segment test, all outputs active
b4 = 1:	increase segment current by 3 mA
b5 = 1:	increase segment current by 6 mA
b6 = 1:	increase segment current by 12 mA
b7:	reserved

A segment is actuated (switched on) by making the associated bit '1'; a '0' switches it off again. This means that we are not limited to just displaying numbers 0 through 9: characters A through F are also possible, which is useful for making a hexadecimal readout.

Diskette ESS 1671 contains a demonstration file, LDIS.PAS, which contains the I^{2}C driver as well as the routines for controlling the A-D/D-A converter and the I/O port. LDIS.PAS is written in Turbo Pascal. Figure 5 shows the main procedure, which starts with moving the decimal point of the most significant digit to the least significant digit. Next, the program counts up from 0 to 9999, and starts again. The counter values appear on the display as well as on the PC screen. ■

References:
1. "I^{2}C interface for PCs". *Elektor Electronics* February 1992.
2. "ADC/DAC and I/O for I^{2}C bus". *Elektor Electronics* March 1992.

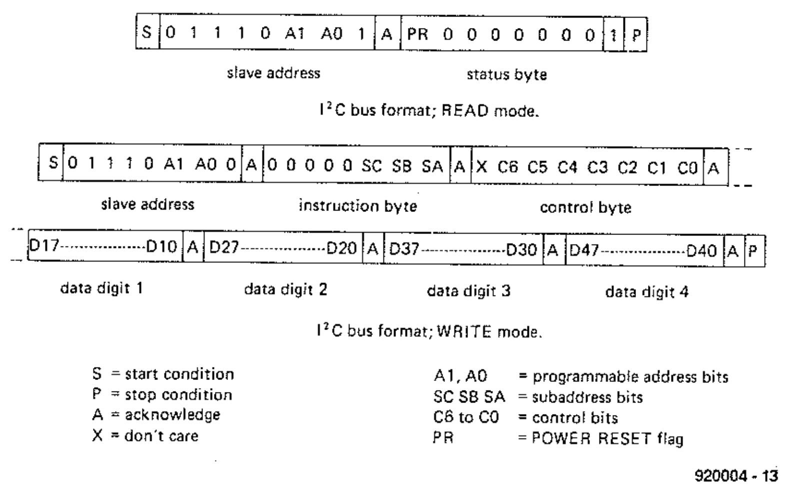

S 0 1 1 1 0 A1 A0 1 A PR 0 0 0 0 0 0 0 1 P
slave address
status byte
I²C bus format; READ mode.
S 0 1 1 1 0 A1 A0 0 A 0 0 0 0 0 SC SB SA A X C6 C5 C4 C3 C2 C1 C0 A
slave address
instruction byte
control byte
D17 D10 A D27 D20 A D37 D30 A D47 D40 A P
data digit 1
data digit 2
data digit 3
data digit 4
I²C bus format; WRITE mode.
S = start condition
P = stop condition
A = acknowledge
X = don't care
A1, A0 = programmable address bits
SC SB SA = subaddress bits
C6 to C0 = control bits
PR = POWER RESET flag
920004 - 13

Fig. 4. Sending commands to the controller is pretty simple if you make use of the I^{2}C driver contained on the floppy disk supplied for this project.

```
begin (* LedDisplayTest *)
  Start(Bus);                        {-Start communication on I2C-bus.}

  InitLedDisplayTest;

  Address(DisplayAddr);              {-After being addressed, the LED-driver
                                       expects an instruction byte.}

  Inst:=GetInstructionByte(1);       {-The byte following the instruction
                                       byte will be stored in the control
                                       register....}
  write(Bus,Inst);

  Ctrl:=GetControlByte(1);           {-Prepare loop.}
  Counter:=0;

  Repeat
    write(Bus,Ctrl);                 {- go!}

    with Digit do                    {-Copy counter to LED-display.}
      begin
        Split(Counter,D1,D2,D3,D4);
        write(Bus,DCode[D1],DCode[D2],DCode[D3],DCode[D4]);
      end;
    Screen(1);

    write(Bus,Du,Du,Du);             {-After these three dummy bytes have
                                       been sent, the LED-driver expects
                                       a control byte again....}

    delay(t);

    if Counter=0                     {-Shift decimal points.}
      then
        for Counter:=5 downto 0 do
          begin
            write(Bus,Ctrl,DP,Bl,Bl,Bl,Du,Du,Du); Screen(2); delay(t);
            write(Bus,Ctrl,Bl,DP,Bl,Bl,Du,Du,Du); Screen(3); delay(t);
            write(Bus,Ctrl,Bl,Bl,DP,Bl,Du,Du,Du); Screen(4); delay(t);
            write(Bus,Ctrl,Bl,Bl,Bl,DP,Du,Du,Du); Screen(5); delay(t);
          end;

    inc(Counter);
    if Counter>MaxCount
      then
        Counter:=0;
  Until keypressed;

  UnInitLedDisplayTest;

  Close(Bus)                         {-Stop communication on I2C-bus.}
end. (* LedDisplayTest *)

                                                  920004-14
```

Fig. 5. Main procedure in the display test program LDIS.PAS.