THE ELECTRONICS & COMPUTER MAGAZINE
MARCH 2000
£2·95

ELEKTOR ELECTRONICS

PC TOPICS:

- 32 inputs on RS232 port
- versatile LED display
- Java multimeter

midget MW radio

digital volume control

charge controller

for solar cells

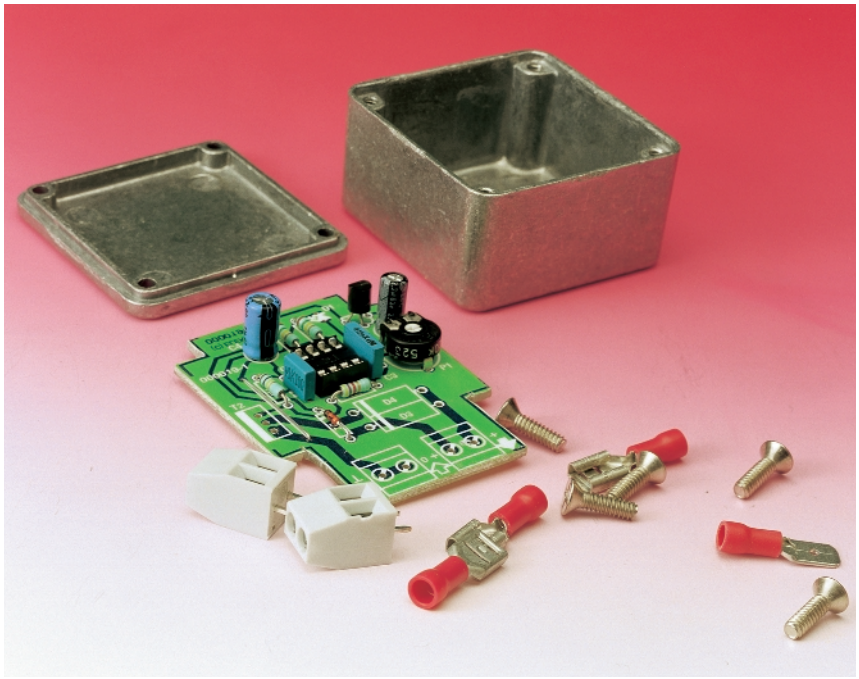
solar cell technology update

03
9 770268 451081

solar charging regulator

for panels up to 53 watts

A small solar power installation consists of at least three components, namely a solar panel, a storage battery and a charging regulator. The charging regulator limits the final charging voltage and also protects against reverse current flow.



Design by H-Tronic, Hirschau, Germany

Specifications of the solar charging regulator

Supply voltage	*12 V, from the battery*
Solar panel	*up to 53 W*
Operating current	*2.1 mA*
Voltage drop	*0.43 V at 3 A*
Temperature range	*–10 to +45 °C*
Short-circuit and reverse-polarity protected	

If you were thinking that nothing new has appeared in *Elektor Electronics* in the last ten years regarding solar power technology, this article is the proof!

In a self-contained solar power installation that can provide electrical energy even when the weather is bad or it's dark outside, an energy reservoir in the form of a lead-acid battery is indispensable. In order to prevent the battery from being discharged via the solar panel when the terminal voltage of the panel drops below the actual level of the battery voltage, reverse-current protection is necessary. In its most rudimentary form, such a 'solar current valve' is just a simple diode. A Schottky diode, which has a low forward voltage drop, is normally used to minimise losses.

Unfortunately, the terminal voltage of a 12-V solar panel is significantly higher than its nominal rated voltage when it is illuminated by strong sunlight, so it is not possible to avoid exceeding the fully-charged (terminal charge) voltage of the battery using only this single diode. If the voltage applied to the battery is too high, it produces gas, which reduces the lifetime of the battery and can also be dangerous, since the gas is explosive. A regulator circuit is thus necessary, in addition to the reverse-current protection diode, to limit the terminal charge voltage of the battery to 2.30 V per cell (equivalent to 13.8 V for a 12-V battery).

The regulator circuit presented here fulfils these two tasks — reverse current protection and voltage regulation — in an elegant manner.

SHORT-CIRCUIT CONTROL

In the circuit diagram of the regulator, shown in **Figure 1**, it's easy to identify the reverse-current protection. If the terminal voltage of the battery is higher than that of the solar panel, the Schottky diode D3 prevents any current from flowing from the positive terminal of the battery to the positive terminal of the solar panel, independent of the state of the rest of the circuit. In the reverse situation, the charging current has free access to the battery. The voltage drop across the diode is 0.43 V at a current of 3 A.

However, the solar panel current can also flow through D4 and T2 when transistor T2 is switched on. The transistor is driven by the opamp IC1, which is wired as a comparator. Transistor T1 and potentiometer P1 provide

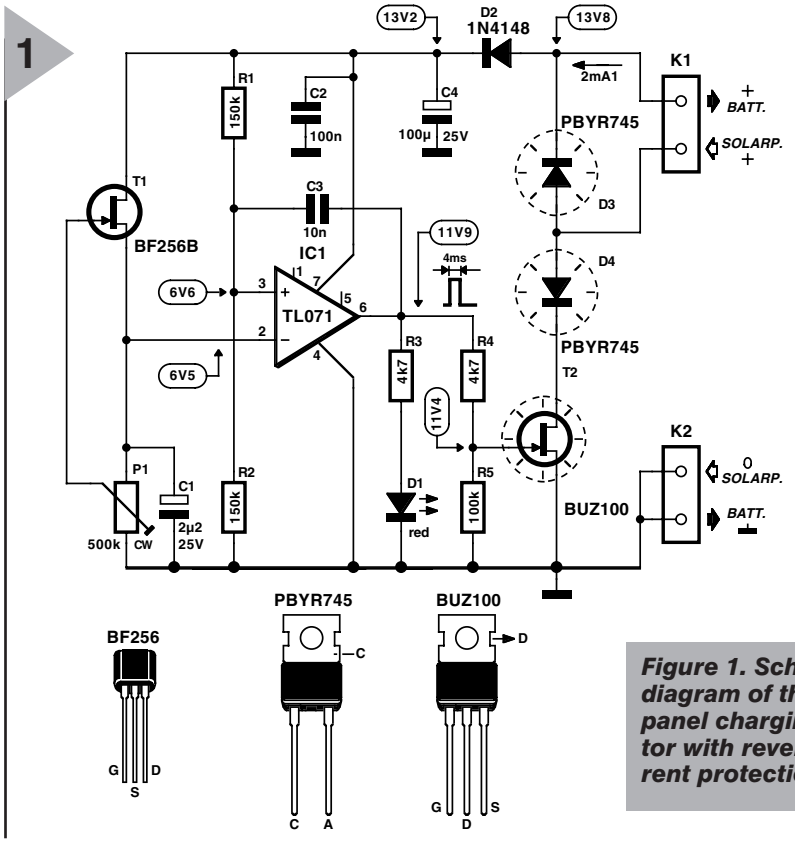


Figure 1. Schematic diagram of the solar panel charging regulator with reverse-current protection.

a reference voltage, which is filtered by capacitor C1. This voltage is set to roughly half of the terminal charge voltage of the battery. The opamp compares the reference voltage to the voltage at the junction of R1 and R2, which is half of the battery voltage less the 0.6-V drop of diode D2. The exact values are shown in the circuit diagram. If the battery voltage is less than the terminal charge voltage, the output of the opamp remains low and T2 is cut off. Led D1 is thus off, which indicates that the full solar panel current is flowing into the battery.

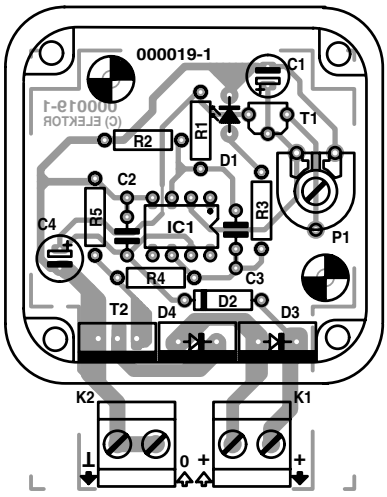
If the battery voltage rises above the terminal charge voltage, the comparator output changes to high (D1 on) and switches on T2, so that the output of the solar panel is short circuited. Since a solar panel represents a current source, which can deliver only a limited current even when it is strongly illuminated by the sun, this otherwise brutal form of shunt regulation is fully acceptable.

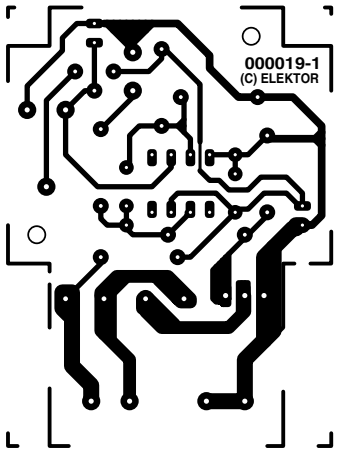
Nevertheless, an additional measure is used to minimise the power dissipation in T2 and D4, and thus avoid the need for a large heat sink. This is provided by capacitor C4, which produces a brief positive feedback pulse (lasting around 4 ms) to the opamp whenever it changes state. This significantly improves the switching behaviour of the opamp, so that the edges of the output signal are distinctly steeper.

The power dissipated by an n-channel MOSFET (such as the BUZ100) is the lowest when it is either fully on or fully cut off, or in other words when it passes either a high current or no current at all. In the 'analogue' region between these two extreme states, the power dissipation is much greater. The edges of the drive signal should therefore be as steep (and thus as short) as possible. This is precisely what C3 achieves.

When the charging current to the battery drops out, due the short-circuiting of the solar panel, the battery voltage drops slightly. This causes the comparator to switch states and allow the battery to be charged again. In practice, this means that the fuller the battery is, the faster the LED blinks.

Figure 2. The circuit board fits exactly into the recommended enclosure.





PACKAGING

The solar panel regulator can be mounted in any suitably large enclosure. The small aluminium enclosure that is specified in the components list is particularly 'chic', since it not only protects the electronics against the external environment but can also serve as a heat sink for the three power semiconductors. Only the edge of the circuit board that holds the terminals protrudes through a slot in the enclosure.

To construct the board, first mount all the components, except for the terminals, on the printed circuit board shown in **Figure 2**. Insert the two Schottky diodes and the power tran-

COMPONENTS LIST

Resistors:
R1,R2 = 150kΩ
R3,R4 = 4kΩ7
R5 = 100kΩ
P1 = preset 500kΩ

Capacitors:
C1 = 2µF2 25V radial
C2 = 100nF
C3 = 10nF
C4 = 100µF 25V radial

Semiconductors:
D1 = LED, red, high efficiency
D2 = 1N4148
D3,D4 = PBYR745 (Philips)
T1 = BF256B
T2 = BUZ100* (Siemens)
IC1 = TL071CP or TL081

Miscellaneous:
K1,K2 = 2-way PCB terminal block, raster 5mm
Enclosure: Hammond 1590LB (50x50x32 mm)
PCB, order code **000016-1**, see Readers Services page and Elektor website.

sistor as far as possible into the board. Now you're ready for a first test. Turn P1 fully to the left and then connect an adjustable power supply (set to 13.8 V) to the battery terminals. Next, turn P1 slightly past its middle position. If the LED illuminates, the construction of the board should be OK.

If the LED does not behave as it should, check the two input voltages of the comparator opamp, and make sure that the comparator works properly for both input conditions.

By the way, the BUZ100 is nothing more than an improved version of the good old BUZ10 that can theoretically dissipate up to 250 W. This is naturally not at all necessary, so that a BUZ10 or BUZ11 transistor can be substituted for the BUZ100 if the latter is difficult to obtain. Similar considerations apply to the Schottky diode. Any type that can handle at least 3 A can be used.

DRILLING AND FILING

You will have to make a slot in the small aluminium enclosure through which the 'external' portion of the circuit board can be passed. The slot should 32 mm long and 6 mm wide. The circuit board is laid out such that the three power semiconductors lie flat against the inner wall of the enclosure. Mark and drill the two holes for fastening the board, and then mount the board in the enclosure using 5-mm stand-offs. Next, carefully measure and mark the holes for fixing the three power semiconductors, above the slot. Remove the board, drill 2.5-mm holes in the marked locations and tap them for an M3 thread.

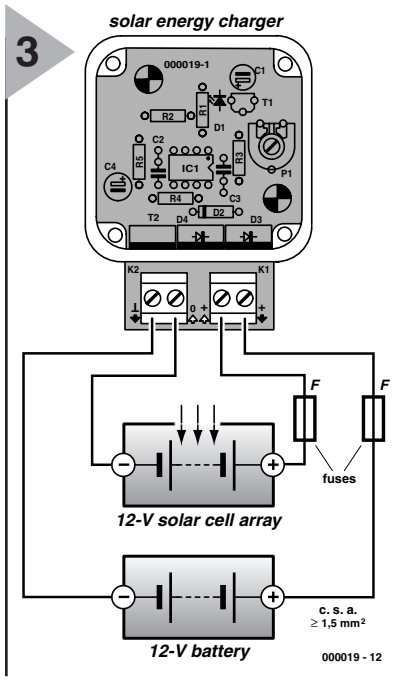
The two Schottky diodes and the MOSFET transistor must be insulated from each other and from the enclosure. You can use mica or plastic insulators that are cut so that they fit next to each other and allow the lid of the enclosure to be closed.

Now remount the circuit board, and then check that the holes in the cooling tabs of the three power semiconductors exactly line up with the drilled holes. If they do not, you must either file down the stand-offs or lengthen them using thin shims. Only after all five holes line up can you tighten all the screws without generating mechanical stresses. Don't forget to use heat-sink paste on both sides of the insulators, and make sure that there is an insulating washer (with a collar) between the screw and the heat sink tab of each semiconductor. Be sure to check the results of your work using an ohmmeter! After everything has been proven to be properly insulated, mount the terminals on the circuit board.

Before you close the lid of the enclosure, you should adjust P1 while the regulator is connected to a real solar power installation (see **Figure 3**), with the regulator located as close as possible to the battery. This is necessary to take into account the voltage losses due to the terminal connections and the cables, which are significant even with relatively thick wires (diameter 1.5 mm or more), and thus achieve optimum performance of the installation. Do not forget to provide suitable fuses for both circuits, since a current of more than 100 A can flow in case of a short circuit!

(000019-1)

Figure 3. Integrating the regulator into a solar power installation.



Dimensioning a self-contained solar power installation

The individual components of a solar power installation must be properly dimensioned in order to optimally exploit the available energy. The ***current consumption*** *(including the losses in the solar regulator), together with the time behaviour of the current consumption, essentially determines the planning of the installation. Based on this information, you can calculate the average daily or weekly current consumption. In small solar power installations, with an average consumption of at most 3 kWh/day and a peak load of around 3 kW, you can usually assume a system voltage of 12 V. With heavier loads, a higher system voltage is recommended in order to keep the wire thickness and/or the wiring losses within reasonable limits. A voltage of 24 to 48 V, or even up to 230 V, can be obtained by connecting several solar panels in series.*

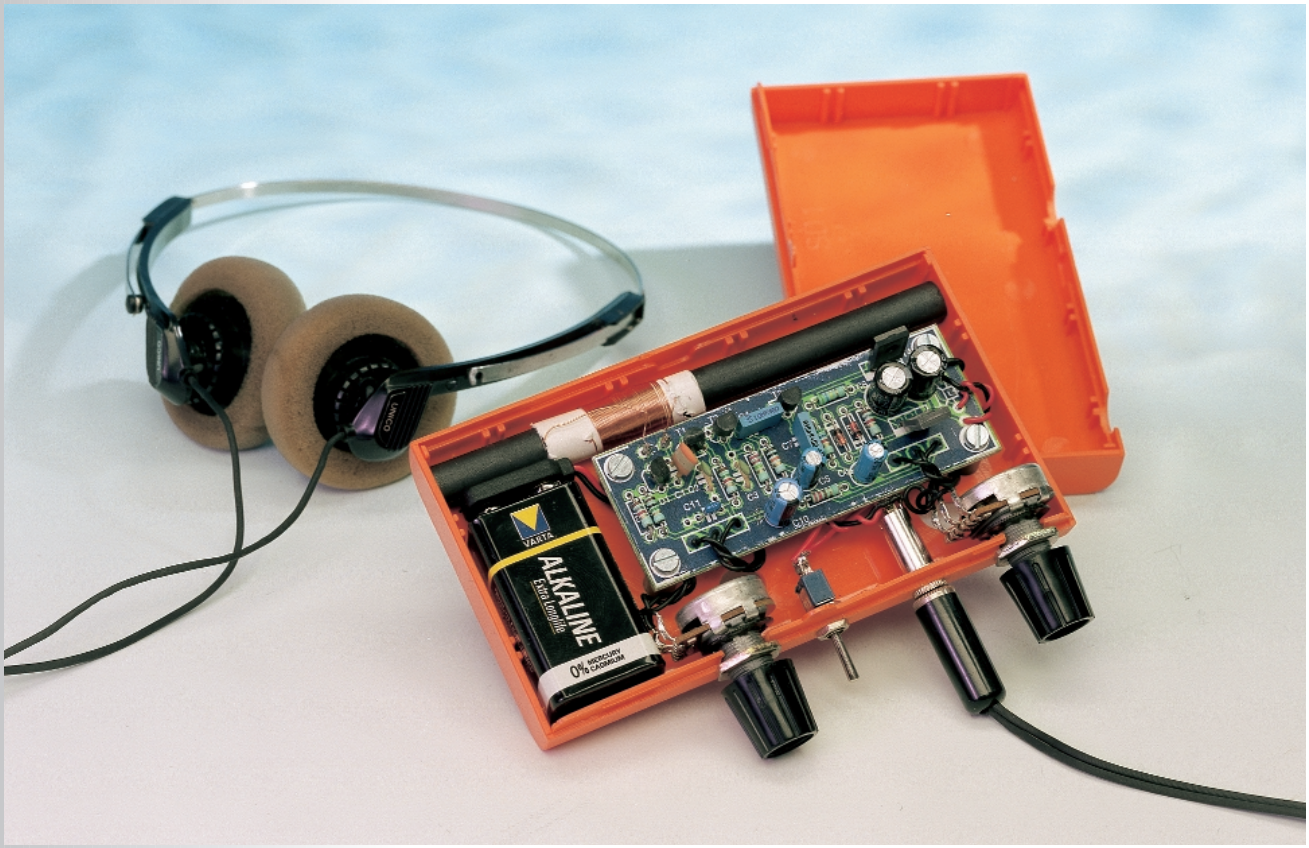
The next step is to estimate ***the required size of the solar generator****. The orientation of the solar panels and their angle with respect to the horizontal, the efficiency of the solar panels and naturally the geographic location of the installation (since the sun shines more often and more intensely in some places than in others) all enter into this calculation. The result is either the required nominal current capacity or, in the case of a given type of panel, the required number of panels (connected in parallel).*

Finally, you must determine the ***capacity of the storage battery****, which compensates for the climatic variations in the available solar energy. There are two criteria here, namely the already-determined average daily or weekly power consumption in ampére-hours and the number of days for which the installation must be able to supply power from the battery alone, which strongly depends on the individual application. The allowable discharge capacity of the battery must be taken into account here.*

In addition to these main criteria, there are several ostensibly secondary considerations that should be taken into account, such as the peak demand, the voltage reserve at high operating temperatures, the mechanical preparation of the modules, the available and required amount of space, costs, building code restrictions, subsidies and so on.

If you are seriously interested in the construction of a solar power installation, it is strongly recommended that you first read some introductory literature. There are also a number of more or less useful PC programs on the market to help in calculating the requirements for the components of a solar power installation. A good introduction to the subject is provided by the CD-ROMs Solar Power and Solar Craft from the Energiewende Verlag (www.solarenergie.com). These include a multimedia introduction, basic information, address directories, tips and tricks and simple construction information. Solar Power also includes a calculation program, while Solar Craft includes an interactive calculation program that guides you step by step through the construction of a domestic installation.

midget medium-wave receiver



The tiny medium-wave (MW) receiver described in this article is an ideal construction project to while away the dreariness of a rainy winter's day. Solder a couple of standard components onto a tiny printed-circuit board, connect the finished board to a battery, and Bob 's your uncle.

Design by G. Baars

Nowadays there are few construction projects or, indeed, commercially available equipment, that use discrete components only. Most apparatus encountered today has at least a processor and/or a number of integrated circuits. Although such equipment has the great advantages of reliability and simplicity, some readers may find the present project in which they can potter about without running into great difficulties a welcome surprise.

The midget radio receiver consists of five common-or-garden transistors, one inductor and a handful of standard passive components. Most readers may well find that they have all the necessary components to hand.

Anticipating the question of some readers whether five transistors are sufficient to build a reasonable receiver, it may already be said that actually

only two of these transistors are used in the radio-frequency (RF) stages proper. The other three are needed for the integral mini output amplifier.

Clearly, miracles cannot be expected from two transistors. Nevertheless, it is not the case that a very large antenna is needed to make a tiny signal audible. The midget receiver is perfectly capable of receiving a number of local medium-wave signals without an external antenna. The tuning inductor wound on a ferrite rod picks up these broadcasting stations without external aid.

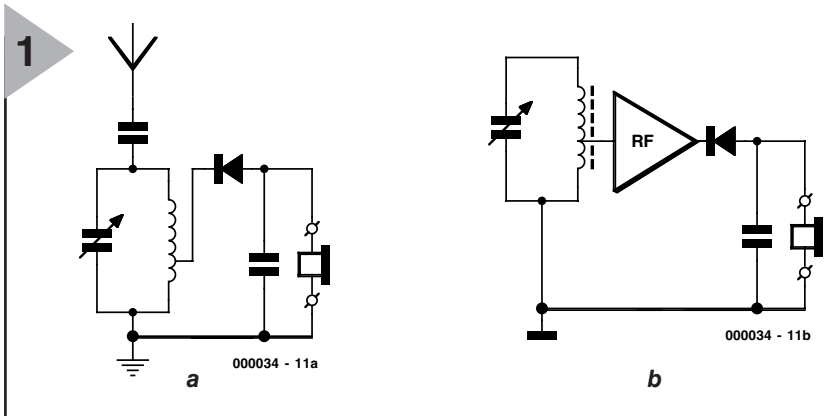


Figure 1. A simple diode receiver (1a) may be improved considerably by the addition of an RF amplifier (1b).

DIRECT CONVERSION RECEIVER

The design is that of a direct conversion receiver. Most constructors will know that there are basically two types of receiver design: the superheterodyne and the direct conversion receiver (also called Tuned Radio Frequency – TRF – receiver).

The superheterodyne receiver, normally called a superhet, is far and away the more popular of the two. It is a design in which a number of tuned circuits are used to increase the selectivity and the received signal is mixed with that of a tuned oscillator. The RF input circuit and the oscillator circuit are tuned in synchrony so as to hold the difference between the frequencies in the two circuits constant. This difference frequency is called the intermediate frequency – IF.

The fact that the output of the mixer, that is, the IF, has a constant value, has the advantage that the stages following the mixer can be filtered without the need for the relevant inductance-capacitance – *LC* – circuits to be tuned for each and every different received signal.

The foregoing makes it clear that a superhet is not a suitable design for a small and simple receiver. However, a direct conversion design is perfectly suitable. In this type of design the received signal is selected, often amplified, and then demodulated. This means that there is no need of an oscillator or conversion to an intermediate frequency.

Figure 1a shows the circuit diagram of a TRF receiver in its simplest form. In this receiver the wanted frequency is selected by a tuned *LC* circuit from the spectrum of frequencies picked up by the antenna. The selected signal is demodulated by the diode and the resulting audio-frequency (AF) signal is fed to a pair of headphones.

A simple receiver as in Figure 1a needs a fairly large antenna. Note that the tuned *LC* circuit is connected between the antenna and earth. So as not to load the circuit too heavily (which would adversely affect the selectivity of the circuit), the detector is normally connected to a tap on the inductor.

The great advantage of the simple receiver is that it does not need a supply voltage. On the other hand, it is not very sensitive and its performance depends largely on the antenna. These drawbacks may be negated to an appreciable extent by the addition of an amplifier. If this is not done, the level of the incoming wanted signal must exceed the forward bias of the diode, that is, the voltage needed to be applied to the diode to make it conduct. In the case of a germanium diode as used here, the forward bias is 100–200 mV.

The addition of an RF amplifier is shown in **Figure 1b**. Such a receiver may have a sensitivity which makes reception with an antenna formed by an inductor wound on a ferrite rod possible.

It must be understood that a direct conversion receiver cannot be compared as regards selectivity and sensitivity with a superhet. After all, the receiver in Figure 1b has only one tuned circuit and one RF amplifier. Nevertheless, it also has some advantages over a superhet. For instance, a TRF receiver is simple to construct,

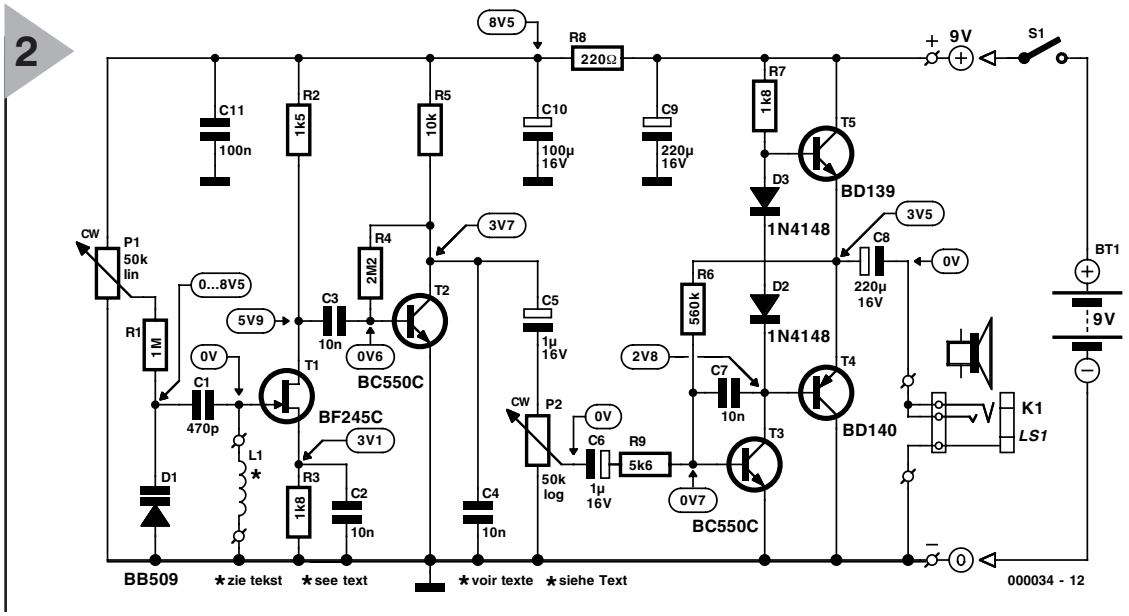


Figure 2. Circuit diagram of the Midget Medium-wave Receiver which is based on only five transistors.

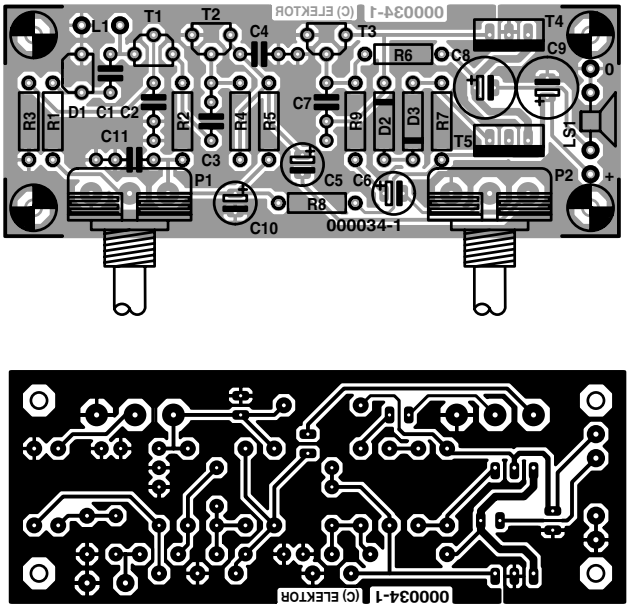


Figure 3. Printed-circuit board for the Midget Medium-wave Receiver.

cheap to make, not critical to set up, does not require much tuning, is free of interfering whistles, and produces good-quality sound.

CIRCUIT DESCRIPTION

The circuit diagram of the midget receiver is shown in **Figure 2**. The receiver proper, that is, the RF section, is to the left of preset P2. Although this is still tiny, it has rather more components than the receivers in Figures 1a and 1b.

The tuned RF circuit consists of inductor L1, capacitor C1, and diode D1. To reduce costs, C1 is not a variable capacitor, but a variable-capacitance (varicap) diode. The capacitance of such a diode varies in accordance with the voltage applied across the diode. That voltage is varied with P1, so that this potentiometer enables the circuit to be tuned as required.

Note that inductor L1 wound on the ferrite rod has no taps unlike those in Figures 1a and 1b. As mentioned earlier, such a tap to which the demodulator or RF amplifier is connected reduces the loading of the inductor and so prevents the *Q*(uality) factor and thereby the selectivity of the circuit being degraded. This is particularly important in the case of a receiver with only one tuned circuit (which determines the overall selectivity of the receiver).

There is, however, a drawback to using a tap and that is that the signal applied to the RF amplifier or demodulator is not the maximum available. In the midget receiver the signal is taken from across the inductor but, since preamplifier T1 is a field-effect transistor – FET – which has a very high input impedance, the inductor is loaded only lightly.

Although T1 provides some gain, the main amplification is provided by

COMPONENTS LIST

Resistors:
R1 = 1MΩ
R2 = 1kΩ5
R3,R7 = 1kΩ8
R4 = 2MΩ2
R5 = 10kΩ
R6 = 560kΩ
R8 = 220Ω
R9 = 5kΩ6
P1 = 50kΩ linear potentiometer
P2 = 50kΩ logarithmic potentiometer

Capacitors:
C1 = 470pF
C2,C3,C4,C7 = 10nF
C5,C6 = 1μF 16V radial
C8,C9 = 220μF 16V radial
C10 = 100μF 63V radial
C11 = 100nF

Inductor:
L1 = 50 turns 0.3mm dia. (SWG30) enamelled copper wire on ferrite rod (10x100mm)

Semiconductors:
D1 = BB509 *
D2,D3 = 1N4148
T1 = BF245C or BF256C
T2,T3 = BC550C
T4 = BD140
T5 = BD139

Miscellaneous:
K1 = 3.5 mm jack socket
S1 = on/off switch
Bt1 = 9-V battery with clip-on connector
Ls1 = earpiece, headphones or 8Ω loudspeaker
Enclosure: e.g., Conrad Electronics 52 09 93 (123x30x70mm)
PCB, order code **000034-1** (see Readers Services page)

*) Available from. Barend Hendriksen Elektronica, P.O. Box 66, NL-6970-AB Brummen, The Netherlands. Tel. (+31) 575 561866, fax (+31) 575 565012. Web site: http://www.xs4all.nl/~barendh/

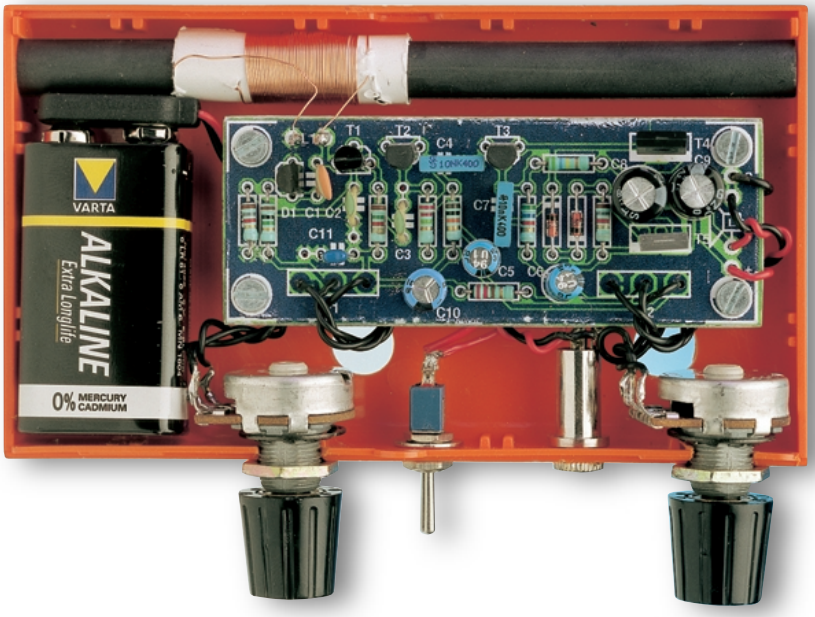


Figure 4. The prototype receiver fits exactly in the chosen enclosure.

transistor T2. This transistor also functions as the demodulator, which is possible since its base-emitter junction is a p-n diode. The residual components of the carrier (signal) frequency are shorted to earth by capacitor C4. The demodulated signal is applied to output amplifier T3–T5 via capacitor C5 and volume control P2

It would have been feasible to use an integrated circuit (IC) for the output amplifier, but discrete transistors were chosen to keep the overall design in line with the RF section. The transistors do not add to the dimensions of the receiver. Transistor T3 functions as a voltage amplifier cum driver, while T4 and T5 form a simple push-pull output stage.

Diodes D2 and D3 compensate for the forward voltage of T4 and T5 so that crossover distortion is an absolute minimum. In quiescent operation, the output stage draws a current of only a few milliamperes.

The output stage is capable of not only driving a pair of headphones (ear pieces in parallel), but also a small 8 Ω loudspeaker. Although the maximum power output is about 1 watt, this is sufficient to fill the average living room with quite enough sound.

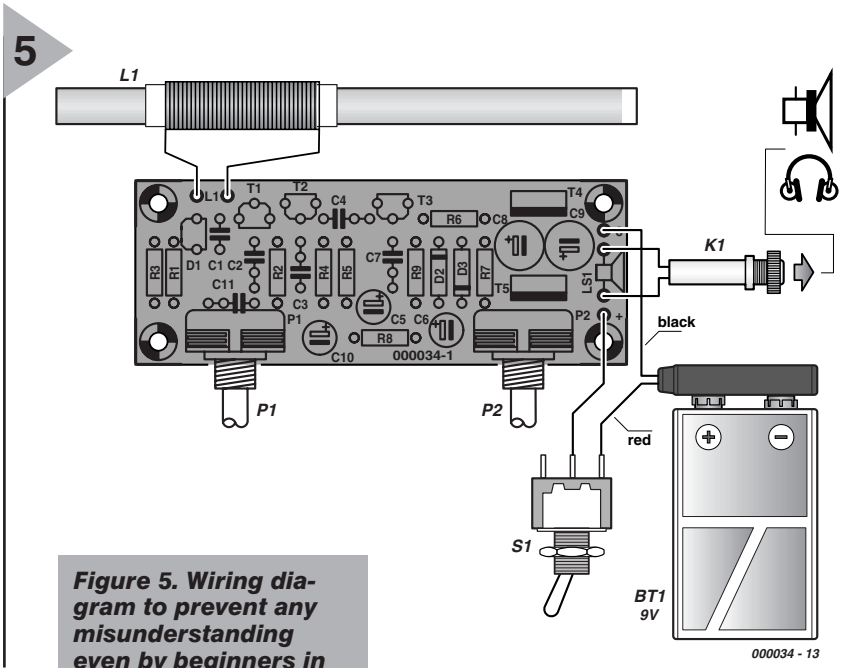


Figure 5. Wiring diagram to prevent any misunderstanding even by beginners in electronics.

POWER SUPPLY

The receiver needs a power supply of 9 V which, since the current drain in normal circumstances does not exceed 30 mA, enables a primary battery to be used. In normal use, the battery will have a fairly long life.

If a mains adaptor is chosen as the power source, it needs to be regulated because the voltage for varicap P1 is derived directly from it.

The absolute level of the supply voltage is not terribly critical. The prototype worked fine with a supply of 7 V, while one of 12 V offers the advantage of improving the sensitivity of the receiver to some extent.

CONSTRUCTION

The midget receiver is best built on the printed-circuit board shown in **Figure 3**. This board is a compromise between small dimensions and a convenient, manageable layout. The latter ensures that populating the board is straightforward.

Start completing the board with the passive components and then the transistors, all the time with constant reference to the illustrations and the parts list. The only item that is not soldered onto the board is inductor L1. This is mounted alongside the board with its two terminals linked to terminations L1 on the board.

Inductor L1 has to be made at home or in the workshop, but this is not difficult. It is wound on a 10–12 cm long ferrite rod with a diameter of 10 mm. Make a cylinder of thin plastic or stiff paper with a diameter that enables it to be just moved to and fro along the ferrite rod. Close-wind 50–55 turns of 0.3 mm diameter enamelled copper wire onto the cylinder. This should be done with great care, because the quality of the inductor will determine the performance of the receiver. The terminals should be fixed in place with sticky tape or by feeding them through tiny holes drilled through the cylinder.

The inductor is the only device in the receiver that needs to be aligned: in this case to the medium-wave broadcast band. Depending on the construction, the self-inductance of the inductor may vary from case to case. This variation may be compensated by sliding the coil across the ferrite rod: when it is at the centre, the self-inductance is a maximum, while at the ends it is reduced somewhat. Measuring instruments are not needed for the alignment. Use another MW receiver as reference and slide the coil until the MW bands on the two are more or less identical. The MW range extends from about 530 kHz to 1605 kHz.

If sliding the coil does not allow the range to be obtained, the number of turns must be adapted. If the range does not extend to 530 kHz, a few turns need to be added; if it does not go up to 1605 kHz, a few turns should be removed. Since removing turns is much simpler than adding some, it is wise by commencing with 55 turns.

ENCLOSURE

When the board has been completed and the receiver works satisfactorily, the board should be fitted in a suitable enclosure. This should be of hard plastic (ABS) or wood, because metal screens the ferrite antenna from the outside world. Ideally, the enclosure should measure 123×30×70 mm and such cases should be available in many model shops or at electronic retailers. The prototype is fitted in one that is of almost exactly the right dimensions (see **Figure 4**). Unfortunately in this, and perhaps also other makes, it proved necessary to fit the potentiometers to the enclosure rather than on the board. If the ferrite antenna cannot be clamped into the case, use two suitable packing blocks at the ends.

Whether a pair of headphones or a small loudspeaker is used is a matter of personal preference. The prototype is used with headphones since the enclosure was just not large enough to house even a tiny loudspeaker. Constructors wishing to use a loudspeaker may find that those commercially available for use with small portable receivers (Walkman) are ideal. These are hardly any more expensive than the constituent parts of a loudspeaker and often give surprisingly good sound reproduction.

CONNECTING UP

Although the various terminations are clearly marked on the board, for convenience's sake **Figure 5** shows exactly how the battery, loudspeaker if used, and ferrite antenna should be linked to the board. This may be of particular help to beginners in electronics.

[000034-1]

Text (Dutch orginal): S. van Rooij.
Design editing: K. Walraven.

BASIC Stamp programming course (7)

Part 7: subsumption programming

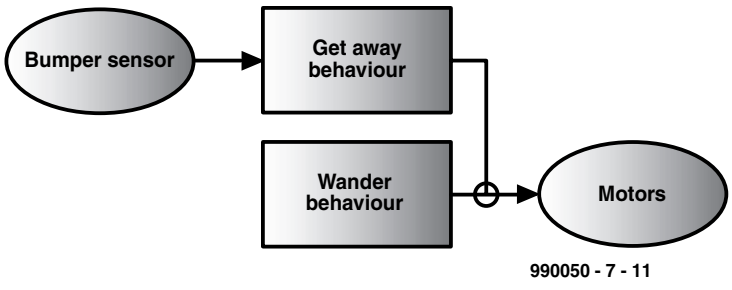


Figure 22. Simple representation of procedures based on subsumption.

By Dennis Clark

SUBSUMPTION ARCHITECTURE AND ROBOTIC BEHAVIOUR

To subsume a task is to supersede it with a higher priority task. When we speak of subsumption in robotic programming we are describing the process by which one behaviour subsumes, or over-rides another based on an explicit priority that we have defined. This behavioural architecture was first described by Dr. Rodney Brooks in his article "A robust layered control system for a mobile robot" in *IEEE Journal of Robotics and Automation*, RA-2, April, 14-23, 1986. The *Brooksian* ideal subsumption architecture does not require that any behaviour know anything about any other behaviour. A robot programmed in this manner responds reflexively to its environment using simple behaviours. What this also means is that new behaviours can be added to a robot without changing *any other behaviour*. This makes enhancing a robot's programming very simple. An example of subsumption programming in our little BoE-Bot would be, for instance, if we had a behaviour that simply randomly chooses a direction to travel and a duration of time to travel in that direction. When complete, that subroutine (behaviour now) would choose a new direction and duration. But we don't want to get stuck up against a wall or table leg, so we add a behaviour that looks at a bumper and if we run into something will make our robot back up and turn away. We want the *bumper* behaviour to have priority over the *wander* behaviour, so it will subsume that behaviour and take over the control of the wheel motors to get itself away from the wall. Because the *bumper* behaviour has a higher priority than the *wander* behaviour we can be assured that our little BoE-Bot can get away from obstacles that block its path. Further, if *wander* had set a very long duration on the last direction that it wanted to go, when *bumper* was done, the *wander* behaviour would take up the control again, continuing on its merry way as if nothing had happened.

Because each behaviour is independent, and many simply react to the robot's environment, we can build up a set of behaviours whose interactions with each other are *not* programmed, and we will begin to see combinations of behaviours unforeseen! This is called *emergent behaviour* because we didn't plan it, it came about because of the robot's interaction with its environment. When emergent behaviour is allowed to occur, our robots appear to take on a life of their own and stop acting as if they just do the same thing over and over again. More importantly in the *real* world, some behaviours can be programmed to do a certain task, or retrieve an object, and not have to be concerned with the moment-to-moment job of avoiding falling into a hole or running into a rock.

Our simple sets of behaviours we just discussed can be graphically displayed by a subsumption network diagram. An example of our simple wander/bumper behaviours is shown in **Figure 22**. The ovals on the left are inputs, the rectangles in the middle are behaviours and the ovals on the right are outputs.

Where a line from a higher priority behaviour intersects a line from a lower priority behaviour to its output actuator (in this case, Motors) the higher priority behaviour is said to have *subsumed* control from the lower priority behaviour. This is shown on the network above by *Get away*'s line intersecting *Wander*'s line with a circle around the junction. There can be multiple intersections in multiple locations on the direction lines that can show the subsumption logic very clearly. Feel free to innovate how you show your network for your robot, there are many ways to build a subsumption network diagram.

WHERE DO WE GO FROM HERE?

Get used to these diagrams and ways of thinking. In the chapters ahead we will be using these ideas and expanding on the simple diagrams that we have seen so far. In each section I will break our work down into easy to see pieces, such as variables needed, I/O

Listing 10.

```
'Servo routine cons and vars
LEFT con 15 'port 15,left motor
RIGHT con 3 'port 3,right motor
SACT con 5 'times through the
loop
drive var word 'both sides in var
ldrive var drive.byte1 'left side is here
rdrive var drive.byte0 'right side here

aDur var byte 'duration counter
```

```
act:'servo controller subroutine
   if aDur > 0 then aDec
      aDur = SACT 'do state 1
      pulsout LEFT,lmotor * 10
      pulsout RIGHT,rmotor * 10
      goto aDone 'state 1 done
aDec:
      aDur = aDur - 1 'do state 2
aDone:

return
```

ports used and new instructions. I will also show the process by which the state machines are defined and our subsumption network diagrams can be used to understand our behavioural priorities.

Phase 1: making the BoE-Bot move, and the Finite State Machine

A robot that doesn't do anything isn't very interesting. Now that you have modified your servos, attached them to Stamp II I/O ports and have checked their operation, we will make our BoE-Bot wander randomly around its environment. To implement this simple functionality we need to have two modules: One that determines a direction and a duration to go in that direction and another to send commands to our wheels.

We will call these modules *wander* and *act* respectively. *Act* simply operates the motors. It isn't really a behaviour, its an output and will be labelled on our diagram as *motor*s. *Wander* is a behaviour so it will appear on our subsumption diagrams as a behaviour.

The State Machine and Stamp II Code for Act

We know that to get our servos to turn the wheels we need to output a pulse whose width needs to be approximately 1 ms to 2 ms long, with 1.5 ms being the stop position. We also know that this pulse needs to be repeated every 20 ms to 30 ms for our servos to continue moving. This suggests that we will have two operations that we will need to program. One will be 'output the pulse', the other will be 'wait for 20 ms' after which we go back to state 1. We have two servos, so we could say that we have three operations, output left servo pulse, output right servo pulse and wait. To keep this module simple we will limit it to two states, one to output the pulses and one to delay before the next pulse time. Since we know that we need to repeat the pulse every 20 ms to 30 ms and we know that we don't want to spend all of our time in the *act* module waiting for this time to elapse, we therefore know that we will be exiting and entering this block of code several times before the full set of state transitions occurs. We know that each instruction takes about 250 μs to execute for the Stamp II, this tells us about how long each module that we will be calling will take to run, we count up the instructions and multiply by 250 μs. For now we'll guess and say that it will take 5 passes through state 2 for 20 ms to elapse. As we write other modules we will be revising this estimate, but this is a good starting point for now. Let's graph our state machine so that we can understand what we want, and maybe even see any mistakes we made in our logic!

Before we can build our state machines it is helpful to define all of the actions that we wish to have our behaviour or output function perform. Each of these functions can be grouped by their activities into an individual state. Here is a way to define a motor control function, which we saw above as the subroutine *act*.

```
State 0
 Output left servo pulse value
 Output right servo pulse value
  Set the number of iterations
(aDur) = 5
State 1
 Decrement aDur
 If aDur = 0 then go to state 0
```

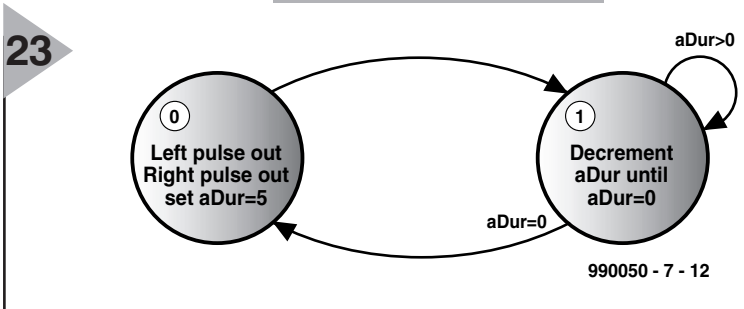
We now have a detailed list of actions that need to occur in our state machine for *act*. We can now draw a state diagram that has the needed detail, such as **Figure 23**.

This state diagram tells us when each transition is taken and exactly what each state is doing. The transition from state 0 (on the left) and state 1 always occurs, there is no condition. However, the transition from state 1 (on the right) to state 0 occurs when aDur = 0. Notice that state 1 has a loop that exits and turns back on itself. This indicates that this state *iterates* on itself, in our case, this means that we will enter this part of the module several times before it is complete. Look at this diagram carefully. It shows us very clearly what our *act* module must do, and in what order it must do it. Think about how it would look if we were to output the left motor pulse in a different state than the right motor pulse, how would that look? For our purposes, we can say that each circle (state) is a place where we enter the module in our program and decide what to do next. You should now be starting to see how powerful this concept is for us!

The code that implements our *act* module is shown in **Listing 9**. I will show the variable declarations on the left and the actual code on the right from now on. To make a program easier to read and to modify, you should make liberal use of *con* statements and not use "magic numbers" embedded in your code. This is how I will show our programs. Remember that everything that follows a ' is a comment, not an instruction. I will explain why we multiply the lmotor and rmotor values by 10 in the next phase.

If we are to write a large program we need to conserve variable space, the PBASIC language has some very clever ways to use that variable space. The *pulsout* instruc-

Figure 23. State diagram for the 'act' procedure.



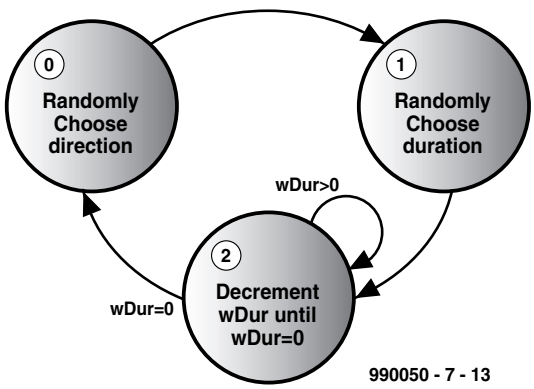


Figure 24. State diagram for the 'wander' procedure.

tions output a pulse to our servo motors of the proper width for the speed and direction that we want. The *drive* variable is a word which is two bytes, a byte0 and a byte1. You will see why this is a useful way to represent the motor drive variables when we design the code for the *wander* module in the next phase!

PHASE 2: MAKING THE BOE-BOT WANDER

All we need *wander* to do is randomly choose a direction and a duration to go that direction. This module will be as easy to design as the *act* module, and it will demonstrate the ease with which we can add new behaviours, and how these behaviours can remember what they are doing. The first step in designing our form of a finite state machine is to write down all of the steps that are to be taken and all of the transition functions that need to be tested. After we have our list, the state machine can be drawn.

Here is our action list for *wander*:

State 0
  Choose a direction to go by using the *random* function (wDir)
  Go to State 1
State 1
  Choose a duration to go by using the random function (wDur)
  Add some minimum time to the duration so its not too short
  Go to state 2
State 2
  Decrement wDur
  If wDur = 0 then go to state 0

The state machine diagram for our *wander* behaviour is shown in **Figure 24**.

Some of the arrows do not have transition labels attached to them. When a state machine automatically transitions from one state to another and does not need to make a decision a transition function is not needed.

The purpose of the *wander* module is to randomly choose a direction for our robot to go and a random time duration for it to take going there. The *random()* instruction plays an important role in the *wander* logic. *Random* uses a *word* variable type and needs a 'seed' to set up its return value. We will just use the last return value it gave us as the seed for the next one. We will also use a *mask* to only get numbers in a certain range; a small range for the direction changes, a much larger one for the duration. Another interesting feature is the *lookup* instruction. This instruction uses a *word* sized variable, because we break the *drive* variable into a *byte0* and *byte1* variables, we can get both the left and right motor values from a single lookup! You'll see how this is done in the **Listing 10**, and you'll see how this makes it very simple for us to change what the robot will do.

On the left all of the useful motor control commands are listed with a **$** in front of them. This means that those numbers are hexadecimal, I have used hexadecimal (HEX) numbers because its easy to see the left and right halves of them (byte1 and byte0) that we have assigned as the left and right motor values. Of course, this means that we need to understand HEX in order to know what is going on. In HEX, each digit represents a number from 0 to 15, A, B, C, D, E, and F represent the numbers 10, 11, 12, 13, 14, 15 respectively. The digit to the far right is the '1's digit, just like in our decimal system, but it can be from 1 to 15. The next digit to the left is the '16's digit, multiply any number you see there by 16, then add the number in the '1's digit place to that to get the decimal number. Our numbers above are 4 digits, but remember, that is just because we are putting both the left and right motor values in at once, we can look at the two left digits as separate numbers from the two digits to the right. This means that the HEX number $1C = 16+12, or 28. Remember in the last section when I said I'd explain why *act* multiplies the ldrive and rdrive number by 10? Now is the time for that explanation. A byte can only hold values from 0 to 255, our servos need to be sent numbers from 500 to 1000. To get that range, we use 50-100 (which are numbers less than 255) for each motor speed setting, then multiply by 10 to get values from 500 to 1000. This doesn't represent all possible values, but this really doesn't matter when driving modified servo motors as wheels.

Listing 11

```
'Servo drive commands
fd       con      $6432      'forward
rv       con      $3264      'reverse
st       con      $4b4b      'stop
tr       con      $644b      'turn right
tl       con      $4b32      'turn left
rr       con      $6464      'rotate right
rl       con      $3232      'rotate left
'wander values
wstate   var byt  e          'FSM status
wDir     var      word       'wander value
wDur     var      byte       'wander duration


wander:
branch wstate,[wcDir,wcDur]
'This is state 2
wDur = wDur – 1
if wDur > 0 then wDone1
   drive = wDir    'get direction
   wstate = 0      'reset state
wDone1: 'completed
   return
wcDir:                          'choose direction
   random seed     'random direction
   i = seed & %111 'mask off for 0-7
   lookup i,[tr,fd,fd,fd,rr,fd,fd,tl],wDir
   'choose direction
   wstate = 1      'next state
   return
wcDur: 'choose duration
   random seed     'randomize
   wDur = (seed & %111111) + 20
   'mask for 64 and add 20 for more time
   wstate = 2      'next state
   return
```

We now have almost all we need to make our robot wander aimlessly through a room. Almost. Our *wander* and *act* functions are subroutines, this means that something has to call them. The loop defined by *main* and *goto main* is our "brain" for our robot. All behaviours are called from here. **Listing 11** is the full program that has all of the variables and the subroutines as well as the setup and main run loop to show you how it all fits together. Type this into the Stamp II programmer and watch your little robot wander around the room for a while. Make sure it doesn't run into anything! In next month's instalment we'll teach it to avoid running into the wall.

Note the *set up for running* section in the code. We need to make sure that the *wander* behaviour starts out in the correct state, so, we set that state here. Now, we need to make sure that *wander* and *act* are called regularly, so these two subroutine calls are placed in a loop starting at *main*. This loop will be a very important feature of our code when we start adding new behaviours to our robot!

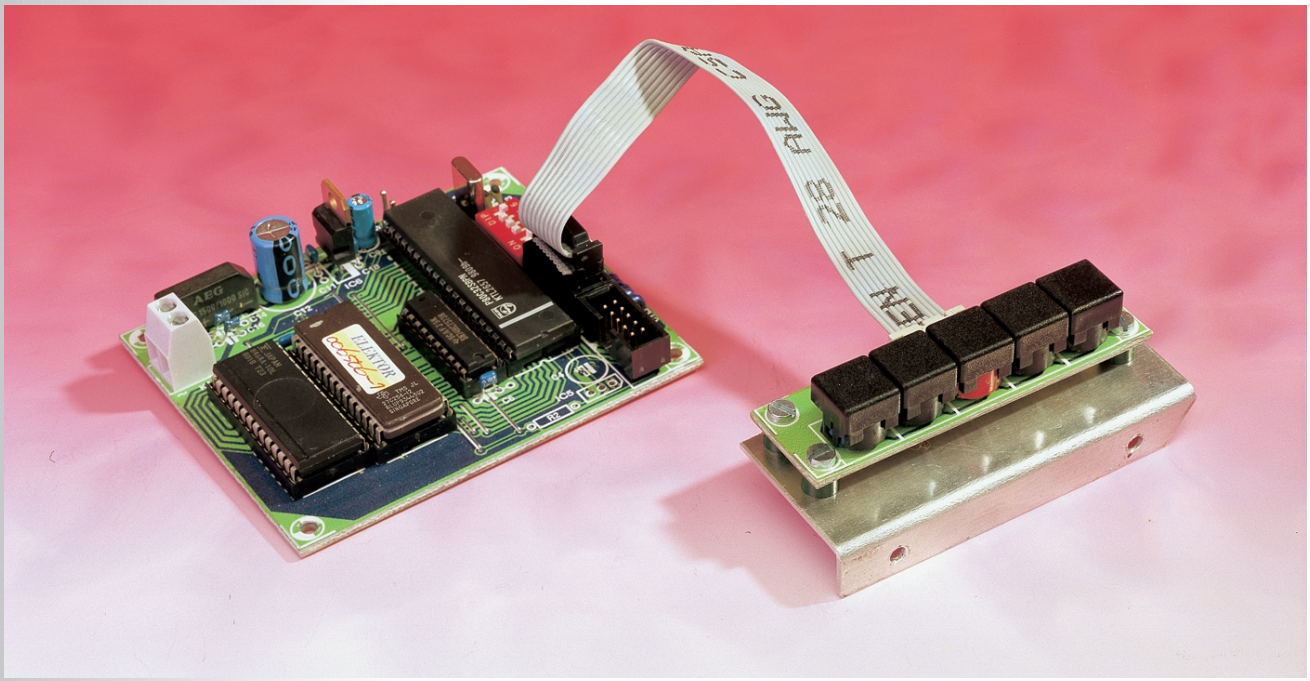
(990050-7)

Listing 12

```
'Generic values
I        var      byte                    'loop counter, whatever
Tmp      var      word                    'temporary holder
Seed     var      word                    'random number seed
'These are for the servo routines
LEFT     con      15                      'left wheel port
RIGHT    con      3                       'right wheel port
SACT     con      5                       'times through act routine
Drive    var      word                    'wheel command combo
Ldrive   var      drive.byte1             'left wheel command
Rdrive   var      drive.byte0             'right wheel command
ADur     var      byte                    'duration of pulse left
'Servo drive commands
fd       con      $6432                   'forward
rv       con      $3264                   'reverse
st       con      $4b4b                   'stop
tr       con      $644b                   'turn right
tl       con      $4b32                   'turn left
rr       con      $6464                   'rotate right
rl       con      $3232                   'rotate left
'wander values
wstate   var      byte                    'shared byte
wDir     var      word                    'wander value
wDur     var      byte                    'wander duration
'set up for running
wstate =0                                 'initial wander state
main:                                     'this is the main activity loop
         gosub wander
         gosub act
goto main
'=======================================
'Behaviours follow
'=======================================
wander:                                   'randomly wander around
         branch   wstate,[wcDir,wcDur]    'state 2 immed. follows
 wDur = wDur - 1
         if wDur > 0 then wDone1
                  drive = wDir            'get direction
                  wstate = 0              'reset state
wDone1:                                   'completed
         return
wcDir: 'choose direction
         random seed                      'random direction
         i = seed & %111                  'mask off for 0-7 only
         lookup i,[tr,fd,fd,fd,rr,fd,fd,tl],wDir     'choose direction
         wstate = 1                       'next state
         return
wcDur: 'choose duration
         random seed                      'random direction and duration
         wDur = (seed & %111111) + 20     'mask for 64 choices
         wstate = 2                       'next state
         return
act:                                      'moves servo motors
         if aDur > 0 then aDec            'already doing one, got here
                  aDur = SACT             'times through this one
                  pulsout LEFT,ldrive * 10
                  pulsout RIGHT,rdrive * 10
aDec:                                          'decrement stuff
         aDur = aDur - 1
aDone:
return
```

digital volume control

for Elektor's Audio DAC 2000



This circuit is especially designed for those who are not satisfied with a single, fixed volume level for their Audio DAC 2000. Minimal modifications to the DAC circuit board and a simple processor circuit are all that are needed to add a *de luxe* pushbutton volume and balance control. An adjustable preset is even included!

Design by T. Giesberts

In part 2 of the description of the Audio DAC 2000 (*Elektor Electronics*, December 1999), we already briefly mentioned that the DF1704 digital interpolation filter includes a digital attenuator that can be adjusted under software control. We more or less suggested that this feature could be used very well in combination with an external processor to realise a digital volume control.

Actually, it wasn't so clever to make this suggestion, since we should have known that such a suggestion would only arouse immediate requests. We thus set to work straight away to develop an extension circuit that allows such a volume control to be added to the Audio DAC.

The result is described in this article. This is a completely custom processor circuit that includes software specially developed for this application. Together, these allow the volume and balance to be precisely adjusted using up/down and left/right pushbuttons. The step size is so small (0.5 dB) that the adjustment is practically continuous over the full range. A fifth button allows the user to quickly select a preset volume level and 'flat' balance, while simultaneously pressing buttons 1 and 5 activates a mute function.

The circuit is easy to build, and connecting it to the Audio DAC circuit board is simply child's play.

A BIT OF HARDWARE

In order to control the digital attenuators in the DF1704, it is necessary to switch this digital filter IC to the software mode. As noted in part 2 of the Audio DAC 2000 project description, this mode is selected by leaving pin 10 of the DF1704 open. Pins 13, 12 and 11 are then the ML, MC and MD inputs, respectively, of the three-wire software control port. All that is needed to make these inputs accessible on the DAC circuit board is to remove DIP switch S2 and replace it with an 8-pin flatcable

connector in DIL format. The relevant part of the Audio DAC circuit diagram is reproduced in **Figure 1**, for clarity.

The complete hardware of the volume control is shown in **Figure 2**. As can be seen, this is truly a basic circuit, consisting of an 80C32 processor (IC1), an address latch (IC2), an EPROM (IC3) containing the necessary software and a RAM (IC4). The flatcable from the DF1704 is attached to connector K1. The five control switches S1 through S5 are located on a separate part of the circuit board, which can be separated from the remainder of the board. A short piece of flat cable between K2 and K3 interconnects the two parts of the circuit board.

The circuit also includes a 5-way DIP switch (S6) for presetting a volume level for switch S3, and there is a place reserved for an optical receiver (IC5). Although there is presently no use for this last component, it at least provides a route for hobbyists to add remote control capability to the volume control. Of course, suitable 'home-made' software is required to support this capability.

All that's left in **Figure 2** is the power supply. As can be seen, this is built in a conventional manner with a rectifier, filter and voltage regulator (IC6). With an eye on possible future applications, the stabilised +5 V supply voltage is also made externally available.

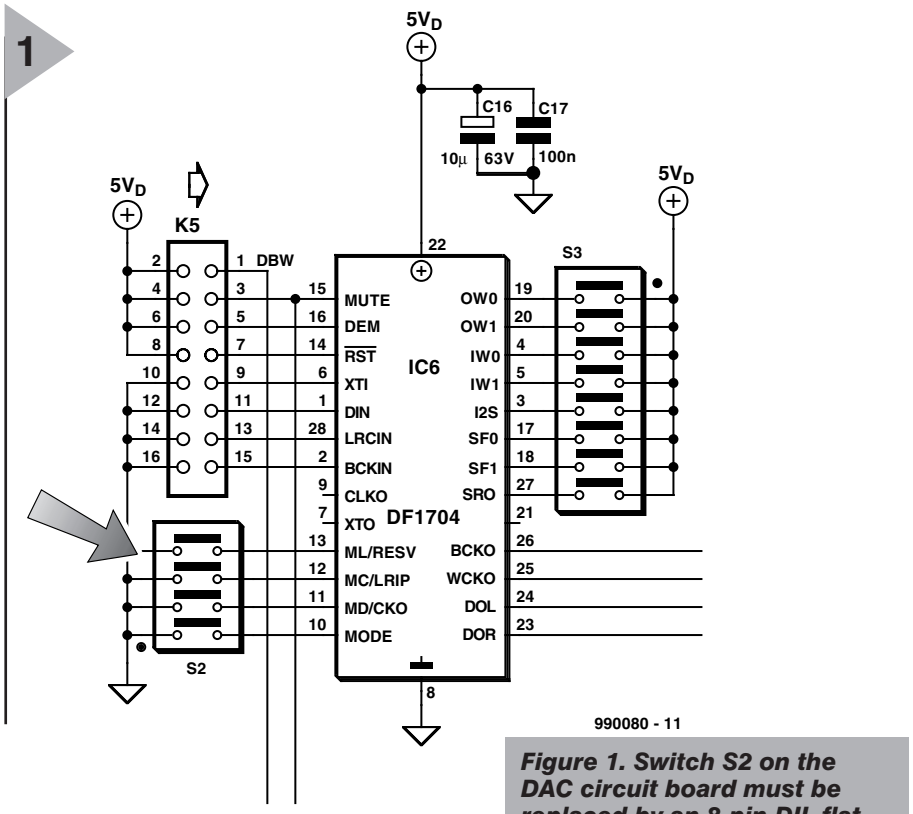


Figure 1. Switch S2 on the DAC circuit board must be replaced by an 8-pin DIL flat cable connector.

FOUR REGISTERS

The DF1704 has four internal registers, called MODE0 through MODE3, that determine the settings of the filter. The attenuators can only be controlled in the software mode, which means that the filters must be supplied with fixed settings by the processor after the circuit is switched on. The DIP switches on the DAC circuit board no longer have any function in the software mode. The detailed information that is needed for programming the various settings can be found on page 10 of the original Burr-Brown data sheet for the DF1704. If you are interested, you can view the data sheet at the Internet site *www.burr-brown.com*.

In addition to the settings that can also be realised using hardware, there are three settings that relate to how the attentuators are controlled. In the MODE3 register, bit 2 is the ATC (attenuator control) bit. This bit controls whether the left and right attentuators work with the same data or separate data. If ATC is set to 0, the two attenuators can be set independently. If ATC is set to 1, both attenuators are coupled and are set by the data for the left attenuator, contained in the MODE0 register. With our volume control, we chose to use separate control data to allow the balance to be controlled, which means that ATC is always set to 0. Register MODE0 thus contains the 8-bit data for the left channel, and register MODE1 contains the 8-bit data for the right channel.

In addition to two address bits, each of the attenuator registers contains an Attenuator Data Load Control bit — LDL — for the left channel and LDR for the right channel. If LDL or LDR is set to 1, the associated attenuator can be set to a new value, while setting LDL or LDR to 0 prevents the setting from being changed. LDL and LDR are always set to 1 by the processor, so that the software determines whether a given attenuator should receive different data.

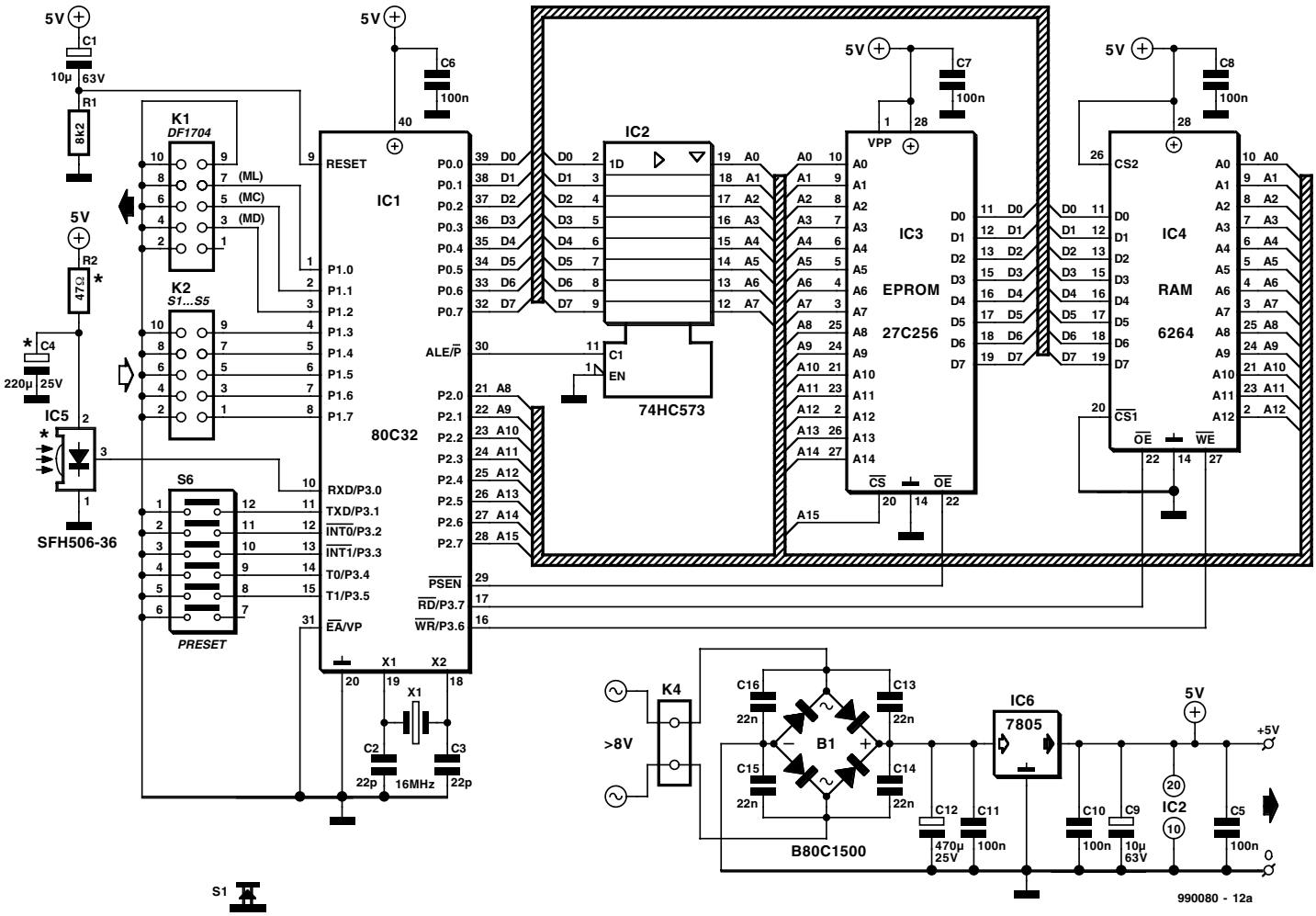
All registers work with 16-bit data, which is sent with the MSB (bit 15) first. The top five bits are always reserved and have no function. Bits 9 and 10 represent the address bits A0 and A1, respectively.

SOFTWARE DESIGN

The software that is needed for the volume control has been developed using a higher programming language known as NiliPascal 3.0. Since the complete program listing takes up a lot of space, we have not printed it here. If you are interested in reading it, have a look at our web site (*www.elektor-electronics.co.uk*). The flow chart shown in **Figure 3** provides enough information to give you an idea of what the processor does after being switched on, and how it responds to pressing any of the five buttons.

At the beginning, the program waits two seconds after the power-up to make sure that there are no longer any resets active. After this, it initialises the digital filter by writing data to all the registers. **Table 1** lists the settings that have been chosen for MODE0 through MODE3. The same data is written to the MODE2 and MODE3 registers each time the Preset button is pressed. Most of the settings correspond to those recommended by Burr-Brown for the hardware mode (Soft Mute = OFF, De-emphasis = OFF, Input Data Format & Word Length = I^{2}S 24 bit, Output Data Word Length = 24 bit, LRCIN Polarity Selection = Left Channel is HIGH and Right Channel is LOW, Digital Filter Roll-Off selection = sharp roll-off, Sampling Frequency Selection for De-emphasis function = 44.1 kHz). In this case, we have deliberately decided not to support the de-emphasis function (which would require an extra connection to the de-emphasis output of the GAL IC), since CD recordings with pre-emphasis are rare.

In the initialisation procedure, the state of the preset DIP switch S6 connected to port 3 is first read. The settings of S6 define the default volume level. This value is also read every time the Preset button (S3) is pressed. After S6 has been read in, the byte value so obtained is adjusted such that the settings of S6 form the five MSBs for both the left and the right attenuators. This amounts to multiplying the read-in value by 4 and then adding 7 to the result. The LSB of S6 (switch S6-1) thus forms the fourth bit of the attenuator



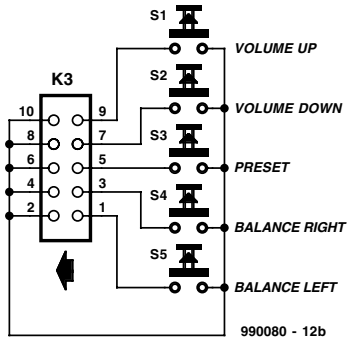


Figure 2. Only a very small amount of hardware is needed. Five pushbutton switches are used to operate the volume control.

nels (with no balance offset), since practice shows that balance controls are almost always set to the middle position. Otherwise there is something wrong with the rest of the installation, such as an incorrect placement of the loudspeakers or something similar. In addition, you know for sure that the two channels are balanced after the preset button has been pressed.

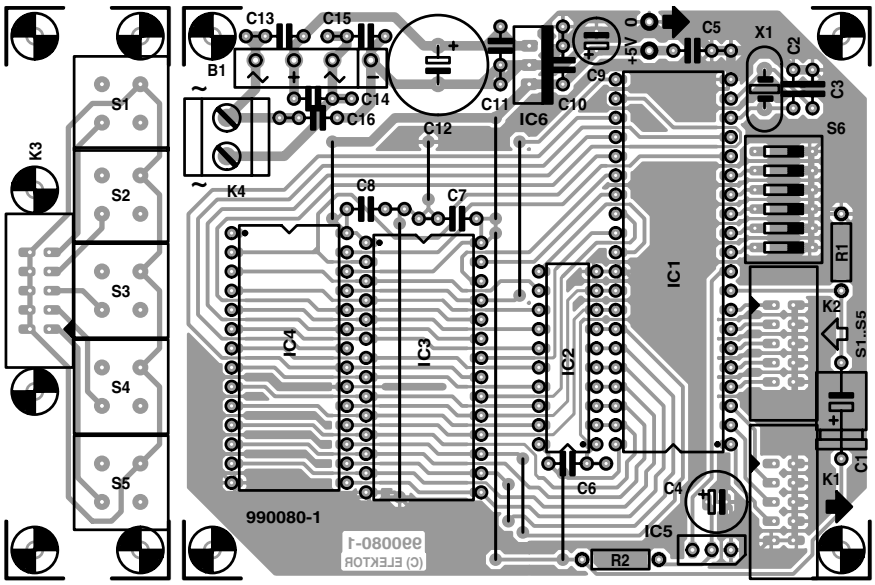
However, the possibility of attenuating one of the channels is provided by two pushbuttons (S4 and S5) that act as a balance control. The software is

data, and has a binary weight of 8.

The step size of the attenuators in the digital filter is 0.5 dB, so the minimum step size of S6 is 4 dB. In practice, this provides adequate resolution to allow the desired average listening level to be set, even though the maximum deviation in the level is never more than ±2 dB when a final amplifier is driven directly. The maximum level, which is obtained when all the switches of S6 are ON, thus corresponds to the level in the hardware mode.

We have expressly chosen to use the same levels for the left and right chan-

4



COMPONENTS LIST

Resistors:
R1 = 8kΩ2
R2 = 47Ω*

Capacitors:
C1 = 10µF 63V axial
C2,C3 = 22pF
C4 = 220µF 25V radial*
C5-C8,C10,C11 = 100nF ceramic
C9 = 10µF 63V radial
C12 = 470µF 25V radial
C13-C16 = 22nF ceramic

Semiconductors:
B1 = B80C1500 (rectangular case)
IC1 = 80C32-16
IC2 = 74HC573
IC3 = 27C256 (programmed, order code 006506-1)
IC4 = 6264
IC5 = SFH506-36*
IC6 = 7805

Miscellaneous:
K1,K2 = 10-way boxheader
K3 = 10-way PCB connector for flatcable
K4 = 2-way PCB terminal block
S1-S5 = pushbutton, 1 make contact, e.g., ITT/Schadow D6-Q-BK + D6Q-BK-CAP
S6 = 5- or 6-way DIP-switch
X1 = 16MHz quartz crystal
PCB, order code 990080-1 (see Readers Services page and Elektor website)
PCB & programmed EPROM: order code 990080-C

*) if required

3

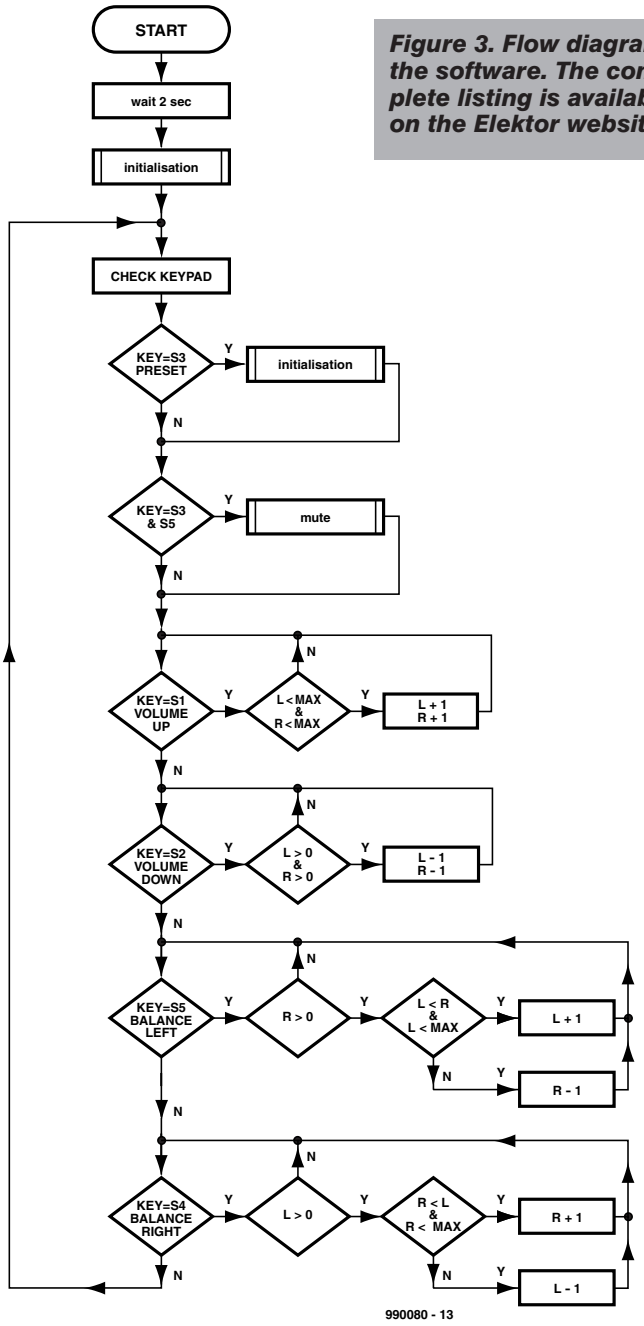
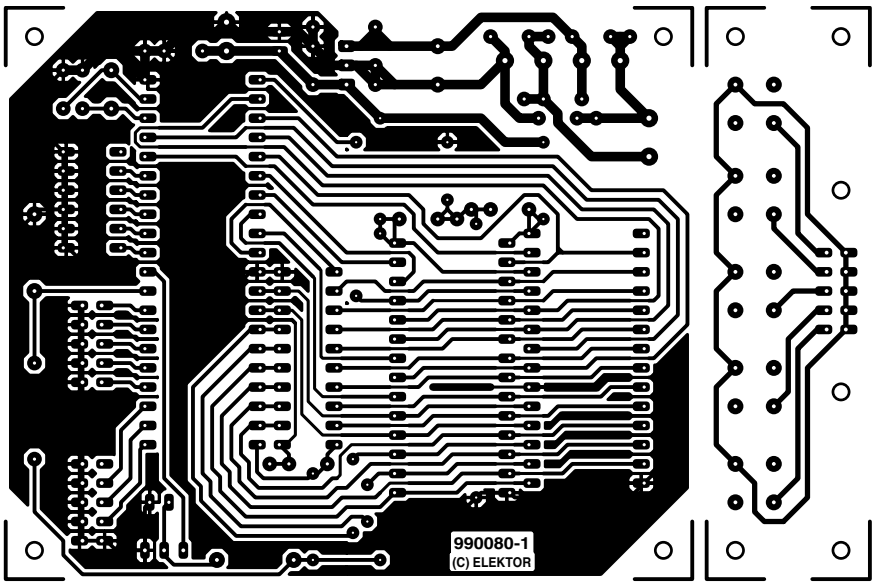
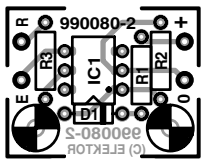
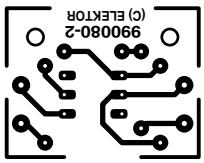


Figure 3. Flow diagram of the software. The complete listing is available on the Elektor website.

Figure 4. The 'keypad' can be separated from the rest of the circuit board.







written such that the balance control works the same way as a 'real' balance potentiometer: if the balance is adjusted to the right the left channel is attenuated, and vice versa. In order to maintain the same level when the balance is adjusted back and forth, whenever the direction of the adjustment is changed the already attenuated channel is first returned to the level of the other channel before the other channel is attenuated.

With the volume control, it is necessary to check that the control value does not exceed the maximum level when the volume is increased or the minimum level when the volume is decreased. If the pushbutton in question is held depressed, a program loop repeatedly increments or decrements the attenuator value by 1. Of course, the maximum and minimum values must be checked for the balance adjustment as well.

Prior to the initialisation, the three LSBs of port 1 are defined as outputs and the rest as inputs. After the initialisation, an endless loop monitors port 1. Each pushbutton switch has its own function, and there is only one valid combination of two switches, which is S1+S5 for the mute function. The program checks in turn whether S3, S1+S5, S1, S2, S5 or S4 is pressed. Pressing any other combination of switches has no effect.

PRINTED CIRCUIT BOARD

The printed circuit board for the digital volume control is shown in **Figure 4**. As already mentioned, this is laid out such that the part with the five pushbutton switches can be sawed loose so that it can (for example) be mounted on the front panel of the enclosure.

What other practical advice can we give regarding the circuit board? Apart from the four ICs, the voltage regulator and the two connectors, there's actually very little on the board. We can thus limit ourselves to suggesting that you use good-quality sockets, pay attention to the polarisation of the electrolytic capacitors, and — don't forget the wire bridges (yes, it's still the most commonly made mistake…)!

As can be seen in the photograph (**Figure 5**), it is easy to build the volume control into the enclosure of the Audio DAC 2000. Just mount the main circuit board on the bottom of the enclosure using stand-offs, and the pushbutton board at the front, possibly with the aid of an angle bracket. Plug in the flatcable between K2 and K3 of the volume control, and then the flatcable between K1 and the 'S2 connector' of the DAC circuit board, and you're all done. Oops, not quite: naturally you have to connect an 8-V supply voltage (AC or DC) to K4. However, if you have used the transformer board described in the January issue of *Elektor Electronics* to provide power to the Audio DAC, all you have to do is connect two wires to connector K4 on the transformer board!

(990080-1)

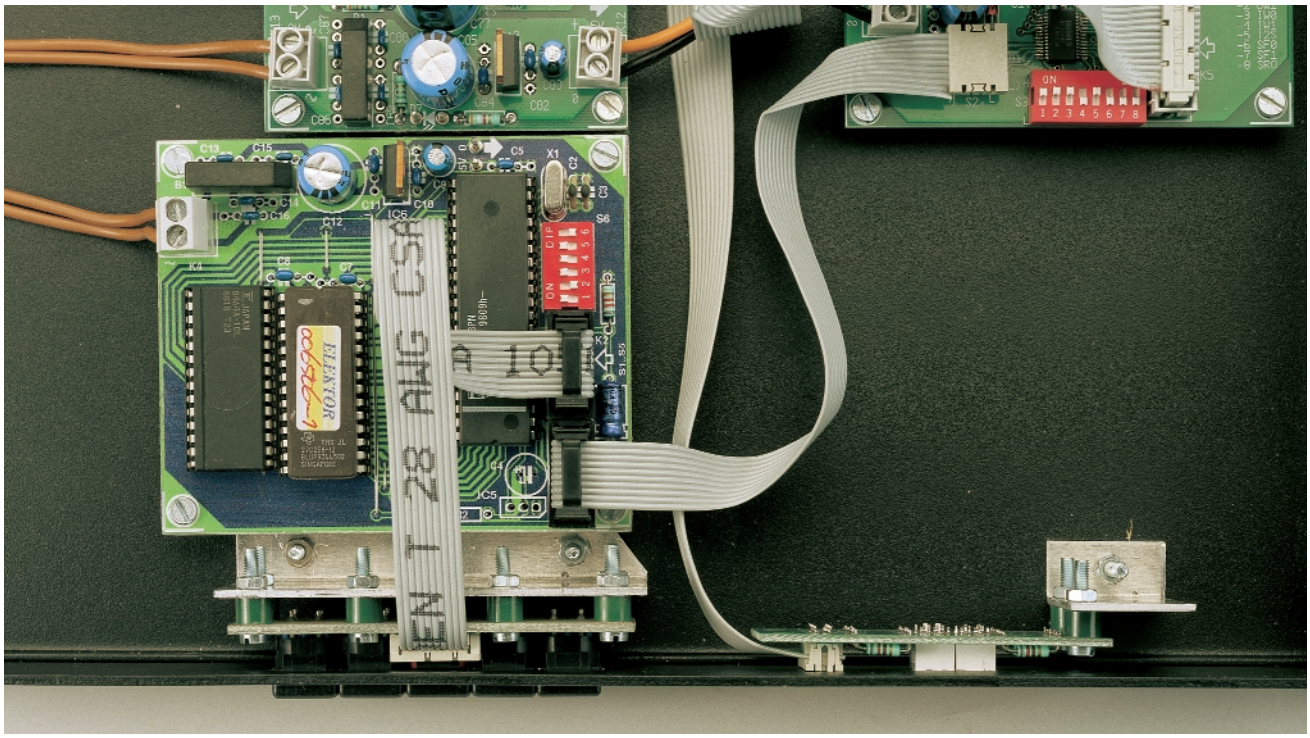
	MODE0	MODE1	MODE2	MODE3
B0	1	1	1	1
B1	1	1	0	0
B2	1	1	0 (res)	0
B3	S6-1	S6-1	1	0
B4	S6-2	S6-2	0	0 (res)
B5	S6-3	S6-3	1	0
B6	S6-4	S6-4	1	0
B7	S6-5	S6-5	0 (res)	0
B8	1	1	0 (res)	0 (res)
B9	0	1	0	1
B10	0	0	1	1

B11...B15 voor alle registers 0 (res)

(res = reserved)

Table 1. Summary of the settings for registers MODE0 through MODE3.

Figure 5. It's easy to fit the volume control into the enclosure of the Audio DAC 2000.



speedometer

for model cars

A reliable speedometer is without question very important for model racing cars. They do cost a pretty penny, after all. This article describes how an ordinary bicycle computer can be used as a speedometer that is convenient, accurate and inexpensive.

From a design by G. Ubaghs

Avid model car fans are naturally enough always interested in the technology and performance of their cars. They would like to know as exactly as possible how fast their model cars actually go, for example so that they can select the final gear ratio for the best performance. Other factors can also be of interest, such as the total distance that the car has travelled, since it is worth knowing how long a car can run on one battery charge or one tank of gas.

There are impressive instruments available in the shops for making these measurements, ranging all the way up to complete telemetry systems. However, they vary in price from expensive to frightfully expensive. This is reason enough for model builders with modest budgets to look for possible alternatives.

The designer of the speedometer described here has worked out such an alternative, and it is as simple as it is inexpensive. He developed an adapter circuit that allows a perfectly ordinary bicycle computer to be used as a speedometer. These devices only cost around £10, and they have the advantage that they can display not only the speed but also elapsed driving time, average speed and the total distance travelled. It's hard to imagine anything better.

A DIFFERENT SENSOR

Most likely everyone knows how a bicycle computer gets its speed pulses. They are generated by a pickup that registers the rotations of the front wheel. This pickup consists of two components. One of these is a magnet that is clamped to a spoke, while the other is a magnetic reed switch that is fixed to the front fork. The reed switch is connected by a thin cable to the computer, which is mounted on the handlebars. Each time the magnet passes the reed switch, it causes the switch contacts to close, and the computer

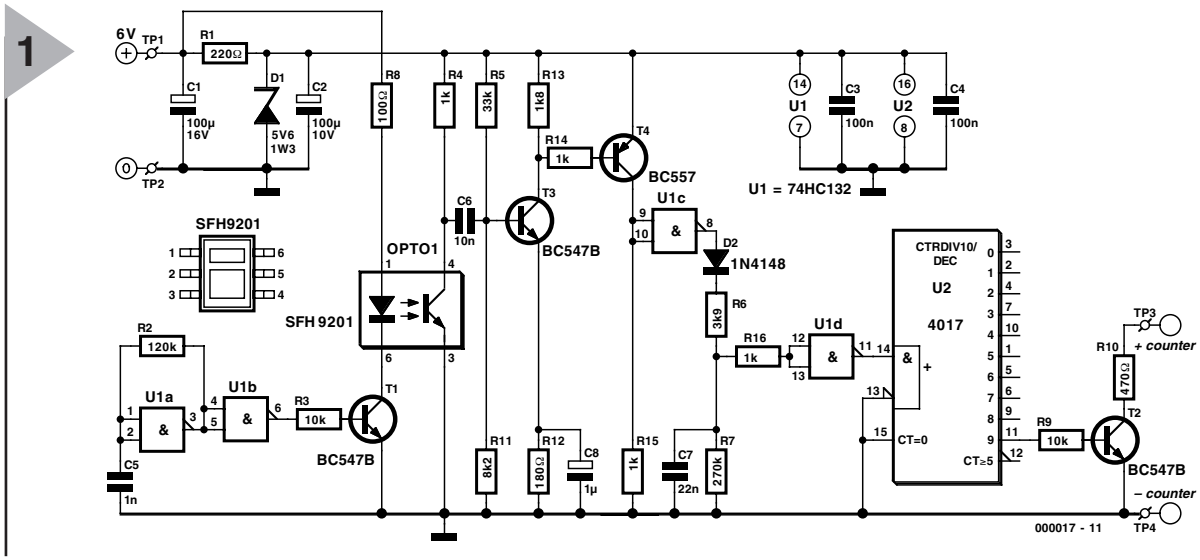


Figure 1. Sensor IC OPTO1 converts variations in the reflected light level into electrical pulses. These are then amplified and detected, and the repetition rate is divided by a factor of ten.

receives a count pulse. This pickup cannot be used with a model car. Even if you could somehow attach the magnet to a wheel, the wheel would then be so out of balance that the car could not be driven. Some other kind of pickup is thus needed.

An optical sensor is an obvious solution. It is a non-contacting and frictionless sensor, just like the magnet and reed switch combination, but with the extra advantage that no additional moving mass is required.

The magnet is replaced in this case by a highly reflective stripe on the side of the tyre, and the reed switch is replaced by an infrared reflective sensor.

The most satisfactory solution for the reflective stripe turns out to be white or silver-coloured paint. From practical experience, the stripe should be around 1 cm wide, but in any case it should not be any wider than one tenth of the width of the non-painted portion of the tyre. The reflective sensor should naturally be mounted on the car in a way that allows it to properly detect the difference between the reflective and non-reflective areas of the tyre.

The adapter circuit

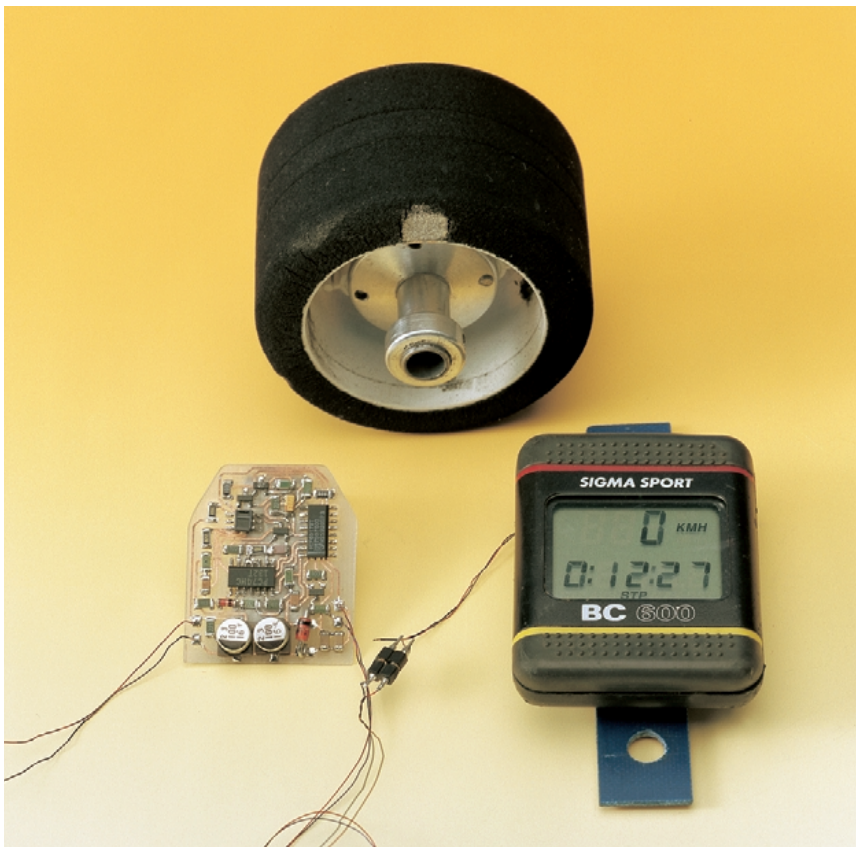
The only other thing that the new sensor needs is a circuit that converts the signal from the reflective sensor into pulses that can be used by the bicycle computer. There are two things that have to be done: first, to convert optical pulses into sufficiently strong electrical pulses, and second to adapt the frequency of the pulses.

The first of these points probably does not need any further explanation. The second has to do with the difference between the circumference of a bicycle wheel and that of a model car wheel. Smaller wheels rotate faster for the same vehicle speed, so they produce pulses at a higher rate. Although the circumference of the bicycle wheel can be set in the computer, there are naturally limits to the range of possible settings. It is not possible to deal with a wheel diameter ratio of ten using the circumference setting alone. This means that the number of pulses must be reduced by a suitable factor.

The practical circuit

As can be seen in **Figure 1**, a relatively simple bit of electronics can adequately realise the requirements just described.

The heart of the circuit is the reflective sensor (OPTO1). A Siemens SFH9201 IC is used for this. It is available from Conrad Electronics, among other sources. In the first version of the circuit, the LED was simply driven by a DC current. This proved to be unsatisfactory, since the sensor also reacted to ambient light. This produced so many erroneous pulses that the accuracy of the speedometer suffered greatly. We thus switched over to driving the LED with a 10-kHz AC current early on in the design. This has the



COMPONENTS LIST

Resistors:
R1 = 220kΩ
R2 = 120kΩ
R3.R9 = 10kΩ
R4,R14,R15,R16 = 1kΩ
R5 = 33kΩ
R6 = 3kΩ9
R7 = 270kΩ
R8 = 100Ω
R10 = 470Ω
R11 = 8kΩ2
R12 = 180Ω
R13 = 1kΩ8

Capacitors:
C1 = 100µF 16V
C2 = 100 µF 10V
C3,C4 = 100nF
C5 = 1nF
C6 = 10nF
C7 = 22nF
C8 = 1µF 10V

Semiconductors:
D1 = zener diode 5V6 1W3
D2 = 1N4148
T1,T2,T3 = BC547B
T4 = BC557B
IC1 = 74HC132SO
IC2 = 4017SO
OPTO1 = SFH9201 (Siemens)

advantage that an AC amplifier can be used for the detector circuit, which largely eliminates the effects of ambient light variations.

The 10 kHz signal for the LED is produced by the oscillator built around IC1a. Gate IC1b acts as a buffer that drives the sensor LED via transistor T1.

Whenever the white stripe on the tyre passes in front of the sensor, the phototransistor in the sensor will briefly conduct at a 10 kHz rate. A pulse train with a frequency of 10 kHz is thus produced across resistor R4. This signal is coupled out by capacitor C6 and then amplified by an AC amplifier formed by transistors T3 and T4. This results in a 10 kHz pulse waveform across resistor R15. This is buffered by gate IC1c and then applied to a detector circuit consisting of the diode D2, resistors R6 and R7 and capacitor C7. The job of the detector circuit is to convert the short series of pulses into a logical '1'. The component values are rather critical, since capacitor C7 should be charged before the stripe has passed completely by the sensor, but it should also be fully discharged via resistor R7 before the stripe again appears in front of the sensor and a new pulse train arrives.

The output signal of the detector is buffered by gate IC1d and finally ends up at the last part of the circuit, the divide-by-ten counter IC2. This allows only every tenth pulse from the detector to be passed on to transistor T2. The open collector of this transistor is connected to the input of the bicycle computer.

Power supply

The circuit runs with a supply voltage of 5 V. This can usually be derived from the receiver module in the car.

In the author's prototype, a 6 V supply voltage was available for the receiver. Capacitor C1 provides extra filtering for this voltage, which is then used directly to supply the LED in the optocoupler (U+). The supply voltage for the rest of the circuit is stabilised at around 5 V by resistor R1 and the Zener diode D1. Capacitor C2 acts as a reservoir capacitor, while C3 and C4 provide local decoupling for IC1 and IC2.

Construction

The circuit is not particularly critical, and given the small number of components, it is also not difficult to build. The best way to build it depends in part on the shape of the model car in question. The most important factor is naturally that the sensor OPTO1 must have an unobstructed view of the reflective stripe on the tyre.

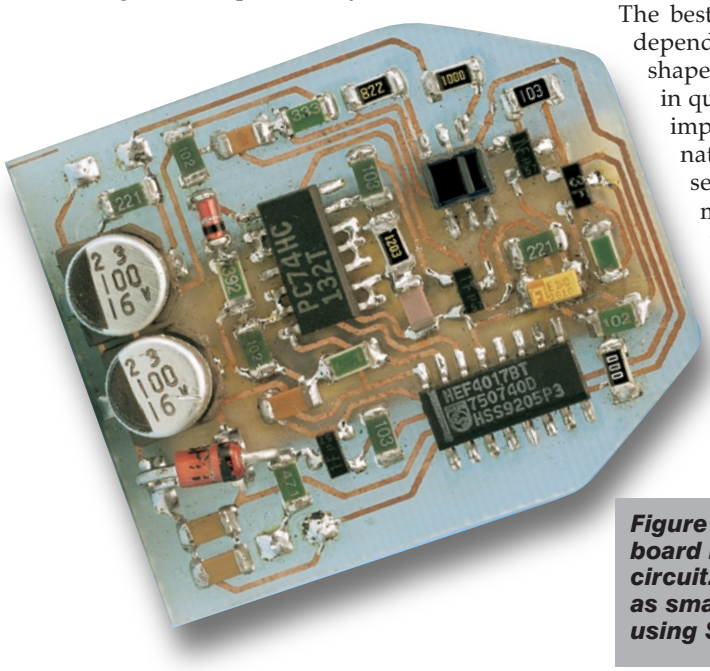
Since space is always a consideration in model building, the author has designed a printed circuit board for the speedometer that largely uses SMDs. **Figure 2** shows the track and component layouts of this board. Although this board worked well in the prototype, we must emphasise that it has not been tested in the Elektor Electronics lab. It should thus be seen as a suggestion, in the sense of 'this is a possible solution.'

In addition to the exact construction of the adapter circuit, the manner in which the bicycle computer itself is mounted will naturally be largely determined by the specific features of the model car in question. We leave this question to the inventiveness all of those who build the speedometer circuit.

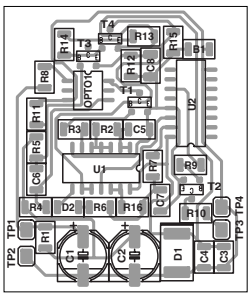
Connecting the circuit is dead easy. Wire the 6 V supply to the electrolytic capacitor C1 (with the right polarity!), and connect the two leads of the computer cable to resistor R10 and earth, respectively. On the prototype board shown in Figure 2, the supply connections can be made out with a bit of effort next to the labels TP1 and TP2, while the output connections are labelled TP3 and TP4.

Finally, there is one last remark regarding setting the value of the wheel circumference in the bicycle computer: don't forget the factor of 10 provided by the divider in the adapter circuit! For example, if the tire of the model car has a circumference of 21 cm, a circumference of 210 cm must be set in the bicycle computer.

(000017-1)



2



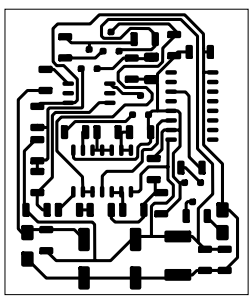
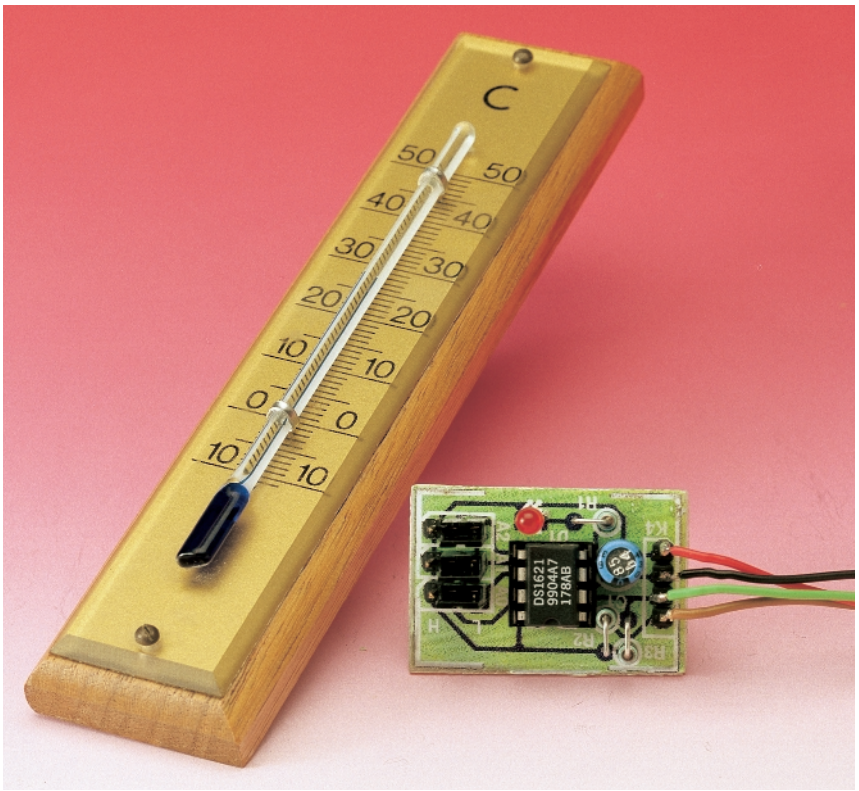


Figure 2. A possible circuit board layout for the adapter circuit. The board can be kept as small as a matchbox by using SMD components.

temperature measurements with a DS1621

a smarter sensor with an I²C-bus interface

In many areas of daily and technical life, making temperature measurements is the most important instrumentation task. 'Intelligent' digital sensors, which can do more than just output temperature readings, are winning more ground as complements to classical analogue techniques. An example of such a sensor is the Dallas Semiconductor DS1621.



Technical specifications

No external components necessary
Temperature measurement range –55 °C to +125 °C
Resolution 0.5 °C
9-bit temperature measurement value with sign and magnitude
Conversion time 1 second
User-programmable thermostat function
Serial two-wire interface (I²C) with open-drain connections

Naturally, the main job of a modern digital temperature sensor is to determine the current temperature. It does this by measuring its own temperature, which is nearly the same as the temperature of its surroundings, and converting this analogue temperature value into a digital word that it sends to a microcontroller via a serial interface. The user doesn't have to worry about any of the analogue technology needed for calibration, amplification of the thermal voltage with opamps, linearisation of characteristic curves and so on, since all this is either done by the chip itself or can be easily set using software. External components are thus largely unnecessary.

In addition, an intelligent sensor provides supplementary functions, such as temperature logging with the DS1615 (Elektor Electronics September

Based on an idea from
Prof. Dr. B. vom Berg

1999) or thermostatic operation, as with the Dallas Semiconductor DS1621 described in this article. The main characteristics of this IC are summarised in the Technical Specifications box. A condensed datasheet of the DS1621 may be found in this month's *Datasheets* section.

TEMPERATURE MEASUREMENT

The primary function of the DS1621 is of course measuring temperature. **Figure 1** shows the modules that are present in the IC for this purpose. The DS1621 measures the temperature by counting the number of clock cycles of produced by an oscillator with a low thermal coefficient within a given time window. The width of this window is determined by a second oscillator that has a large temperature coefficient. The counter is preloaded with a value that corresponds to a temperature of –55 °C. If the counter value reaches zero within the time window, the temperature register (which is also set to the value representing –55 °C) is incremented to indicate that the temperature is greater than –55 °C. The counter is clocked until it reaches zero and then restarted, as long as the time window remains open.

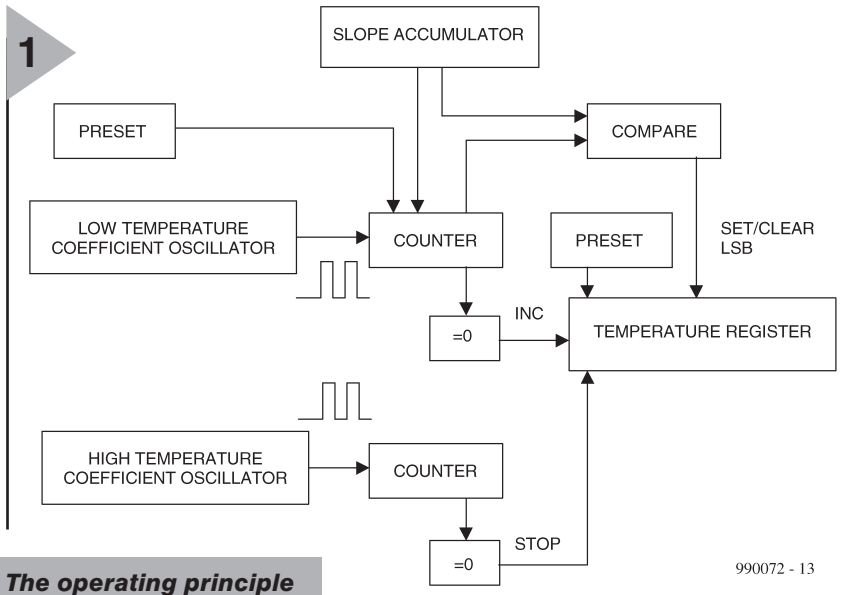


Figure 1. The operating principle for temperature measurement.

The counter does not always accumulate the same number of clock pulses within the time window for every degree of difference in the temperature. This is because a (variable) offset is added to the counter by the Slope Accumulator for each degree of temperature difference, in order to compensate for the non-linear behaviour of the oscillator over the temperature range. This allows a relatively high temperature resolution of 0.5 °C to be achieved.

The measured temperature is stored in the temperature register as a 9-bit value. As shown in **Table 1**, the measurement range of the DS1621 is from –55 °C to +125 °C in steps of 0.5 °C. The current temperature is output via the two-wire serial interface in response to a READ TEMPERATURE command, either as single byte with a resolution of 1 °C or as two bytes with a resolution of 0.5 °C. In the latter case the MSB is sent first, followed by the LSB, which only specifies the least significant bit of the temperature value (0.5 °C). The remaining 7 bits of the LSB are set to zero.

A trick can be used to achieve an even higher resolution. The temperature is first read, and the 0.5 °C bit (the LSB) is chopped off. The resulting value is called TEMP_READ. The value remaining in the counter when the time window closes can be read using the READ_COUNTER command and then stored in the variable COUNT_REMAIN. The value in the Slope Accumulator can then be read using the READ SLOPE command. This value (COUNT_PER_C) is the number of counting edges per degree Celsius at the current temperature. The temperature can then be determined using the following formula:

$$\text{Temperature} = \text{TEMP\_READ} - 0.25 + \frac{(\text{COUNT\_PER\_C} - \text{COUNT\_REMAIN})}{\text{COUNT\_PER\_C}}$$

Table 1. Temperature vs. data value

Temperature in °C	Output data binary	hex
+125	01111101 00000000	7B00
+25	00011001 00000000	1900
+0.5	00000000 10000000	0080
0	00000000 00000000	0000
- 0.5	11111111 10000000	FF80
- 25	11100111 00000000	E700
- 55	11001001 00000000	C900

THERMOSTAT

As indicated in the functional block diagram of the DS1621 shown in **Figure 2**, the intelligent sensor also includes a thermostat function. Two

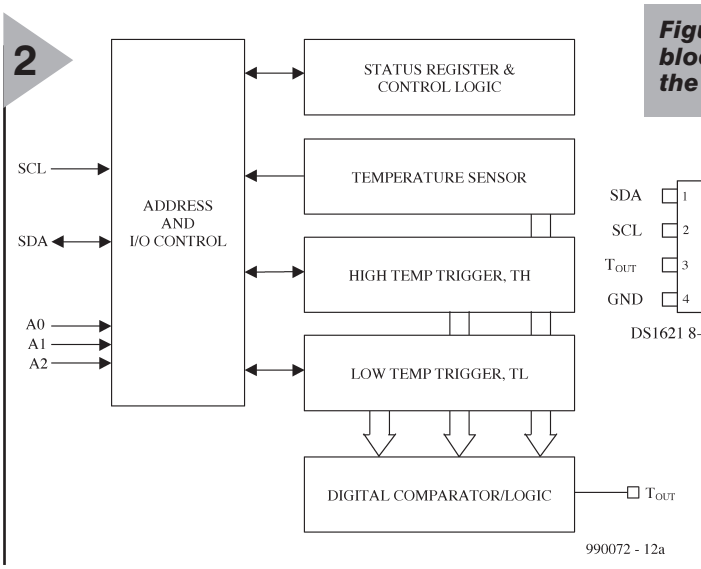


Figure 2. Functional block diagram of the DS1621.

threshold values can be programmed to create a hysteresis function. If the temperature reaches one of the preset values, rises above the upper threshold value TH or drops below the lower threshold value TL, the T_{OUT} output becomes active (High).

In addition, the DS1621 notices whenever a temperature threshold has been exceeded and sets corresponding flags in the Configuration/Status Register. This register also includes other information that controls the operating mode in a very particular application. The register is configured as follows:

DONE	THF	TLF	NVB	1	0	POL	1SHOT

The meanings of these items are as follows:

DONE
High = conversion completed
Low = conversion active

THF (Temperature High Flag)
Set to 1 when the temperature is equal to or greater than the threshold TN. Remains active until reset.

TLF (Temperature Low Flag)
Set to 1 when the temperature is equal to or less than the threshold TL. Remains active until reset.

NVB (Non-volatile memory flag)
Active (1) whenever an EEPROM cell is being written. This can occur every 10 ms.

POL (Output Polarity bit)
Determines whether the output is active High (1) or active Low (0). Non-volatile.

1SHOT
If the 1SHOT Mode bit is High, the DS1621 performs a temperature conversion after receiving a CONVERT T command. Otherwise it operates in the normal mode, in which it continuously performs measurements and conversions. This bit is non-volatile.

COMMUNICATIONS: FORM AND CONTENT

The DS161 works as a slave in a two-wire bus system with open-drain SDA and SCL lines and a clock rate of 100 or 400 kHz. Communication with a microcontroller or PC takes place using the well-known I^2C protocol, which need not be further explained. You can find more information regarding timing, as well as all other technical specifications, in the DS1621 data sheet located at *www.dalsemi.com/doccontrol/pdfs/pdfindex.html*.

Every communication needs a start condition and concludes with a stop bit. The DS1621 responds to each received byte with an Acknowledge.

The start bit is always followed by an address byte, which consists of four bits set to the value 1001 and three bits that are determined by the levels applied to the address inputs A2, A1 and A0. The final bit determines the communications direction (R/$\overline{W}$).

The address byte is followed by a command byte. The DS1621 recognises the read and write commands listed in **Table 2**, including their codes. The two conversion commands are finished when the address and command bytes have been sent, but one or two data bytes follow the command byte for the write commands. With the read commands, the structure is somewhat more complicated, since the IC must be provided with a second address byte to set the R/$\overline{W}$ bit to Read. The one or two data bytes can be read only after this has taken place.

Dallas Semiconductor has a free Demonstration Kit available for the DS1621. This monitors the sensor IC in a temperature logging application. The three temperature values (measurement and hysteresis) are displayed with a resolution of 0.1 °C or 0.5 °C. The user can set the threshold values for the thermostat output and the acquisition interval in seconds, enable direct output of the measured values to a file and select a temperature display in Fahrenheit or Celsius, or as a 2-D or 3-D graphic. The software supports three parallel ports and eight hardware addresses for the DS1621.

The software for this kit, which runs under Windows 3.xx and higher, may be downloaded from this Internet site:
ftp://ftp.dalsemi.com/pub/thermal/ds1621k.zip
If you do not have access to the Internet, you can obtain the software on a floppy disk from our Readers Services under order number **996027-1**.

HARDWARE

As the name suggests, a kit includes a printed circuit board with the sensor and all other components that are needed to connect to the parallel port of a PC. However, any electronics hobbyist would rather construct this minimal amount of hardware himself and thus avoid the stress of ordering the kit. All that you need is the printed circuit board shown in **Figure 3**, the listed components and a hot soldering iron.

COMPONENTS LIST

Resistors:
R1 = 1kΩ5
R2,R3 = 10kΩ

Capacitor:
C1 = 100µF 16 V radial

Semiconductors:
D1 = LED, low current
IC1 = DS1621 (Dallas Semiconductor)

Miscellaneous:
K1,K2,K3 = 3-way pinheader
K4 = 4-way pinheader
Disk, contains project software, order code **996027-1**

3

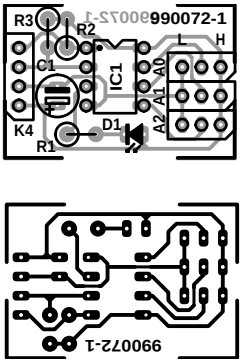


Figure 3. The printed circuit board fits into a matchbox.

4

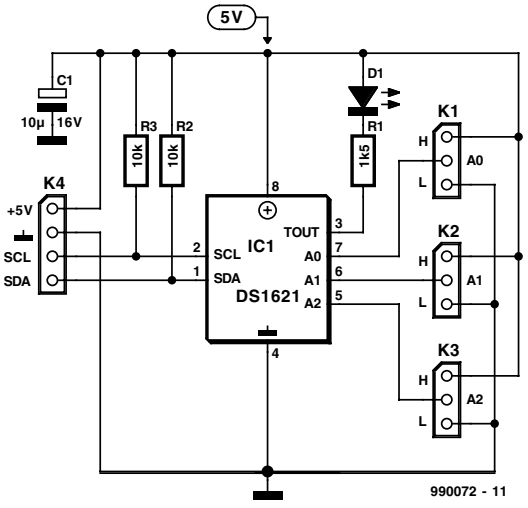


Figure 4. No surprises here: the LED indicates that the threshold has been exceeded.

Table 2. DS1621 command set

Command	Description	Value (hex)	Following bytes
Temperature measurement commands			
Read Temperature	Reads the latest temperature value from the temperature register	AA	reads two data bytes
Read Counter	Reads the remainder in the time window counter	A8	reads one data byte
Read Slope	Reads the value in the Slope Accumulator	A9	reads one data byte
Start Convert T	Starts a temperature conversion	EE	no data
Stop Convert T	Stops a temperature conversion	22	no data
Thermostat function commands			
Access TH	Reads/writes the upper threshold value from/to the TH register	A1	reads/writes two data bytes
Access TL	Reads/writes the lower threshold value from/to the TL register	A2	reads/writes two data bytes
Access Config	Reads/writes data from/to the Configuration register	A1	reads/writes one data byte

The circuit on this board, which you can assemble in a few minutes, corresponds to the schematic diagram shown in **Figure 4**, which needs only a brief explanation.

The address of the DS1621 is set by jumpers K1 through K3. The board can be very easily connected to the I²C interface of a microcontroller system via K4.

However, connecting the board to a parallel port of a PC, in order to run the Dallas Semiconductor program, is another story. For this, you will need a parallel I²C interface, such as that published in the February 1996 issue of *Elektor Electronics*, with the accompanying software (Readers Services order number **950063-C**).

(990073-1)

CORRECTIONS & updates

I²C interface for the printer port

March 1999, Supplement p. 10-11 (990034-1)

If the software keeps producing error messages despite proper operation of the hardware, pins 10 (ACK) and 13 (SLCT) of K1 should be connected to ground.

EEDTS Pro

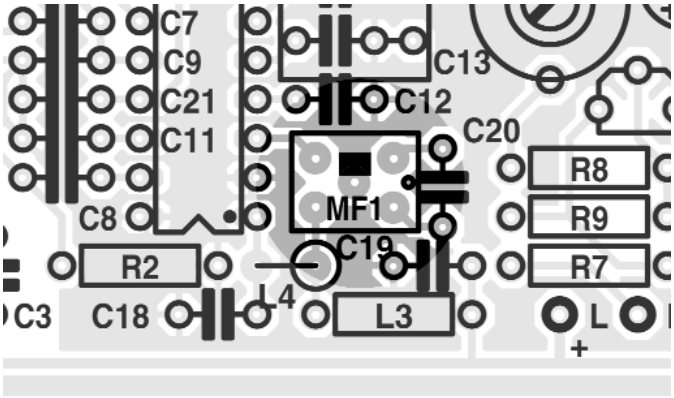
Series, June 1999 – October 1999 (980055)

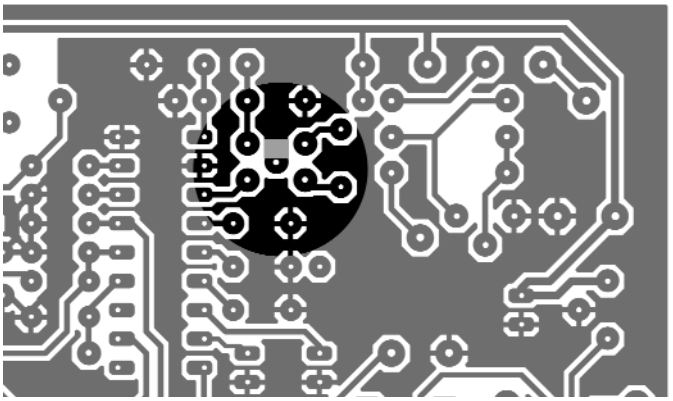
Constructors and users are advised that the author, S. de Vries, has set up a dedicated website at:
www.gironet.nl/home/editspro
Mr. De Vries may also be reached by email on:
editspro@gironet.nl

Poor Man's Shortwave radio

October 1999, p. 42-45 (990068-1)

On the PCB, the centre pin of filter MF11 has no ground connection. This causes insufficient selectivity. The error may be corrected by fitting a small piece of wire as indicated in the drawing.
If you use a capacitor with a pin spacing of 7.5mm in position C13, the ground connection has to be made with a small piece of wire.





I²C interface for the printer port

March 1999, Supplement p. 10-11 (990034-1)

If the software keeps producing error messages despite proper operation of the hardware, pins 10 (ACK) and 13 (SLCT) of K1 should be connected to ground.

EEDTS Pro

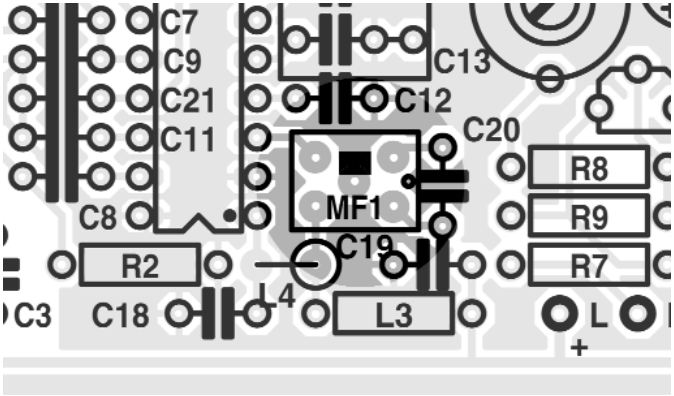
Series, June 1999 – October 1999 (980055)

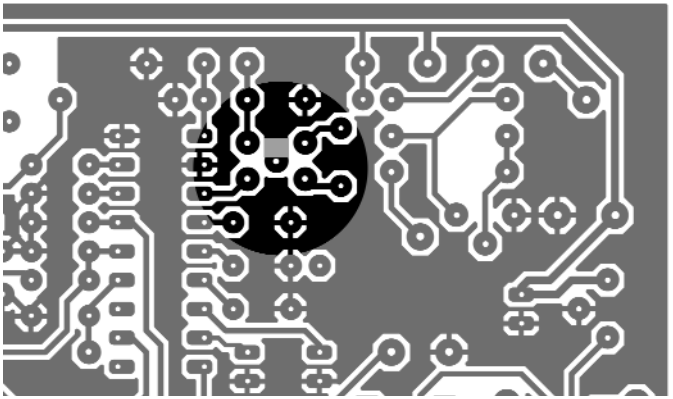
Constructors and users are advised that the author, S. de Vries, has set up a dedicated website at:
www.gironet.nl/home/editspro
Mr. De Vries may also be reached by email on:
editspro@gironet.nl

Poor Man's Shortwave radio

October 1999, p. 42-45 (990068-1)

On the PCB, the centre pin of filter MF11 has no ground connection. This causes insufficient selectivity. The error may be corrected by fitting a small piece of wire as indicated in the drawing.
If you use a capacitor with a pin spacing of 7.5mm in position C13, the ground connection has to be made with a small piece of wire.





solar cells from Royal Dutch Shell

new plant in Germany is most up-to-date in the world

Solar energy for the Gelsenkirchen plant: solar modules fitted to some 420 square metres of roof and frontage areas produce some 100,000 kWh of energy per year. The modules use semi-transparent Optisol™ photovoltaic cells from Pilkington Solar International.

With advanced automation and a planned output per production line of over 1000 cells per hour, the recently opened Royal Dutch Shell Group's new world-scale solar cell factory in Gelsenkirchen, Germany, close to the Dutch/German border sets new standards for the production of multi-crystalline silicon solar cells. The continuing acceleration of production and sales worldwide of solar cells is reducing the cost of solar-cell produced electricity by 6% annually, which must be good news for consumers.

By Ernst Krempelsauer

INTRODUCTION

The Royal Dutch Shell Group's interest in the solar electricity (also called photovoltaic) industry began in the early 1970s. The business grew with the establishment of Showa Solar KK in Japan and Shell Solar Energy BV. Until the opening of the Gelsenkirchen factory, Shell Solar Energy BV in the south-eastern Netherlands was the centre of the company's cell and module production.

The Shell Group's commitment to solar energy was reinforced in 1998 when the Group founded its fifth core business, Shell International Renewables, of which Solar is one of the three business units.

The decision of the Shell Group to build their new solar cell factory in Germany is based partly on the fact that Germany is currently the largest European market for solar cells and, perhaps more importantly, that the regional (Nord Rhein Westphalia) and

national governments undertook to pay about 25 per cent of the cost of the factory. There is further support from the regional government for the associated Photovoltaic Information Centre. Yet another factor is that there is already a solar-cell manufacturing outfit in Gelsenkirchen: that of Pilkington Solar International. Finally, scientific support in Gelsen-kirchen comes from a branch of the Fraunhofer Institute for Solar Energy Systems. There is also the likelihood that Bayer Solar will start producing multi-crystalline wafers, the basis of solar cells, in Gelsenkirchen.

THE NEW FACTORY

Apart from its outstanding architectural design, the factory is noteworthy for its drastic shortening of the production line to only 70 metres (200 ft) and the complete automation of this with numerous newly developed production units. The plant consists of an hexagonal building with sloping roofs, in which production takes place at two levels, as well as a multi-storey, oval administration block with an annexed visitor centre. Most of the complex, which is 83 m long, 47 m wide, and 12 m high, is fitted with solar cells that produce 100,000 kW of energy per year. This means that the solar cells will be produced with the aid of solar energy.

Initially, the plant will use a single production line capable of manufacturing up to 5 million solar cells per year. This output can produce up to 10 MW of energy. Plans have already been approved to extend the plant to provide two more production lines to increase the overall annual output to 13 million solar cells capable of providing 25 MW of energy. These plans also include contingencies to cater for newly developed techniques.

THE MARKETS

The Shell Group believes that solar energy will play an increasingly significant role in meeting the growing energy needs of the world and that there are a wealth of applications for this environmentally sound and affordable energy solution.

A large part of the output of the new Gelsenkirchen plant will, of course, be exported. The Shell Group is particularly interested in rural electrification that brings decentralized renewable power to remote or non-grid supplied communities. Examples are 50,000 solar home systems for South Africa (in a joint venture with Eskom, South Africa's largest power company), and 100,000 for China. There is, of course, also a vast market in the industrialized world, USA, Europe, and Japan, for grid-connected solar systems. In these countries, there is also a market for stand-alone solar power supplies in remote locations, such as telecommunication sites.

AND COSTS?

Solar cells are expensive; at their current price they cannot possibly compete with traditionally generated electricty (fossil and nuclear). For instance, in Germany, the price of solar energy is about 53p per kWh (in Britain, the daytime tariff of traditionally produced electricity is 6.76p and the night-time tariff 3.28p per kWh).The production costs of photovoltaic modules are, however, falling and this trend seems likely to continue. The Shell prognosis is for a 6% cost reduction per year. This means that solar energy technology will become incresingly competitive when compared to fossil fuel or nuclear power stations.

The cost of photovoltaic panels can be reduced further, first by advanced automation in manufacturing and improved light conversion efficiency in current crystalline silicon technology. It could be followed by large-scale deployment of one of several types of thin film technologies. If these assumptions are correct, the price of solar energy and traditionally generated electricity will equate by around 2020.

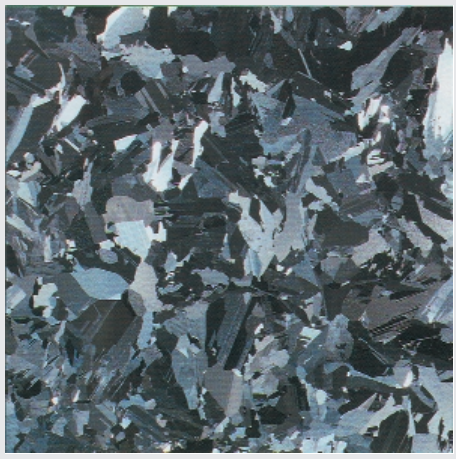
Greenpeace have also provided an interesting response to Shell's solar cell engagement. Shell and Greenpeace recently clashed vehemently on the Brent Spar incident. Although Greenpeace in principle welcomes the new solar cell plant, it criticises the invest-

From silicon wafers to solar cells

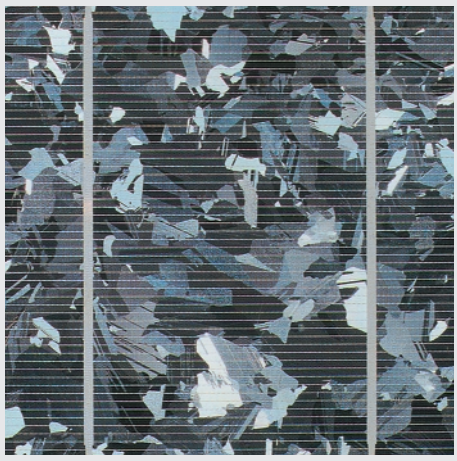
In the Gelsenkirchen plant, solar cells are manufactured from 12.5 cm square, 350 µm thick multi-crystalline silicon wafers. Silicon is one of the most abundant materials on the planet (in the form of quartz sand) and is an environmentally friendly material.

The wafer must first be smoothed and cleaned, that is, etched, thoroughly before it can be used to make a solar cell.

Next, one side of the wafer is made the equivalent of the positive terminal on a battery. This is done by applying a liquid containing phosphorous onto one surface in a process called doping.



Multi-crystalline silicon wafer



Metallized solar cell

The wafer is then heated in an oven to allow the phosphorous to eat its way into the surface of the silicon in a process called diffusion.

The next stage is the printing of electrical contacts on the front surface of the wafer. The contacts are made of a metal that collects and conducts electrons that are generated when sunlight strikes the cell. Obviously, the whole surface cannot be covered with metal, and the contacts therefore consist of a grid of 90 µm wide lines printed onto the wafer.

The wafer is then coated with an anti-reflective film of titanium dioxide (TiO_2) to ensure that as much light as possible is absorbed into the cell.

The cell is then turned over and the negative side of the solar battery is formed by coating the surface with a thin aluminium film in a process called metallization. The wafer is then heated again in an oven to ensure that the aluminium is fully absorbed into the silicon.

Finally, the positive and negative sides of the solar cell are isolated from each other.

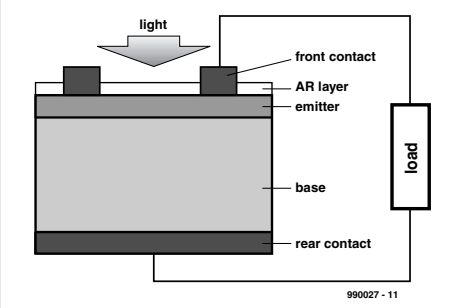
The technology of solar cells

Solar cells depend on the photovoltaic effect, a process in which two dissimilar materials in close contact act as an electric cell when struck by light or other radiant energy.

Light striking such crystals as silicon or germanium, in which electrons are usually not free to move from atom to atom within the crystal, provides the energy needed to free some electrons from their bound condition.

Basically, a solar cell consists of a large semiconductor diode with its p-n junction just under and parallel to the upper surface – see Illustration. The n (emitter) and p (base) regions are provided with ohmic contacts to enable an electric current to flow through an external circuit. The front contact is a grid that allows the loss-free flow of electric current but does not create a shadow on the active upper surface of the cell. An anti-reflection coating ensures that as much light as possible is absorbed into the cell. The back contact is flat.

The semiconductor material absorbs photons, the primary energy packets of light. The photons raise the energy level of the electrons in the semiconductor, exciting some to jump from the lower-energy valence band to the high-energy conduction band. The electrons in the conduction band and the holes they have left behind in the valence band are both mobile and can be induced to move. The electron motion and the movement of holes in the opposite direction constitute and electric current.



Construction of a solar cell

Solar cells based on silicon wafers

Currently, more than 80 per cent of solar cells produced worldwide use crystalline (c-Si) or multi-crystalline (mc-Si) silicon wafers. Power solar modules use these wafers exclusively. Solar cell technology is developing rapidly in the wake of microelectronics technology. Nevertheless, solar cells remain expensive and their efficiency is not high. However, there is a growing market need for them, which makes research into improving their performance/cost ratio essential and worthwhile. Laboratory models achieve efficiencies of up to 25 per cent (small-area modules) and up to 20 per cent(large-area modules). Commercially available cells have efficiencies of about 16% (c-Si) or 15% (mc-Si). These are confidently expected to improve to 18% and 20% respectively in the near future.

A relatively recent development is the crystalline silicon film solar cell. In this, a film 20–50 µm thick is deposited onto ceramic, graphite, or other materials. The advantage of this is a substantial saving of expensive silicon. Small-area laboratory models have achieved an efficiency of 11 per cent.

Solar cells based on gallium arsenide wafers

Gallium-arsenide (GaAs) solar cells have a higher efficiency than silicon-based types with a peak value of almost 26 per cent. However, their production costs are considerable higher than those of silicon types. Also, the cost of gallium-arsenide is much higher than that of silicon. Currently, GaAs solar cells are used almost exclusively in space technology. In future, they may well be used in concentrator systems.

A concentrator acts like a magnifying glass in focusing sunlight before it is absorbed by the solar cell. The power output of normal cells has been increased by up to 21 times by the ue of a concentrator. The highest efficiency has been recorded at around 34 per cent.

Thin film solar cells

Solar cells based on amorphous silicon

Solar cells based on amorphous silicon (a-Si) have no p-n junction but a p-i-n structure. In this structure, a region of almost intrinsic (i-type) semiconductor material is sandwiched between the p-type and n-type regions. The 1 µm thick depletion layer associated with a p-n junction is contained entirely within the i-region. The photo current is produced entirely in the depletion layer.

The cell is produced by depositing amorphous silicon onto glass at low temperature, which makes relatively low-cost, energy-efficient mass production possible. Unfortunately, this type of solar cell has an efficiency of only 7–8 per cent and it is, therefore, used only in low-power applications, such as pocket calculators and wrist watches.

Solar cells based on cadmium-tellurium

Cadmium-tellurium (CdTe) is a compound semiconductor that can be made only as a p-conductor. A p-n junction may be made by depositing a film of CdTe onto an n-conductor such as cadmium sulphide.(CdS). Laboratory models have achieved efficiencies of up to almost 16 per cent, but commercial CdTe modules have an efficiency of only 7–8%. Moreover, the use of cadmium in solar cells is a controversial subject.

Solar cells based on copper indium di-selenium

Copper indium di-selenium ($CuInSe_2$) is a compound semiconductor that can be made only as a p-conductor. A p-n junction may be made by vapour-depositing a film of CdTe onto an n-conductor such as cadmium sulphide (CdS) or zinc oxide (ZnO). Maximum efficiencies achieved under laboratory conditions are up to 18% (small area) and 14% (large area). Solar modules have an efficiency of up to 11%. The use of this material is also a controversial subject.

Solar cells based on dye-sensitized material

Solar cells based on dye-sensitized material belong to the category of photo-electro-chemical solar cells. In these cells, incident light is absorbed by a very thin (0.5–50 nm) dye film that is deposited onto a large, porous layer of rutile (TiO_2). When light is being absorbed, electrons of the dye material are excited, causing them to jump to the rutile. The current loop is closed by an electrolyte, which also provides regeneration of the dye molecules. This type of solar cell is simple and relatively inexpensive to produce. Prototype cells have an efficiency of up to 10%, but commercial models only 8%. Unfortunately, the stability of the cell cannot be guaranteed.

Source: Fraunhofer Institut für Solare Energiesysteme (ISE Freiburg), Dr. Dietmar Borchert

ment of a 'meagre' 50 million DM. The criticism is based on a report requested by Greenpeace from KPMG, the well-known Dutch auditors. According to the report, an investment of 629 million Euro would allow a solar cell plant with an annual output of 500 MW to be built, and the cost of solar-generated electricity to be reduced to 10p immediately, instead of after 20 years.

[990027]

new audio formats

high resolution and lots of features

Digital audio has become commonplace. Many things have changed since digital audio was introduced into the consumer market in the 1970s. The technology has been improved, and media capacities have increased considerably, partly due to the development of new generations of semiconductor lasers.

Many people find it difficult to imagine that music shops looked completely different twenty years ago. In a remarkably short time, vinyl records have been forced to make way for a very broad assortment of CDs, and recently also DVDs. The vinyl phonograph record has now practically disappeared from the face of the earth, along with the equipment used to play it.

A WORLD FULL OF CHANGES

After the first initiatives to digitise audio information for the consumer market in the 1970s, and the appearance of the first audio CDs in the market over fifteen years ago, a lot of effort has gone into developing successors to the CD that can lift the quality of the audio information to even higher levels. Although the audio CD represents the ultimate musical experience for many of us, there are certain technical weaknesses in its design. New developments are certain to change this situation.

A BIT OF HISTORY

The development of the audio CD in the '70s and '80s of the last century was in many regards a tremendous technical breakthrough. Traditional analogue recording techniques, which up to then consisted of grooves in vinyl disks or mag-

One of the first audio CD players from Philips.

By H. Steeman

netic tracks on tapes, were discarded.

The development of A/D and D/A converters was a difficult task in the early days of digital audio. A multibit PCM coding scheme with a relatively low sampling rate was chosen for digitising the audio signals. Based on an audio bandwidth of 20 kHz, the sampling rate was set to 44.1 kHz, with a resolution of 16 bits. At that time, these were top-level specifications that allowed very good quality to be achieved.

CHANGES ON SEVERAL FRONTS

The digital recording of audio information was not the only innovation. The use of the CD as a storage medium was also a technical breakthrough. For the first time, a semiconductor laser was used in a consumer product to read a digital optical storage medium at a high data rate. Nowadays, it's impossible to imagine our daily life without audio CDs.

The manufacturers of consumer electronics products find that the time is ripe to raise the quality of audio reproduction to a new level. Unfortunately, they have not been able to agree on a single new standard. Two new audio media have been introduced into the consumer market, where they will compete with each other in the coming years: Super Audio CD and DVD-Audio.

HIGH-DENSITY MEDIA

Any improvement in the quality of digital audio goes hand in hand with an increase in the amount of digital data that must be stored. Given that the dimensions of a CD ($4^3/_4$ inch or 120 mm) has more or less become an industry standard, it is inconceivable that a physically larger disk (and thus yet another new format) would be used. Furthermore, compatibility with existing devices and/or software is an important prerequisite for the successful introduction of a new system. This means that the storage capacity of the optical medium must be improved. It took a long time for consensus to be reached within the industry regarding a new generation of CDs, but this has finally come in the form of the DVD (Digital Versatile Disc).

DVD: ONE CONCEPT, FOUR FORMATS

Although a DVD appears to be the same as a normal CD to the naked eye, it is considerably more sophisticated. In the first place, a different type of laser is used to read a DVD. This laser uses light with a shorter wavelength (635 to 650 nm instead of 780 nm). This makes the light spot that is used to read the DVD much smaller. This in turn means that the length of the digital data pits can be reduced from 0.83 to 0.4 micron. At the same time, the tracks can be made narrower, so that adjacent tracks can be placed closer together. The track pitch of the DVD has thus been reduced to 0.74 micron, compared to 1.6 micron for a normal CD. These measures combine to increase the storage capacity from 640 MB to 4.7 GB. However, even this is inadequate for some applications. Consequently, there are four formats specified for the DVD: DVD-5, DVD-9, DVD-10 and DVD-18. In addition to these formats, there are also agreements regarding (re)recordable DVDs, but these fall outside the scope of this article.

The smallest DVD has a storage capacity of 4.7 GB, while the largest has a capacity nearly four times as large (17.1 GB). All DVDs employ two substrates (transparent plastic layers carrying the digital information) that are 0.6 mm thick. With traditional CDs, only one substrate is used, with a thickness of 1.2 mm. Since the total capacity of a single substrate layer is 4.7 GB for all DVDs, it follows that the maximum capacity can only be achieved by using two information layers on top of each other on each side of the disk, as well as by using both sides of the disc. Let's have a closer look at all four of these formats.

DVD-5 (4.7 GB)
Single-sided, one layer

This format is the most elementary member of the DVD family. The disc has a storage capacity of 4.7 GB. Only one of the two substrates has an information layer. However, the two substrates are combined (glued together) to produce a single substrate that is 1.2 mm thick.

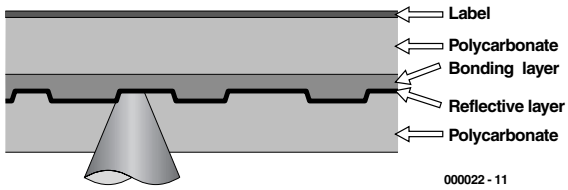


Figure 1. Construction of a DVD disc with one optical data layer (DVD-5).

DVD-9 (8.5 GB)
Single-sided, two layers

This DVD variant is single-sided, but it uses two layers on the one side. This results in a capacity of 8.5 GB, which is a bit less than twice 4.7 GB. The difference is due to measures used by the manufacturers to improve the readability of the second layer. The resulting bit pattern on both layers is thus 10% larger than that of the DVD-5 and DVD-10 formats.

Each layer is located on a 0.6-mm thick substrate, with one of the layers being semitransparent (gold coloured). The two substrates are glued together using a transparent (optically neutral) adhesive to make up a single assembly. The storage capacity is more than adequate for recording a complete feature film with high quality (MPEG2 format), and it also provides room for the storage of supplementary data, including eight tracks of audio information.

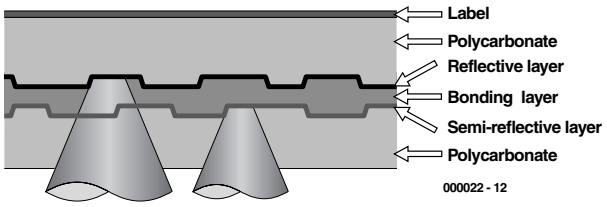


Figure 2. Construction of a DVD disc with two optical data layers on one side (DVD-9).

DVD-10 (9.4 GB)
Two-sided, one layer per side

This DVD variant uses both sides of the disc, with a digital

Table 1. Comparative physical specifications of DVDs and standard CDs.

Specification	*CD*	*DVD*	*DVD*
Number of layers	*single*	*single*	*double*
Substrate thickness	*1.2 mm*	*0.6 mm*	*0.6 mm*
Number of substrates per disc	*1*	*2*	*2*
Track separation	*1.6 micron*	*0.74 micron*	*0.74 micron*
Minimum pit length	*0.83 micron*	*0.4 micron*	*0.44 micron*
Reading speed (m/s)	*1.3*	*3.49*	*3.84*
Laser wavelength (nm)	*780*	*635/650*	*635/650*
Numerical aperture (NA)	*0.45*	*0.6*	*0.6*
Modulation	*EFM*	*8-to-16*	*8-to-16*

storage layer with a capacity of 4.7 GB on each side. In order to play back both sides, the CD must be turned over. For the time being, this must be done manually, although it is in theory possible to make CD players that can read both sides of the disc without requiring it to be turned over by hand.

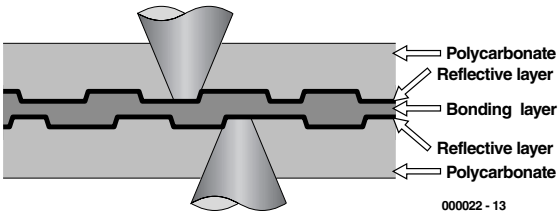


Figure 3. Construction of a DVD disc with one optical data layer on each side (DVD-10).

DVD-18 (17.1 GB)
Two-sided, two layers per side

The DVD-18 format is the variant with the largest imaginable storage capacity. A double data layer is present on each side of the disc, so that each disc contains a total of four data layers.

With this format, the two data layers on each side must be placed on a single polycarbonate substrate. A photographic process is used to produce the data layer that is located in the middle of the substrate. With regard to the manufacturing process, this means that a production system that transfers the bit pattern to the photosensitive polymer must also be available, in addition to the usual DVD press. The production of DVDs according to this specification is very difficult and presently quite expensive.

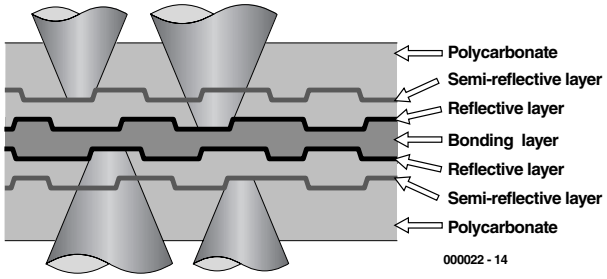


Figure 4. Construction of a DVD disc with two optical data layers on each side (DVD-18). The fabrication of this format is presently very difficult and expensive.

disc with two data layers is used. The information layer that a normal CD player reads lies at the standard depth of 1.2 mm, while a special layer with high storage density is located at a depth of 0.6 mm. The laser of a standard CD player, operating at a wavelength of 780 nm, focuses on the optical layer located at a depth of 1.2 mm and thus looks through the high-density layer at 0.6 mm. The laser of a SACD player, with a wavelength of 650 nm, focuses on the high-density layer and does not see the 'standard CD' layer.

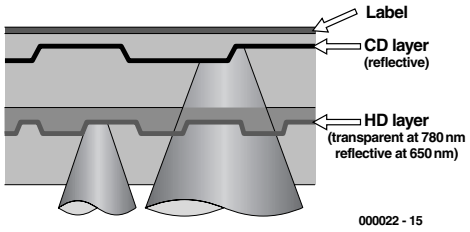


Figure 5. Construction of a Super Audio CD. Two different optical data layers are present on one side of the disc. The layer that is read depends on the type of laser used.

MORE INFORMATION ON THE DISC

Up to now, we have described how the industry has increased the amount of data that can be stored on a disc with a diameter of 120 mm from 640 MB to a maximum of 17.1 GB. In order to make optimum use of this storage capacity, and to also attach a user advantage to it, the recording and coding techniques have also been improved.

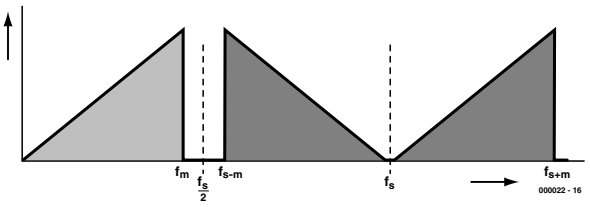


Figure 6. When an audio signal is sampled, a mirror image of the sampled frequency spectrum (f_m) is created on the other side of the sampling frequency (f_S).

SUPER AUDIO CD: THE ODD MAN OUT

As already mentioned, the music industry has introduced the Super Audio CD in addition to the DVD. As a data medium, this is a mixture of a DVD and a normal CD. The developers of this concept, Philips and Sony, needed a storage medium with DVD capacity to accommodate the desired improvements in the quality of the audio information, but they absolutely wanted to maintain compatibility with the traditional CD. The Super Audio CD (SACD) can thus be used with all existing CD players as well as with the new SACD players. Only in the latter case are the improved quality and extra features available. Here again, a single-sided

A compact DVD player for reproducing audio CDs, video DVDs and special video DVDs containing audio tracks.

As already noted, the original audio CD is based on multibit PCM coding with a resolution of 16 bits. Presently, digital technology has advanced so much that speed itself is absolutely no problem. Philips and Sony, who together invented both the standard CD and the Super Audio CD, have over the last several years developed a new recording and coding system that is called Direct Stream Digital (DSD). This works with a resolution of 1 bit and a sampling rate of 2.8224 MHz ($64 \cdot f_s$). **Figure 6** shows how the audio signal is digitised. When an audio signal (f_m) is sampled at a sampling frequency f_s, then two 'mirror' images of the audio signal are generated, one on either side of the sampling frequency. In order to prevent these mirror images from overlapping, the sampling frequency must be chosen to be as high as possible. With the standard audio CD, the bandwidth of the audio spectrum is 20 kHz and the sampling rate is 44 kHz. The original signal and the lowest part of the mirror signal thus lie quite close to each other. Strong filters that sharply cut off the audio spectrum just above 20 kHz are thus essential. Such filters can also affect the signals within the audio spectrum. If a high sampling frequency ($64 \cdot f_s$) is used instead, the filtering problem disappears, since the resulting image frequencies lie in a completely different part of the spectrum from the audio frequencies. In addition, the resolution of the A/D converter can be dramatically reduced, in this case to a single bit. For each sample, the converter checks whether the amplitude of the new sample is greater than or less than that of the previous sample. This type of digitisation, which is called delta-sigma conversion, forms the heart of a DSD converter. If the conversion error (quantisation error) of the previous conversion is taken into account in each new conversion (which is referred to as noise shaping), the quality of the sampling process can be improved even further. **Figure 7** illustrates this process. The result is a digitised audio signal with a bandwidth of 100 kHz (five times as large as that of a normal audio CD) and a dynamic range of 120 dB, compared to 96 dB for a standard audio CD. To achieve these results, four times as much digital data are needed as for a standard audio CD. A normal CD recording that takes up 600 MB thus requires around 2.4 GB in DSD format.

The audio specifications of DSD are so good that, with the present state of knowledge, it is not better to make a better recording. Philips and Sony have advised all recording studios to make their master recordings in the DSD format (although the fact that the DSD format first appeared on the market in mid-1999 means that very few studios have suitable equipment available, but that's a different issue).
In order to guarantee compatibility with the existing PCM technology, Sony developed what it calls Super Bit Mapping Direct. **Figure 8** shows how all standard PCM formats can be derived from a DSD signal using this technique. It is possible to convert the signal to the PCM format with various resolutions (16, 18, 20 and even 24 bits). If other formats are thought up in the future, they can also be derived from the DSD format.
It is generally thought that the audio industry will be able to use the DSD format for the next twenty years.

DSD is the best possible recording technique, and the data stored on the high-density layer of a Super Audio CD are in DSD format. Since a maximum-length recording (74 minutes) does not use the full storage capacity, there is room left over for other information. In this regard, there is consideration being given to multi-channel recording, graphic information and text. Although Super-Audio CD players are presently outside the price range of the average consumer, it is expected that the prices of DSD decoders will drop so much in the coming years that they will be built into more and more equipment.

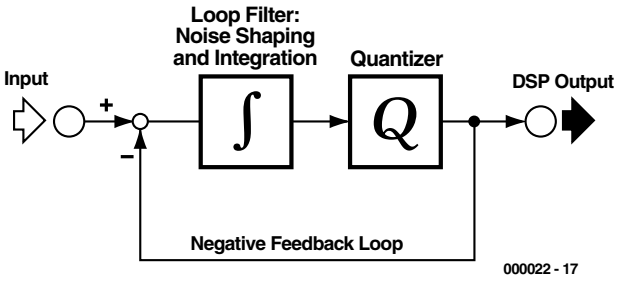


Figure 7. A DSD converter employs a 1-bit delta-sigma converter. If proper feedback is applied, the noise level drops and the signal quality is improved.

DVD-Audio

In addition to the Super Audio CD, an audio standard based on the DVD has also been worked on for the last few years. The resulting formats were definitively laid down in a standard in April of 1999.

This standard is based on two DVD formats, which are called DVD-Audio and DVD-AudioV. These two formats stand in addition to DVD-Video, which is the format that has been developed for the distribution of films. Future DVD players should be able to play back all DVD formats, and computers can work with them as well.
DVD-Audio offers new possibilities for the user, including improved sound quality (due to an increased sampling frequency and higher resolution), surround sound (multi-channel sound), longer playing time and supplementary functions. The storage capacity of DVD-Audio with two layers is adequate for at least two hours of very high quality surround sound or four hours of stereo sound. The specifications of a disc with only one layer are, overall, the half of these. In order to increase the storage capacity, a lossless compression technique called Meridian Lossless Packing has been specially developed for DVD-Audio. This increases the storage capacity so much that at least 74 minutes of audio can be recorded on a single DVD side. The hardware in the DVD player can

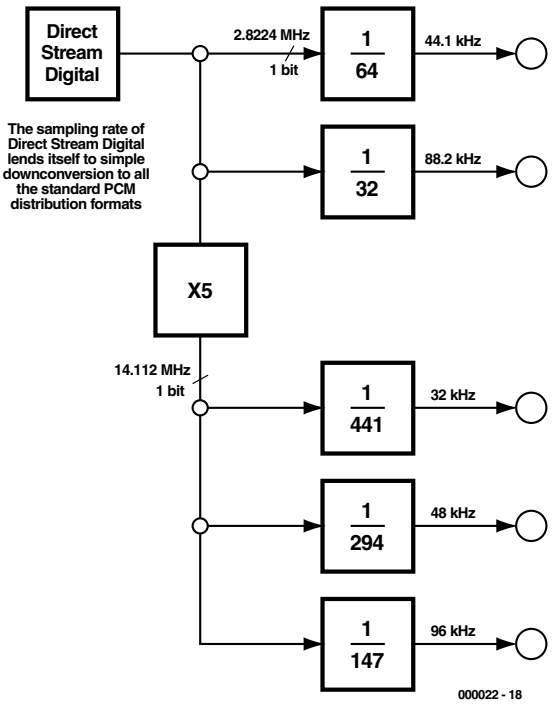


Figure 8. The Super Bit Mapping Direct technique can be used to derive any desired PCM format from the DSD format, with regard to both sampling rate and resolution.

Up to now, only Sony supplies SACD players. These are still quite expensive; the illustrated model costs around £4000.

handle this compression without imposing too much of a load on the processor.

In addition to all these attractive specifications, additional measures have been taken with both DVD-Audio and the Super Audio CD to make copying more difficult. The average user will (for the time being) not be able to make copies that have the same quality as the originals. Criminal organisations will have difficulty bringing copies on the market that resemble the originals closely enough that the unwitting consumer will not be able to tell the difference between an original and a copy. The special watermarks that can be applied to the data side cannot easily be reproduced.

In light of the fact that the anti-copy protection of DVD-Video discs using the Content Scrambling System (CSS) has recently been cracked, and that the cracking program DeCSS was freely distributed via the Internet, it is a good question how long these security measures will remain effective. A variation of the CSS is also used for DVD-Audio.

DVD-Audio in practice

The maximum data rate for DVD-Audio is 9.6 Mbit/s. This means that the maximum sampling rate for multi-channel audio is limited to 96 kHz. In order to make the best possible use of the available bandwidth, it is possible to apply different specifications to the various channels. For example, the left, right and centre audio channels could be sampled at 96 kHz with 24-bit PCM resolution, while the two rear channels could 'make do' with a sampling rate of 48 kHz and 16-bit coding. Other coding schemes can be used in addition to PCM, such as Dolby Digital (AC3), Digital Theatre Sound (DTS), MPEG1 stereo or MPEG2 multi-channel audio (although the MPEG audio format appears to have already lost the race).

DVD-AudioV is a supplementary standard that combines audio with video, such that the audio portion can be played back on special DVD-Audio player and the video portion on a normal DVD video player.

Standard DVD-Video as an audio medium

As a complement to these audio formats, it is interesting to note that a DVD-Video disc can also have eight audio channels. In this regard, the disc manufacturer can make a trade-off between multiple mono channels, for example for speech, and multi-channel sound recording. At least three multi-channel audio recordings can be combined with a video signal.

In the mean time, several music producers (such as Denon and Chesky) have been making video DVDs containing only two audio tracks of 96 kHz, 24-bit audio with very good quality. As long as no hardware and software are available for DVD-Audio, these special recordings can be regarded as an excellent alternative.

After this summary of the new audio formats, we must ask ourselves if the consumer is really interested in all of this. DVD-Video can be considered to be a successful new format, but this clearly represents a qualitatively high-value combination of video and audio. It is doubtful that the true audio fan will be overly enthusiastic about two new audio media.

(000022-1)

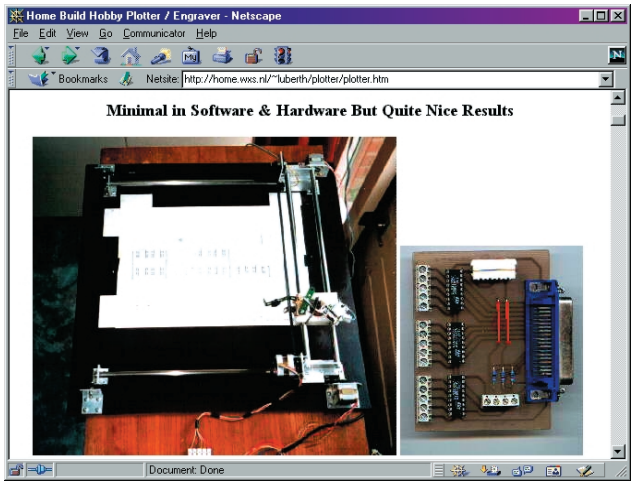
Primary DVD-Audio and DVD-Video specifications.

	DVD-Audio	*DVD-Video*
Coding	*linear or compressed PCM*	*linear PCM, Dolby AC-3 or DTS*
Sampling frequency	*44.1/48/88.2/96/176.4/192 kHz*	*48/96 kHz*
Resolution	*16/20/24 bits*	*16/20/24 bits*
Number of channels	*6 (96 kHz maximum) 2 (176.4/192 kHz maximum)*	*8*
Maximum bit rate	*9.6 Mb/s (linear PCM)*	*6.144 Mb/s*

build your own plotter

drawing, cutting and milling with stepper motors

A plotter is a drawing device which is typically not affordable for private use, particularly when large paper formats are involved. However if you feel you have suitable stepper motors and the necessary mechanical skills, why not have a go at building a plotter yourself? The results may more than adequate.



Long-standing readers of this magazine are sure to remember our Plotter design described in the May and June 1988 issues. This was followed by a Mark II version (March 1990) and extended control software (September 1991). These were once revolutionary articles, but technology has progressed since.

We were a little surprised to see that there are still hobbyists who not only design and build their own plotters, but are also willing and able to share their knowledge with other interested people. The Internet contains a vast amount of such information which is accessible to everyone.

The first enthusiast we should mention here is Luberth Dijkman who, a big plotter fan himself, drew our attention to this subject. His plotter page at
http://home.wxs.nl/~luberth/plotter/plotter.htm
is a kind of repository of material on plotters. It contains, among others, information on Luberth's own DIY plotter, and a large number of programs for driving such a device from a computer. Software is available for different 'platforms' as well as in different programming languages (QBASIC, Pascal, C) — there's something to suit everyone! In addition to the control programs (adapted for different types of driver IC), there's a collection of conversion programs, for example, for conversion of true-type fonts into vector format.

The site is also valuable for its collection of links to other internet sites covering the same subject, ranging from complete DIY descriptions of plotters and fraising machines to suppliers of parts kits for home construction.

Luberth also presides and moderates the **Homebuilt Plotter Discussion Forum** which is open to everyone with a question on plotters.

Another interesting plotter page is **Scumari Technohobbies** run by Rick Schuitemaker at
http://www.scumari.demon.nl/plotter/plotnl.htm

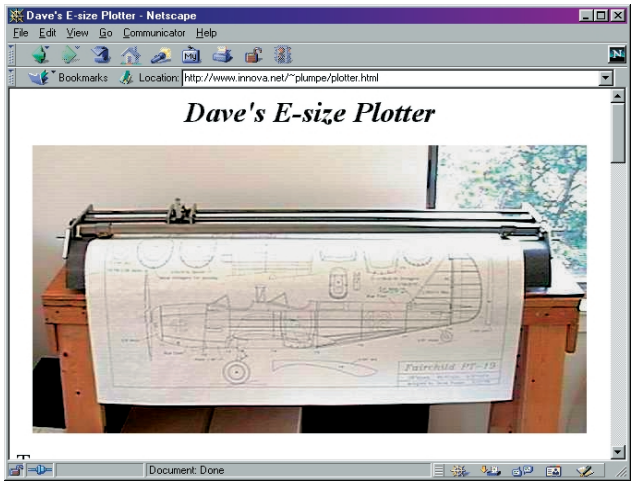
A beautiful home-made CNC fraising machine is shown on the web by **BCS Computerssysteme** from Germany. It should be noted though that this device is a private development and that BCS does not supply any parts for the project. The URL is
http://www.bcs-computersysteme.com/software/cnc/index.htm
Even in Brazil, plotters are being designed and built, as may be seen at
http://www.geocities.com/ResearchTriangle/3215/plotter.htm-
Remarkably, this site contains link to Luberth Dijkman's site!

If you want to know more about building a really large plotter, take a look at **Dave's E-Size Plotter** at
http://www.innova.net/~plumpe/plotter.html

Dave Rigotti is one of a number of private persons offering components fr building a plotter, like stepper motors and driver ICs:
http://hometown.aol.com/drigotti/HobbyCNC.htm

If you do not mind a lot of mechanical work, then lots of inspiration may be obtained from the websites mentioned here.

(005021-1)



MEMORY FUNCTION EXAMPLE

Example: Bus master sets up DS1621 for continuous conversion and thermostatic function.

BUS MASTER MODE	DS1621 MODE	DATA (MSB FIRST)	COMMENTS
TX	RX	START	Bus Master initiates a START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	ACh	Bus Master sends Access Config command protocol.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	02h	Bus Master sets up DS1621 for output polarity active high, continuous conversion.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	A1h	Bus Master sends Access TH command.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	28h	Bus Master sends first byte of data for TH limit of +40°C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	00h	Bus Master sends second byte of data for TH limit of +40°C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	A2h	Bus Master sends Access TL command.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	0Ah	Bus Master sends first byte of data for TL limit of +10°C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	00h	Bus Master sends second byte of data for TL limit of +10°C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	EEh	Bus Master sends Start Convert T command
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	STOP	Bus Master initiates STOP condition.

DS1621

Integrated circuits
Special Function

ELEKTOR ELECTRONICS

DATASHEET 3/2000

DS1621
Digital Thermometer and Thermostat

Manufacturer:
Dallas Semiconductor. Internet:
http://www.dalsemi.com

Features

- Temperature measurements require no external components
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to 257°F in 0.9°F increments
- Temperature is read as a 9-bit value (two byte transfer)
- Wide power supply range (2.7V to 5.5V)
- Converts temperature to digital word in 1 second
- Thermostatic settings are user definable and non-volatile
- Data is read from/written via a 2-wire serial interface (open drain I/O lines)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermal sensitive system.
- 8-pin DIP or SOIC package (150 MIL and 208 MIL)

Description

The DS1621 digital thermometer and thermostat provides 9-bit temperature readings which indicate the temperature of the device. The thermal alarm output, T_{OUT}, is active when the temperature of the device exceeds a user-defined temperature TH. The output

DC ELECTRICAL CHARACTERISTICS (–55ºC to +125ºC; V_{DD}=2.7V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Thermometer Error	T_{ERR}	0ºC to 70ºC			$\pm^1/_2$	ºC	1
Low Level Input Voltage	V_{IL}		–0.5		0.3 V_{DD}	V	
High level Input Voltage	V_{IH}		0.7 V_{DD}		V_{DD}+0.5	V	
Pulse width of spikes that must be suppressed by the input filter	t_{SP}	Fast Mode	0		50	ms	
Low Level Output Voltage	V_{OL1}	3 mA sink current	0		0.4	V	
	V_{OL2}	6 mA sink current	0		0.6	V	
Input Current each I/O Pin		$0.4<V_{I/O}<0.9V_{DD}$	–10		10	µA	2
I/O Capacitance	$C_{I/O}$				10	pF	
Active Supply Current	I_{CC}	Temperature Conversion			1000	µA	3, 4
		E^2 Write			400		
		Communication Only			100		
Standby Supply Current	I_{STBY}				1	µA	
Thermostat Output (T_{OUT}) Output Voltage	V_{OH}	1 mA source	2.4			V	
	V_{OL}	4 mA sink			0.4	V	

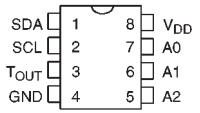
Notes:
1. Thermometer error reflects sensor accuracy as tested during calibration.
2. I/O pins of fast mode devices must not obstruct the SDA and SCL lines if V_{DD} is switched off.
3. I_{CC} specified with T_{OUT} pin open.
4. I_{CC} specified with V_{CC} at at 5.0V and SDA, SCL = 5.0V, 0ºC to 70ºC

remains active until the temperature drops below user defined temperature TL, allowing for any hysteresis necessary.
User defined temperature settings are stored in non-volatile memory, so parts may be programmed prior to insertion in a system. Temperature settings, and temperature readings are all communicated to/from the DS1621 over a simple 2-wire serial interface.

Application example
DS1621 Temperature sensor with I²C control, *Elektor Electronics* March 2000.

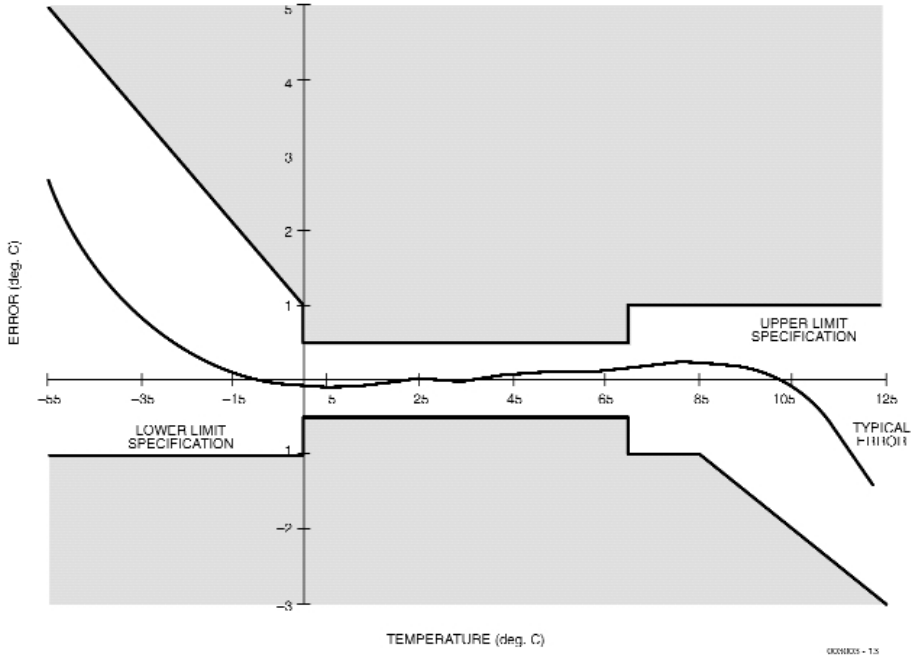
Pin description

SDA	2–Wire Serial Data Input/Output
SCL	2–Wire Serial Clock
GND	Ground
T_{OUT}	Thermostat Output Signal
A0	Chip Address Input
A1	Chip Address Input
A2	Chip Address Input
V_{DD}	Power Supply Voltage



DS1621
8–PIN DIP (300 MIL)

003003 - 11



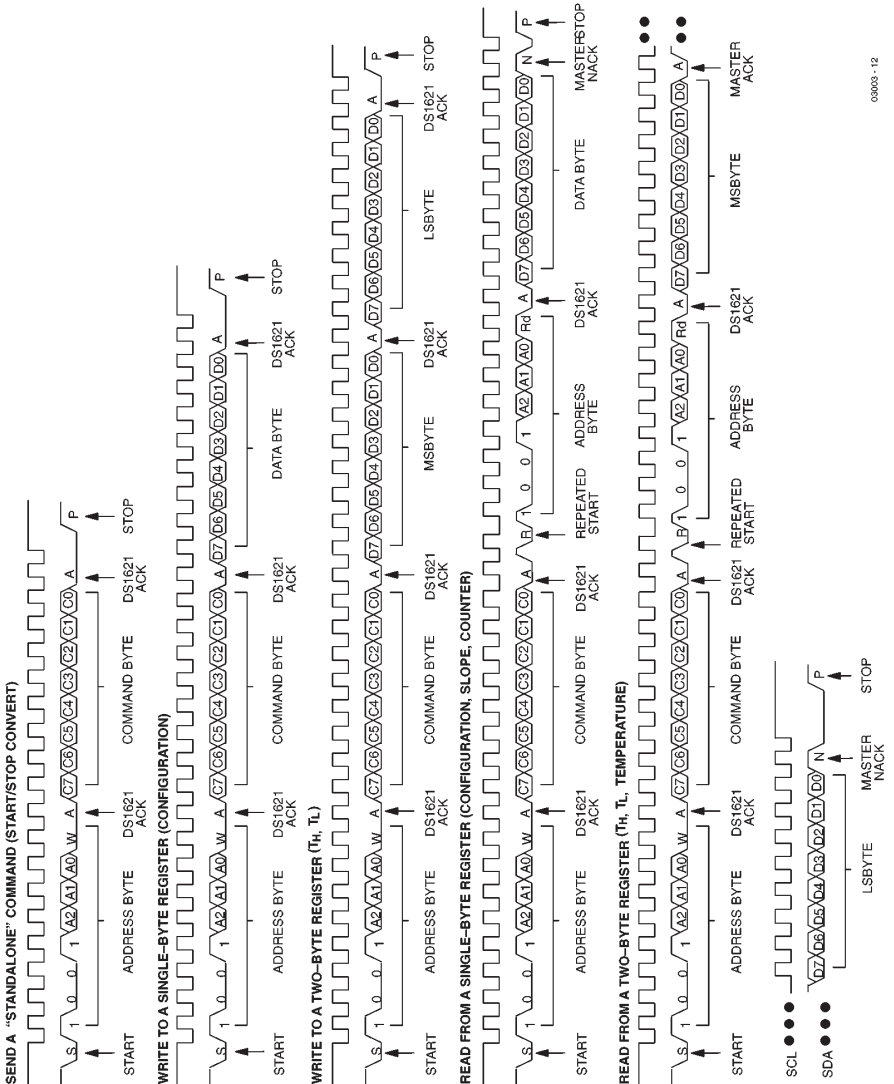
Typical performance curve

DS1621

Integrated circuits
Special Function

ELEKTOR ELECTRONICS

DATASHEET 3/2000



2-wire serial communication with DS1621

The thing that makes this card different from other DAQ & Control cards for the PC serial port is the way it communicates with the computer. Many DAQ cards you may have seen include a microcontroller or a UART chip for serial communication. The present card does not have any such chips because it employs direct accessing of UART registers to enable serial to parallel conversion.

Design by George Vastianos,
Student at Electronics Department, Technological Educational Institute of Piraeus, Greece

32-channel digital input card for PC serial port

link your PC to the real world

Main specifications

Number of Inputs:	32
Type of Inputs:	TTL compatible ("0" = 0 V, "1" = 5 V)
Connection:	Serial Port
Communication:	Direct access of UART registers
Power Supply:	9-15 V DC

The field of Data Acquisition (DAQ) & Control is not something new for Electronics Science. The field really started out just after Semiconductor Technology succeeded in making the first microprocessor in an Integrated Circuit (IC) package. These were the days when computers started to take a secure position in industry, and after that, in our homes.

Data Acquisition & Control is all about processing of data and controlling through computers. So the goal of all DAQ & Control systems is to allow the computer to communicate with, and to some extent control, the 'outside world'.

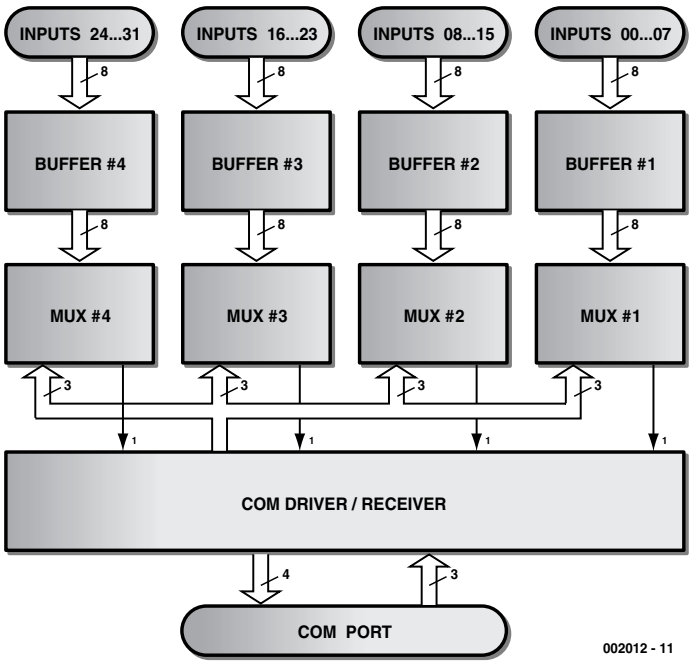


Figure 1. 32-channel D/I card block diagram.

The applications of DAQ & Control systems have to do with industry, where the use of the right input/output cards enables automation and control systems to solve complex tasks.
A DAQ & Control Card may have inputs only, outputs only or a combination of these. Each input or output may be analogue or digital. Another characteristic of these cards is the way they are connected with the computer or computer system. So, many cards have been designed for installation in one of the local buses of the main board (ISA, EISA, PCI etc), or for connection to the available ports of the computer (parallel, serial, game, keyboard ports).
The project described in this article is a card with 32 digital inputs (32 Channel D/I Card) for external connection to the serial (RS232) port on your IBM PC or compatible.

About the serial port

Serial ports are used mainly for communication between computers, or for communication between a computer and peripherals like a modem or a mouse. The controller at the heart of this port is almost invariably a UART (Universal Asynchronous Receiver Transmitter) chip found on the main board. This chip works as a serial-to-parallel and parallel-to-serial adapter.

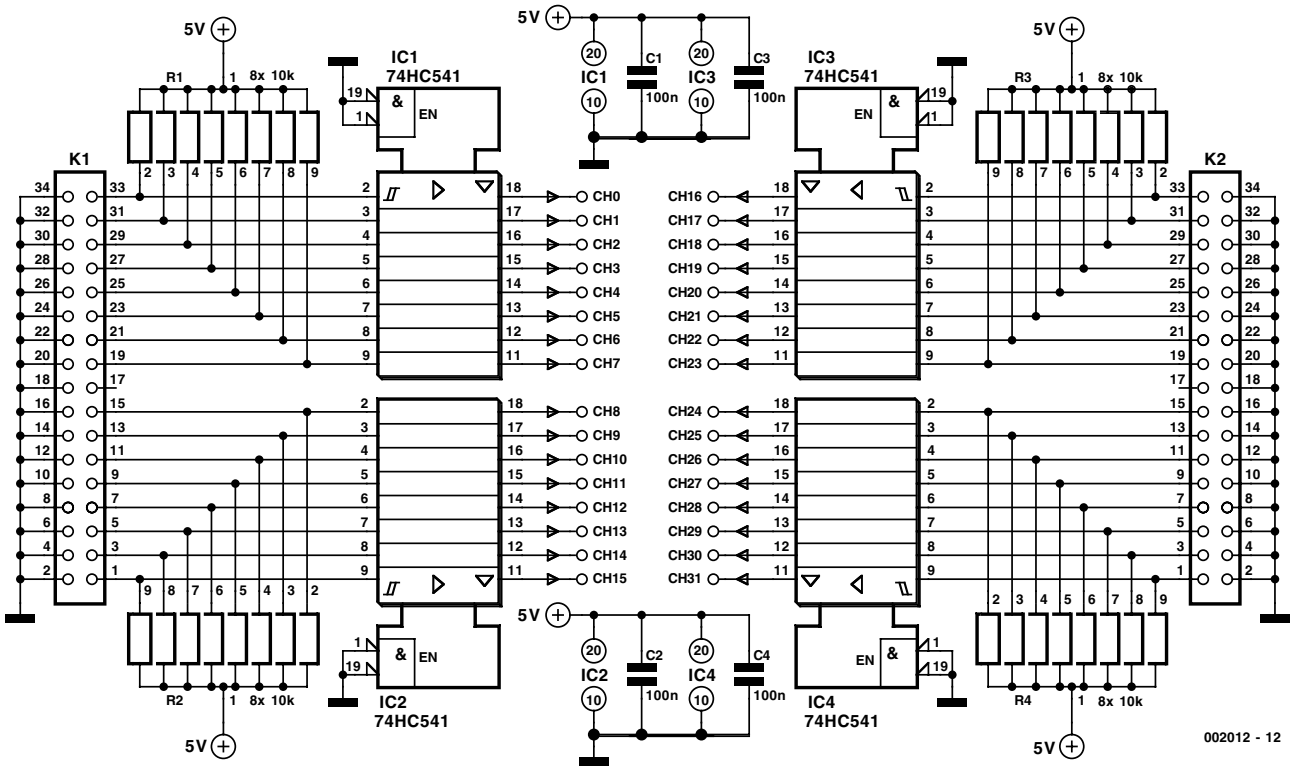


Figure 2. Buffers Unit circuit diagram.

A computer may have one to four serial ports (COM1 to COM4), where each port occupies eight locations in its I/O memory area. See **Table 1** for the relevant details.

The basic lines used by a UART in serial communication for transmission and reception are called TxD and RxD. Also a group of extra lines (DCD, DSR, RTS, CTS, DTR, RI) is used to establish different types of serial communication. Although some of these extra lines work as inputs and others as outputs, each one (except RxD) may controlled through a bit of a UART register. **Table 2** summarizes the interface pin connections and system I/O addresses.

The voltage levels used on the serial port (RS232 levels) are different from TTL levels. In RS232 lingo, a logic '1' is represented by a voltage of –12V, and a logic '0' by +12V.

Table 1.

Transmit/Receive Buffer	3F8h	2F8h	3E8h	2E8h
Interrupt Enable Register	3F9h	2F9h	3E9h	2E9h
Interrupt Identification Register	3FAh	2FAh	3EAh	2EAh
Line Control Register	3FBh	2FBh	3EBh	2EBh
Modem Control Register	3FCh	2FCh	3ECh	2ECh
Line Status Register	3FDh	2FDh	3EDh	2EDh
Modem Status Register	3FEh	2FEh	3EEh	2Eeh

The hardware

In the block diagram of the circuit (**Figure 1**), the available inputs have been divided in four groups of eight inputs (00-07, 08-15, 16-23, 24-31), and they all enter the Buffers Unit. After that, all the lines leaving the Buffers Unit enter the Multiplexers Unit, where only one input of each group is selected. The four selected inputs pass through the COM Driver/Receiver Unit (where they are converted from TTL compatible to RS232 compatible) and arrive at four serial port inputs (CTS, DSR, RI, DCD). For the selection of the inputs, we use the three outputs of on the serial port (TXD, DTR, RTS). Having passed the COM Driver/Receiver Unit (and being adapted from RS232 compatible to TTL compatible), the output signals arrive on the Multiplexers address inputs.

Table 2.

Pin Name	Pin # on 25-pin connector	Pin # on 9-pin connector	COM1	COM2	COM3	COM4	Bit	I/O
TxD	2	3	3FBh	2FBh	3EBh	2EBh	6	O
DTR	20	4	3FCh	2FCh	3ECh	2ECh	0	O
RTS	4	7	3FCh	2FCh	3ECh	2ECh	1	O
CTS	5	8	3FEh	2FEh	3EEh	2EEh	4	I
DSR	6	6	3FEh	2FEh	3EEh	2EEh	5	I
RI	22	9	3FEh	2FEh	3EEh	2EEh	6	I
DCD	8	1	3FEh	2FEh	3EEh	2EEh	7	I

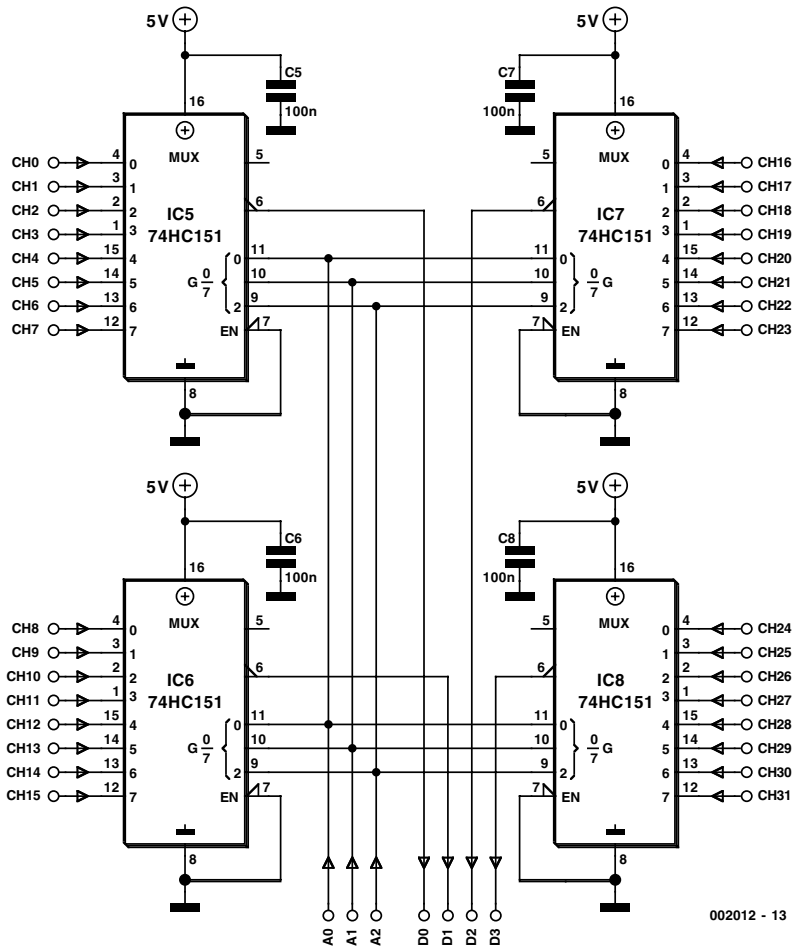


Figure 3. Multiplexers Unit circuit diagram.

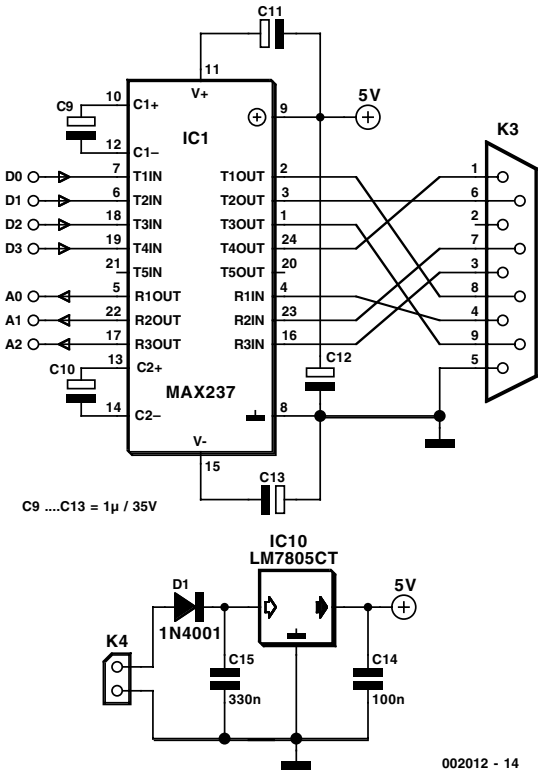


Figure 4. COM Driver/Receiver and Power Supply circuit diagram.

Buffers Unit

In the schematic, **Figure 2**, the 32 inputs have been divided in two groups of 16 (for easy PCB design) and enter the circuit through connectors K1 and K2. The correspondence between the inputs and the connectors can be seen in **Table 3**.

All inputs are fitted with pull-up resistors R1-R4 (10 kΩ) to establish termination in case where one or more inputs are not connected. The buffers are the four chips IC1-IC4 (74HC541). The four capacitors C1-C4 (100 nF) work as bypass capacitors to improve the stability of the circuit. The Output Enable control inputs (pins 1 and 19) of the 74HC541s are connected to ground to make the buffers work continuously.

Multiplexers Unit

As shown in **Figure 3**, the multiplexers are in reality four 74HC151s (IC5-IC8). Four capacitors C5-C8 (100 nF) are added to ensure adequate supply decoupling. The Output Enable control input (pin 7) of each 74HC151 is connected to ground to make each multiplexer work continuously. The A, B, C address inputs of all 74HC151 are connected together to implement multiplexing of all 32 inputs.

COM Driver/Receiver Unit

The last unit includes a voltage regulator so that the card will not need a regulated power supply to work. In the schematic circuit of the COM Driver/Receiver Unit (**Figure 4**), IC9 (MAX237) works as an RS232 Driver/Receiver. It has 3 channels converting from RS232 to TTL and 5 channels converting from TTL to RS232. Five satellite capacitors (C9-C13; 1 μF, 35V max. working voltage) enable the MAX237 to perform voltage doubling (so the Driver section can produce a voltage of 10 V for the five RS232 outputs, using a 5 V supply voltage). Inside the MAX237, each channel has an inverter. To overcome this problem we use the four inverted outputs of the four 74HC151s (pin 6), so that after inverting two times we have no inverting at all. IC10, a 7805 together with two capacitors C14 and C15, steps down the supply voltage to 5 V. Diode D1 protects the circuit against damage from supply polarity reversal.

The control software

The software for the communication with the card was developed in QBasic. The communication routine is called CARD32DI and its source code

Figure 5. Screendump of the demo program in action.

may be found in **Listing 1**.
If you call this routine (from any program written in QBasic), follow this syntax:

```
CALL CARD32DI (COMADDRESS, CHANNEL (), DATA0, DATA1, DATA2, DATA3)
```

Where:
COMADDRESS: Integer type variable, which (before calling) must contain the base address of the serial port. Acceptable values of this variable are &H3F8 (for COM1), &H2F8 (for COM2), &H3E8 (for COM3), &H2E8 (for COM4).

CHANNEL (): Matrix Integer type variable (with pointers from 0 to 31), which (after calling) contains the logic state of each channel (values 1 or 0).

DATA0, DATA1, DATA2 & DATA3: Integer type variables that (after calling) contain the arithmetic value of each group of eight channels (00-07, 08-15, 16-23, and 24-31). The logic states of all 32 channels make a Double Word (32 bit) with Ch0 as the LS Bit and Ch31 as the MS Bit. This Double Word may be expressed though the four bytes DATA0, DATA1, DATA2 & DATA3 where LS Byte is DATA0 and MS Byte is DATA3. Employ these variables in cases where you want to save the logic states of all channels in a file. The 'compresssion' allows you to save only 4 bytes instead of a whopping 32.
A demonstration program has been developed to test the 32-channel D/I card. A screenshot of this program is shown in **Figure 5**.
To change the serial port address use the keys <1> to <4>. If you want to quit, just press <Esc>.

How to obtain the software

The source code of the communication routine (CARD32DI.SUB) and the demonstration program (32DICARD.BAS), with an executable version of the demonstration program (32DICARD.EXE) may be obtained via this website

Table 3.

K1		K2	
Channel	**Pin Number**	**Channel**	**Pin Number**
00	33	16	33
01	31	17	31
02	29	18	29
03	27	19	27
04	25	20	25
05	23	21	23
06	21	22	21
07	19	23	19
08	15	24	15
09	13	25	13
10	11	26	11
11	9	27	9
12	7	28	7
13	5	29	5
14	3	30	3
15	1	31	1

http://members.xoom.com/robofreak/download/32dicard.htm

which for the actual downloading will take you to the author's website at

http://www.robofreak.xs3.com

Finally, the author may be reached by email on *sebastian@mail.kapatel.gr*

(002012-1)

```
REM ************************************
REM *        32 Channel D/I Card         *
REM *   CARD32DI Communication Routine   *
REM *     Copyright (c)  April 1999      *
REM *        by George Vastianos         *
REM *  email:robofreak@technologist.com  *
REM * http://members.xoom.com/robofreak/ *
REM ************************************
'
SUB CARD32DI (COMADDRESS, CHANNEL(), DATA0, DATA1, DATA2, DATA3)

  DATA0 = 0: DATA1 = 0: DATA2 = 0: DATA3 = 0

  FOR BIT = 0 TO 7

    IF (BIT AND 1) = (INP(COMADDRESS + 4) AND 1) THEN
      OUT (COMADDRESS + 4), INP(COMADDRESS + 4) XOR 1
    END IF
    IF (BIT AND 2) = (INP(COMADDRESS + 4) AND 2) THEN
      OUT (COMADDRESS + 4), INP(COMADDRESS + 4) XOR 2
    END IF
    IF (BIT AND 4) = (INP(COMADDRESS + 3) AND 64) / 16 THEN
      OUT (COMADDRESS + 3), INP(COMADDRESS + 3) XOR 64
    END IF

    OUT COMADDRESS + 1, 0
    OUT COMADDRESS + 2, 0

    INDATA = INP(COMADDRESS + 6) AND 240

    CHANNEL(BIT) = (INDATA AND 16) / 16
    CHANNEL(BIT + 8) = (INDATA AND 32) / 32
    CHANNEL(BIT + 16) = (INDATA AND 64) / 64
    CHANNEL(BIT + 24) = (INDATA AND 128) / 128

    DATA0 = DATA0 + CHANNEL(BIT) * 2 ^ BIT
    DATA1 = DATA1 + CHANNEL(BIT + 8) * 2 ^ BIT
    DATA2 = DATA2 + CHANNEL(BIT + 16) * 2 ^ BIT
    DATA3 = DATA3 + CHANNEL(BIT + 24) * 2 ^ BIT

  NEXT BIT

END SUB
```

With a set of Maxim A/D and D/A converters and a handful of other components, you can make a curve tracer that can be driven from the printer port of a PC. A simple BASIC program takes care of the communication between the PC and the measurement circuit, and also converts the measured values into graphic form for display on the monitor.

Source: Maxim Integrated Products

I/U curve tracer

driven via the PC printer port

Not long ago, it was common practice to measure the I/U characteristics of a semiconductor device using an X-Y oscilloscope and a suitable measuring circuit. An electronics engineer can extract a lot of information about the operation of the semiconductor from these curves.

In the present computer age, the I/U curve may well have dropped into obscurity, but it is still useful for evaluating a number of specific semiconductor characteristics. The only difference is that we no longer need an oscilloscope, since we can instead employ the versatile PC.

With the measurement circuit presented here, which comes from Maxim, the I/U characteristics of a semiconductor or IC can be measured and then displayed on the monitor. This is made possible by the use of two serial-interface ICs: a 12-bit DAC and a 12-bit ADC. A short BASIC program looks after controlling the circuit and displaying the data on the screen. Since the circuit is designed for use with a standard PC printer port, the interaction between the circuit and the computer is very simple.

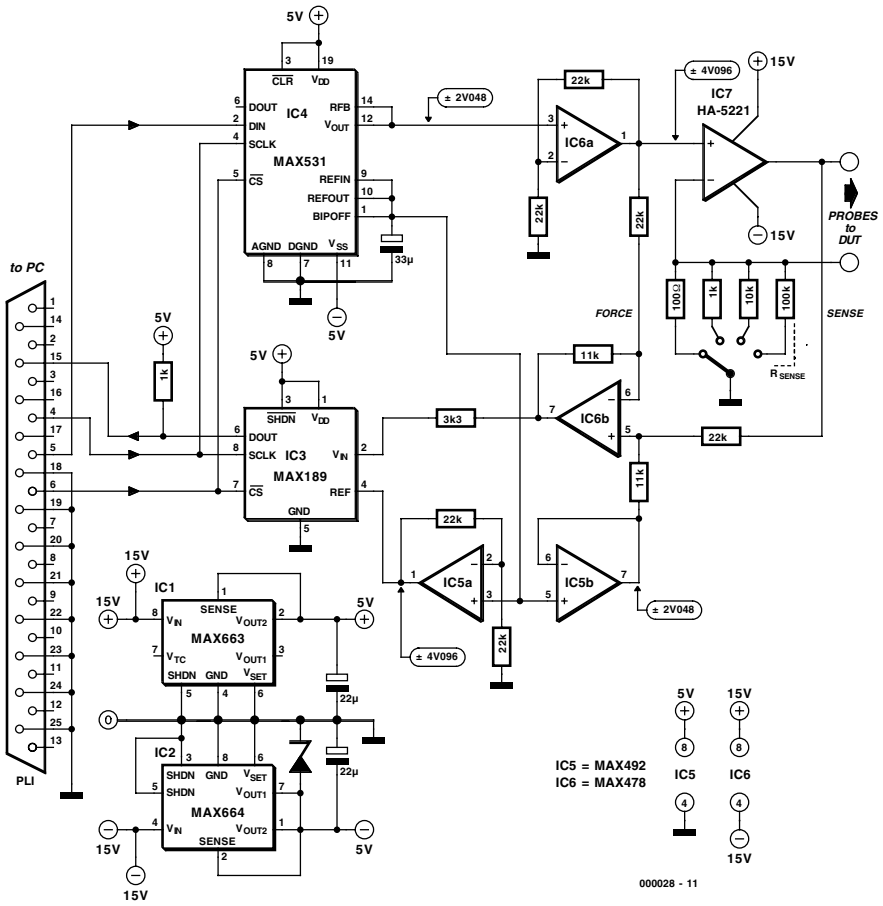
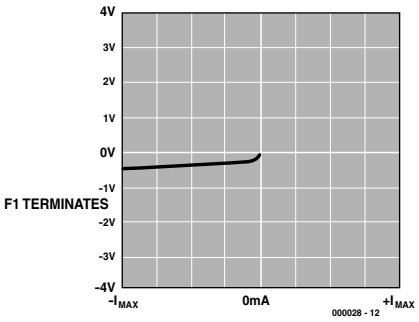


Figure 1. The main components of the circuit are a set of Maxim serial-interface A/D and D/A converters.

The hardware

Figure 1 shows the schematic diagram of the hardware. IC4 is a 12-bit D/A converter that is configured for a bipolar output voltage with a range of plus and minus 2.048 V. The opamp IC6a, which is connected to the V_{out} pin, amplifies this voltage by a factor of 2, so that a voltage ranging between plus and minus 4.096 V is present at the output of IC6a. IC7 in turn converts this voltage into a current that is proportional to the voltage. This is the test current for the device being measured.

The current ranges from ±40 µA to ±40 mA, depending on the value selected for the resistance of R_{SENSE} (100 Ω, 1 kΩ, 10 kΩ or 100 kΩ). The maximum output current is approximately equal to the value of the output voltage of IC6a (4.096 V) divided by the value of R_{SENSE}.



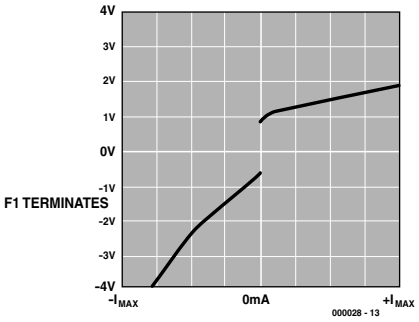


Figure 2. Two examples of measured I/U curves: (a) shows a Schottky diode, while (b) shows a more complex analogue IC.

The current through the DUT (device under test) produces a voltage drop across the component. This voltage is measured by the difference amplifier IC6b. In order to avoid an offset error that depends on the value of the selected current-sensing resistor, the signal for the inverting input of this opamp is taken from the low-impedance non-inverting input of IC7. The disadvantage of this is a fixed error equal to the input offset of IC7.
The amplification of the difference amplifier, in combination with the added offset, yields a maximum output swing of 0 to 4.096 V. This is exactly the right range for the unipolar input of the 12-bit A/D converter (IC3). The 3.3-kΩ resistor in series with the input of the ADC limits the input current in case the output voltage of IC6b should be greater than the supply voltage of IC3. When a measurement is being made, the software drives the DAC such that it produces a 'staircase' current. The resulting voltage across the DUT is measured by the ADC and then displayed on the PC monitor, with a resolution of

The BASIC program 'I-V Curve Tracer' was written by Terrry Millward of Maxim UK, and is available at the Maxim web site: http://www,maxim-ic.com/TechSupport/other.htm

```
'LISTING 1 - BASIC PROGRAM


'LPT1 OUT @ &H378, IN @ &H379
'LPT2 OUT @ &H278, IN @ &H279

'25 WAY D TYPE

'SIGNALS TO CIRCUIT
'PIN 4 D2 (OUT)   SCLK
'PIN 5 D3 (OUT)   DATA OUT (FROM PC)
'PIN 6 D4 (OUT)   CS\

'SIGNALS FROM CIRCUIT
'PIN 15 D3 (IN)  DATA IN (TO PC). ALSO SIGNALS EOC
'PINS 18-25 ARE GROUND

'INTRODUCTION

PRINT "WELCOME TO THE MAXIM CURRENT-VOLTAGE CURVE TRACER"
PRINT

INTRO:
PRINT "WHERE IS THE CIRCUIT CONNECTED? ENTER 1 FOR LPT1 OR 2 FOR LPT2"
INPUT P%
IF P% = 1 THEN PORT& = &H378                'SET LPT1
IF P% = 2 THEN PORT& = &H278                'SET LPT2
IF P% = 1 OR P% = 2 THEN GOTO INTRO1        'TRAP ERROR
PRINT "WRONG!!!!!! PLEASE TRY AGAIN"
GOTO INTRO
INTRO1:

ON KEY(1) GOSUB FINISH                      'F1 EXITS
KEY(1) ON

MAIN:                                       'MAIN BIT OF PROGRAM

'DEFINE SOME VARIABLES
DIM Y(512)                                  'ARRAY TO HOLD PLOT DATA
DIM DIN(12) AS INTEGER, DOUT(12) AS INTEGER 'DATA IN AND DATA OUT

'SET UP DISPLAY FOR OUTPUT
CLS 0
SCREEN 12                                   'VGA SCREEN
WINDOW (-120, -55)-(520, 435)               'DEFINE WINDOW CO-ORDINATES
LINE (0, -5)-(0, 410), 1                    'DRAW A FEW LINES
LINE (-5, 0)-(512, 0), 1
LINE (512, -5)-(512, 410), 1
LINE (-5, 410)-(512, 410), 1
LINE (-5, 205)-(512, 205), 1                'CENTRAL AXES
LINE (256, -5)-(256, 410), 1
LINE (-5, 51)-(0, 51), 1                    'MARKERS
LINE (-5, 102)-(0, 102), 1
LINE (-5, 153)-(0, 153), 1
LINE (-5, 256)-(0, 256), 1
LINE (-5, 308)-(0, 308), 1
LINE (-5, 359)-(0, 359), 1
LINE (64, -5)-(64, 0), 1
LINE (128, -5)-(128, 0), 1
LINE (192, -5)-(192, 0), 1
LINE (320, -5)-(320, 0), 1
LINE (384, -5)-(384, 0), 1
LINE (448, -5)-(448, 0), 1

COLOR 9                                     'LABELS
LOCATE 1, 1, 0: PRINT "MAXIM CURRENT-VOLTAGE CURVE TRACER"
LOCATE 20, 1, 0: PRINT "F1 TERMINATES"
LOCATE 2, 12, 0: PRINT "+4V"
LOCATE 15, 13, 0: PRINT "0V"
LOCATE 27, 12, 0: PRINT "-4V"
LOCATE 28, 15, 0: PRINT "-Imax"
LOCATE 28, 46, 0: PRINT "0mA"
LOCATE 28, 75, 0: PRINT "+Imax"

LOCATE 10, 1, 0
PRINT " Rs   Imax"
PRINT "100R  40mA"
PRINT "  1K   4mA"
PRINT " 10K 400uA"
PRINT "100K  40uA"

START:                                      'START OF PLOT ROUTINE
IDATA& = &H10                               'INITIALISE PORT, SCLK=0, CS\=1
OUT PORT&, IDATA&

Z& = 8                                      'INITIALISE MAX531
GOSUB IO
```

```
FOR X& = 1 TO 511 STEP 1                    'X& IS X AXIS FOR PLOT
 Z& = 8 * (X& + 1)                          'NEXT VALUE FOR MAX531
 GOSUB IO

 PSET (X&, Y(X&)), 0                        'RESET BACKGROUND
 IF Y(X&) = 0 THEN PSET (X&, Y(X&)), 1      'CATCH BOTTOM LINE AT START
 IF Y(X&) = 205 THEN PSET (X&, Y(X&)), 1    'CATCH HORIZ CENTRE LINE
 IF X& = 256 THEN PSET (X&, Y(X&)), 1       'CATCH VERT CENTRE LINE
 Y(X&) = INT(ODATA& / 10)
 IF Y(X&) > 408 THEN Y(X&) = 409            'CHECK FOR OVER-RANGES
 IF Y(X&) < 1 THEN Y(X&) = 1
 PSET (X&, Y(X&)), 4                        'SET NEW VALUE
NEXT X&
GOTO START                                  'AROUND AGAIN

IO:                                         'DATA I/O ROUTINE
ODATA& = &H0
IDATA& = IDATA& AND &HEF                    'CS\ SET TO 0
OUT PORT&, IDATA&
WAIT (PORT& + &H1), &H8                     'WAIT FOR EOC FROM CIRCUIT

                                            'SCLK CYCLE TO SHIFT 189 DATA TO
DOUT
IDATA& = IDATA& OR &H4                      'SCLK TO 1
OUT PORT&, IDATA&
IDATA& = IDATA& AND &HF3                    'SCLK TO 0
OUT PORT&, IDATA&

FOR SHIFT = 11 TO 0 STEP -1
 DOUT(SHIFT) = (INP(PORT& + &H1) AND &H8)   'GET DATA FROM MAX189
 ODATA& = ODATA& + DOUT(SHIFT) * 2 ^ (SHIFT - 3)'COMPILE DATA
 IDATA& = IDATA& OR ((Z& AND (2 ^ SHIFT)) / (2 ^ SHIFT) * 8)
                                            'DATA TO GO TO MAX531
 OUT PORT&, IDATA&
 IDATA& = IDATA& OR &H4                     'SCLK TO 1
 OUT PORT&, IDATA&
 IDATA& = IDATA& AND &HF3                 'DATA BIT BACK TO 0, SCLK TO
 OUT PORT&, IDATA&
NEXT SHIFT
IDATA& = IDATA& OR &H10                     'CS\ SET TO 1
OUT PORT&, IDATA&

  RETURN
  FINISH:
  END
```

640 x 480 pixels. The two examples shown in **Figure 2** illustrate some typical results. A resolution of 12 bits is actually somewhat luxurious for this application, but the software can be modified to work with higher resolutions, and you could even add a loop to the software for this.

Now we come to a few practical remarks regarding the power supply. IC7 and IC6 need a symmetrical supply voltage of ±15 V in order to operate over the desired range. All other ICs need only a ±5 V supply. The voltage regulators MAX663 and MAX664 (IC1 and IC2) provide the necessary voltages.

The QBASIC program shown in the accompanying listing starts by asking which printer port the circuit is connected to (1 or 2). After this, the graphic plot of the I/U curve of the device being tested appears on the screen. You can exit the program by pressing the function key F1. Make sure to run the program in true DOS mode, rather than in a DOS box under Windows. QBASIC can usually be found in a folder labelled 'OLDMSDOS' on the Windows CD, or else you should use an old version of MS-DOS.

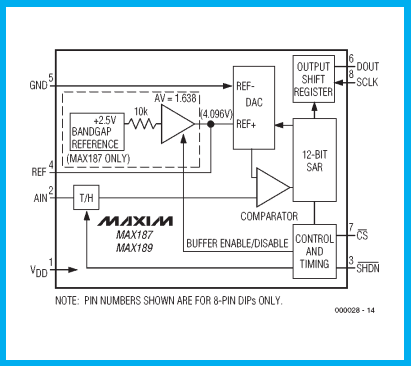
(000028-1)

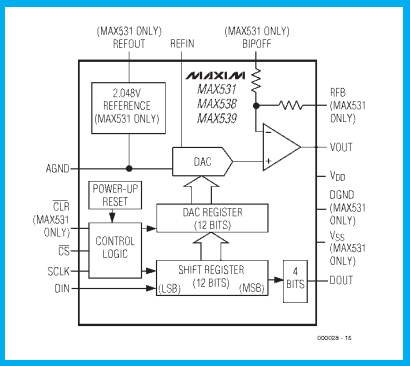
Source: Maxim Integrated Products

Serial-interface converters

The simplicity of the circuit described in this article is primarily due to the use of a set of serial-interface converters, namely the MAX 189 and MAX 1531.

The MAX 189 is a serial A/D converter with a resolution of 12 bits that works with a single +5 V supply. Its input voltage range also lies between 0 and +5 V. The core of this IC is a successive-approximation ADC with a conversion time of 8.5 µs. This is complemented by a fast sample and hold circuit (1.5 µs), an on-chip clock generator and a fast serial three-wire interface (see the block diagram in **Figure A**).





The conversion rate of the MAX 189 is 25 kilosamples/s. Thanks to its built-in interface, this IC can be easily connected to a PC or microprocessor, as can be seen in this application. In contrast to its companion MAX 187, the MAX 189 does not have an internal reference, but this is not necessary in this application since the DAC that is used provides the reference voltage. The very low operating power consumption (7.5 mW) makes this IC very suitable for battery-powered applications. In the shutdown mode, the power consumption drops to only 10 µW.

The MAX 189 is available in an 8-pin DIP package and a 16-pin SOI package.

The MAX 1531 can be seen as a complement to the MAX 189. Its block diagram is shown in **Figure B**. This 12-bit D/A converter also works with a single +5 V supply. The MAX 1531 was specifically chosen for this application, which requires a bipolar output voltage, since it can also work with a symmetrical ±5 V supply. The current consumption of the MAX 1531 is only 260 µA, including the internal 2.048 V reference source. The IC is housed in a 16-pin DIP or SOI package. The offset voltage, amplification and linearity are adjusted during manufacturing, so the user does not have to be concerned with them.

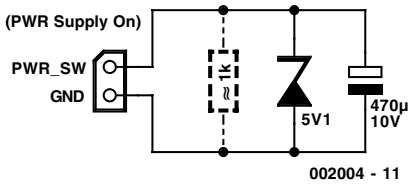
The internal output opamp of the MAX 1531 can be configured for an amplification factor of 1 or 2, and for a unipolar or bipolar output voltage. An internal shift register stores the serial data supplied to the IC.

If you install an AT motherboard with both ATX and AT power supply connections, in combination with an ATX power supply, in an AT enclosure, this circuit will save you from also having to install an additional pushbutton switch.

By R. Freitag

ATX power switch substitute

An additional pushbutton switch is normally required for the ATX Power Switch/Soft Power Switch signal, but you can do without it if you use this simple circuit. It is an artful design, but it has been repeatedly tested. The zener diode is intended to provide protection against excessive voltages and reverse-polarity connection. In the latter case, the resulting short-circuit current (approximately 1 A) will exceed the allowable limit and cause the ATX power supply to shut down after around five seconds. It might be possible to use a smaller capacitor; this must be tested experimentally in actual use. If the motherboard documentation is poor, you should verify the earth pin using a continuity tester.



The resistor is only needed if you want to be able to switch on the PC within ten seconds after switching it off. It discharges the capacitor quickly enough to make this possible. With a 1-kΩ resistor, the time constant is around 0.5 s. Since the capacitor also tends to stabilize the voltage, this circuit could also help in situations in which the ATX power supply switches off unintentionally due to voltage fluctuations on the PWR Supply On line.

(002004-1)

Construction guidelines

Elektor Electronics (Publishing) does not provide **parts and components other than** PCBs, fornt panel foils and software on diskette or IC (not necessarily for all projects). Components are usually available form a number of retailers – see the adverts in the magazine.

Large and small values of components are indicated by means of one of the following prefixes :

E (exa) = 10^{18}	a (atto) = 10^{-18}
P (peta) = 10^{15}	f (femto) = 10^{-15}
T (tera) = 10^{12}	p (pico) = 10^{-12}
G (giga) = 10^{9}	n (nano) = 10^{-9}
M (mega) = 10^{6}	μ (micro) = 10^{-6}
k (kilo) = 10^{3}	m (milli) = 10^{-3}
h (hecto) = 10^{2}	c (centi) = 10^{-2}
da (deca) = 10^{1}	d (deci) = 10^{-1}

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,

3k9 = 3.9 kΩ 4µ7 = 4.7 µF

Unless otherwise indicated, the tolerance of resistors is ±5% and their rating is ⅓–½ watt. The working voltage of capacitors is ≥ 50 V.

In **populating a PCB**, always start with the smallest passive components, that is, wire bridges, resistors and small capacitors; and then IC sockets, relays, electrolytic and other large capacitors, and connectors. Vulnerable semiconductors and ICs should be done last.

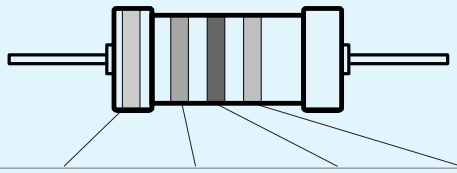
Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICs and semiconductors. Unsoldering is best done with a suction iron or special unsoldering braid.

Faultfinding. If the circuit does not work, carefully compare the populated board with the published component layout and parts list. Are all the components in the correct position? Has correct polarity been observed? Have the powerlines been reversed? Are all solder joints sound? Have any wire bridges been forgotten?

If voltage levels have been given on the circuit diagram, do those measured on the board match them – note that deviations up to ±10% from the specified values are acceptable.

Possible corrections to published projects are published from time to time in this magazine. Also, the readers letters column often contains useful comments/additions to the published projects.

The value of a resistor is indicated by a **colour code** as follows.



color	1st digit	2nd digit	mult. factor	tolerance
black	–	0	–	–
brown	1	1	$\times 10^{1}$	±1%
red	2	2	$\times 10^{2}$	±2%
orange	3	3	$\times 10^{3}$	–
yellow	4	4	$\times 10^{4}$	–
green	5	5	$\times 10^{5}$	±0,5%
blue	6	6	$\times 10^{6}$	–
violet	7	7	–	–
grey	8	8	–	–
white	9	9	–	–
gold	–	–	$\times 10^{-1}$	±5%
silver	–	–	$\times 10^{-2}$	±10%
none	–	–	–	±20%

Examples:
brown-red-brown-gold = 120 Ω, 5%
yellow-violet-orange-gold = 47 kΩ, 5%

Java is a world language. Many people think that this programming language, which was developed by Sun Microsystems, is only suitable for embellishing Internet pages with cute applets. Nothing could be further from the truth. Java is a fully-fledged programming language that can be used for making Internet applets and stand-alone applications for all possible platforms. What about a Java application in a ring, a credit card or your toaster? It's all possible!

By G. Polder

Java MultiMeter (JMM)
handy software for RS232 multimeters

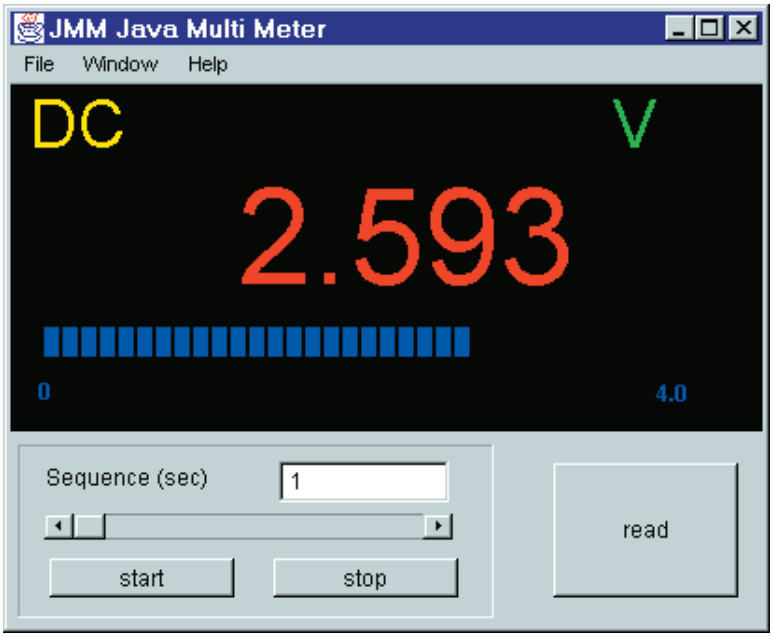


Figure 1. The main window of Java MultiMeter shows the settings of the multimeter connected to the PC, and the measured value in the form of a number and a bar graph. The buttons can be used to manually make a new measurement or enable the program to automatically read in measurements at a regular interval.

In this project, Java is used for reading data from a multimeter that has an RS232 port (such as the Voltcraft model 3850d). For more background information on this type of multimeter, see the April 1997 issue of *Elektor Electronics*.
Java MultiMeter (JMM) has a very simple design. The main window displays the measurement value, the range and a bargraph (a semi-analogue measurement bar), plus a number of buttons. The READ button at the lower right is used to read in a value, while the START and STOP buttons can be used to control repetitive measurements at an adjustable interval that can be set using a slider (Sequence).
Up to now, this is nothing unusual. However, there is more. At the top of the window there is a menu bar that has two options in the Window menu, namely 'Plot' and 'Log'. If you click on 'Log', a logging window appears that displays all measured values and the time of each measurement. This can be useful for looking at what happened in the past. In addition, the information in the log window can be stored for subsequent processing, for example using a spreadsheet program.
If you click on 'Plot', a plot window appears that displays the measured values in graphic form. This plotting function works in a fairly advanced manner. The scale is automatically adjusted after each new measurement point is added. You can use the mouse to draw a rectangle over part of the plot, and then zoom in by moving the mouse down and to the right, or zoom out by moving the mouse up and to the left. If you press the 'Fill' button, the full set of data is displayed again. Several plot windows can be open at the same time. New measurement data are always entered in the most recently opened window.

Operation

Java is an object-oriented language. This means that the programmer makes objects that communicate with each other and that are 'semi-concurrently' active. The most important objects in JMM are the following:

- the main object, which contains the measurement data, the buttons and a timer that periodically sends a 'D' to the RS232 port if it is enabled;
- an RS232 receiver object, which collects RS232 data and sends the entire string to the main object after it receives a <CR>(Carriage Return) sequence;
- a log object, which receives measurement strings from the main

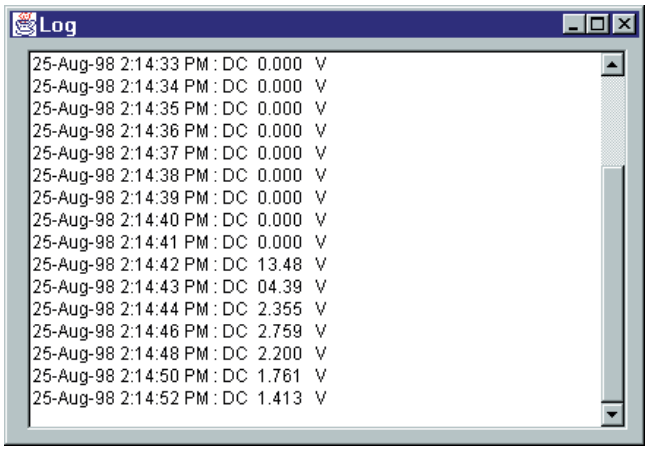


Figure 2. The log window displays a summary of all measurements, with their associated dates and times.

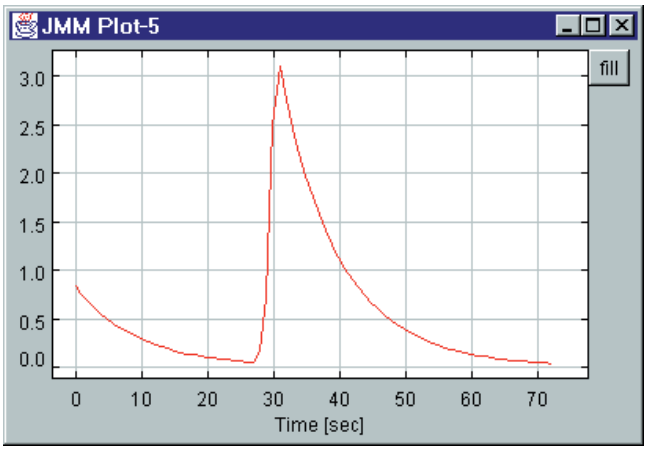


Figure 3. The plot module shows a graphic diagram of the most recently measured values. Several plot windows can be open at the same time.

object, attaches date and time information to them and organises them into a list;
- plot objects, among which the most recent object graphically displays the data that are received from the main object.

Required hardware

The demands placed on the computer are not particularly high. It must of course be a PC that runs under Windows 95/98 or Windows NT.
In addition, a digital multimeter with an RS232 port (such as the Voltcraft model 3850d) is needed to provide the measurements to the computer via a serial channel. Finally, a suitable RS232 cable is needed to interconnect the meter and the computer.

The software

The software was produced using the following software packages:

- *Symantic Visual Cafe:* this is a powerful Java programming environment;
- *Solutions Consulting serial port software:* these are Java classes (routines) and Windows DLLs that allow Java to make use of the RS232 ports of the computer;
- *PT-Plot:* a Java plotting package.

One of the features of Java is that it is platform-independent. This is achieved by running Java programs on what is called a Java Virtual Machine. With Visual Cafe, it is possible to combine the Java programs with the Virtual Machine into a single .EXE file. JMM utilises this capability. This however means that a number of Semantic DLLs must be installed. An installation program for these DLLs, which also includes the remaining components of JMM, may be found on the Elektor CD-ROM *PC Software 98-99* (ISBN 90-5381-106-0).

(002010-1)

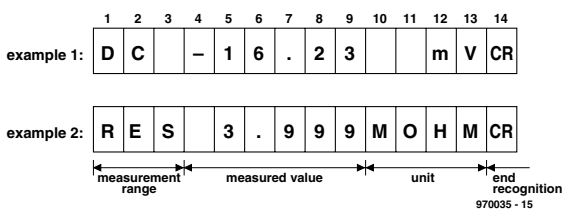
The program JMM won the third prize in the Netherlands division of the PC software competition organised by Elektor in June 1998. All winning entries for this competition are collected on the above-mentioned CD-ROM.

Required files

jmm.exe	The Java MultiMeter program.
csall	This folder contains a number of *Solutions Consulting* serial port DLLs. Place these DLLs in the folder c:\windows\system32 (for Windows 95/98) or c:\winnt\system32 (for Windows NT).
snjrt20.exe	Installation program for *Semantic Java* runtime DLLs.
libwin95	Property files for Windows 95/98. Copy these files to the folder c:\windows\system\lib.
libwinnt	Property files for Windows NT. Copy these files to the folder c:\winnt\system32\lib.

The serial protocol

Programming the communications between the PC and a digital multimeter with a serial interface is fairly easy if you use a high-level programming language such as Visual BASIC, Visual C (*Java*) or Visual Pascal (*Delphi*).
To start with, the serial interface that is used must be initialised. For most DMMs, it should be configured to 1200 bps, no parity bit, 7 data bits in ASCII format, 2 stop bits and software handshaking. This completes the preparations for transferring measurement values. All that the computer or program has to do now is to request a measurement value. This is done by sending the character 'D' (#68) to the DMM. In response, the PC receives a string of characters that contain the current measurement value, the unit and the measurement range. The received string is 14 bytes long, as shown in the figure below. The DMM models 506 and M-3860M are exceptions to this rule. The latter model works at a data rate of 9600 baud and always sends the measurement values for the main display and the three auxiliary displays as a single data packet that is 56 bytes long. The model 506 works at 1200 baud, but it sends a data packet whose length varies between 6 and 15 bytes. All DMMS are rather slow in operation; new measurements are sent to the PC no more often than once every 0.5 second.



Microprocessors are regular devils for work. In many cases, this involves gathering data, performing calculations and finally making the results visible. For this last, often very important task, a display is necessary. The circuit presented here shows that only a few components are needed to make an inexpensive display that has high contrast and is thus easy to read.

Design by I. Gerlach

versatile LED display

for use with many microprocessor systems

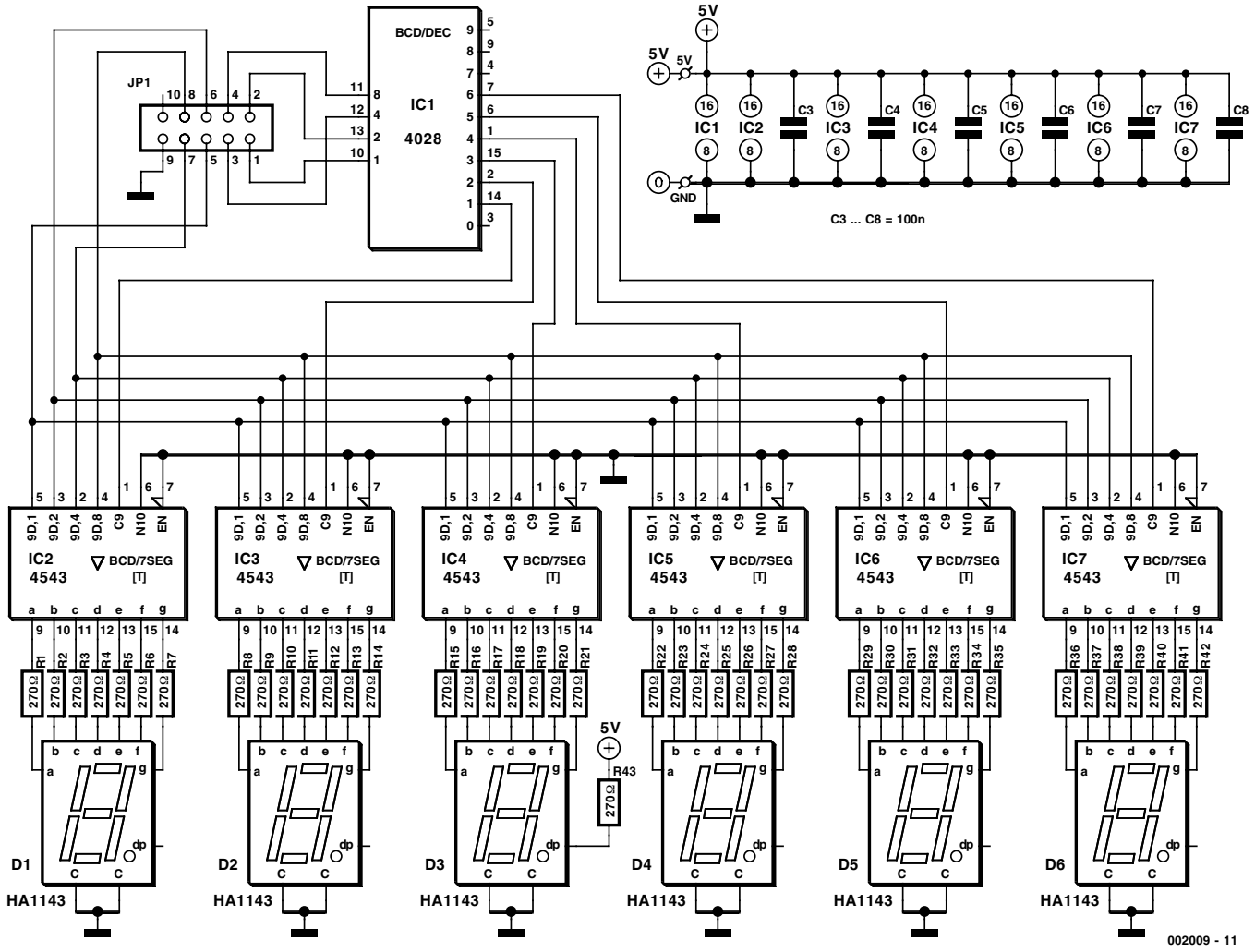


Figure 1. The schematic diagram of the LED display. Only standard components are used. The design of this display makes it very flexible, without imposing a heavy load on the processor.

Most microprocessor systems are difficult to use without a good display. Microprocessors are after all black boxes that can quickly perform a whole lot of computations, totally hidden from the outside world. Since computation is not an end in itself, and the user wants to see clear conclusions, many projects need displays. Liquid crystal (LC) displays are used more and more often for this purpose, for evident reasons:

- both letters and numbers can be displayed on a LC display,
- a built-in LCD controller makes the connection simple and minimises the load on the processor,
- LC displays use little energy,
- the displays are compact.

In addition to their advantages, LC displays naturally have disadvantages:

- they emit no light and thus need a backlight under conditions of poor illumination,
- they have a limited viewing angle,
- they can only be used over a limited temperature range,
- they have relatively low contrast,
- they are relatively expensive,
- different software may be required, depending on the type of display used,
- the dimensions of the display are limited,
- the useful lifetime is limited.

In summary, an LC display is sometimes a good choice, but in other cases an LED display may be a better option. In this article, we show that a six-character LED display can be built at a relatively low cost. There were several primary considerations in the development of this idea. In particular, the display must:

- use relatively few I/O lines,
- be inexpensive to build,
- require relatively little computing time from the processor,
- be flexible and thus extensible,
- be usable with practically every type of microprocessor.

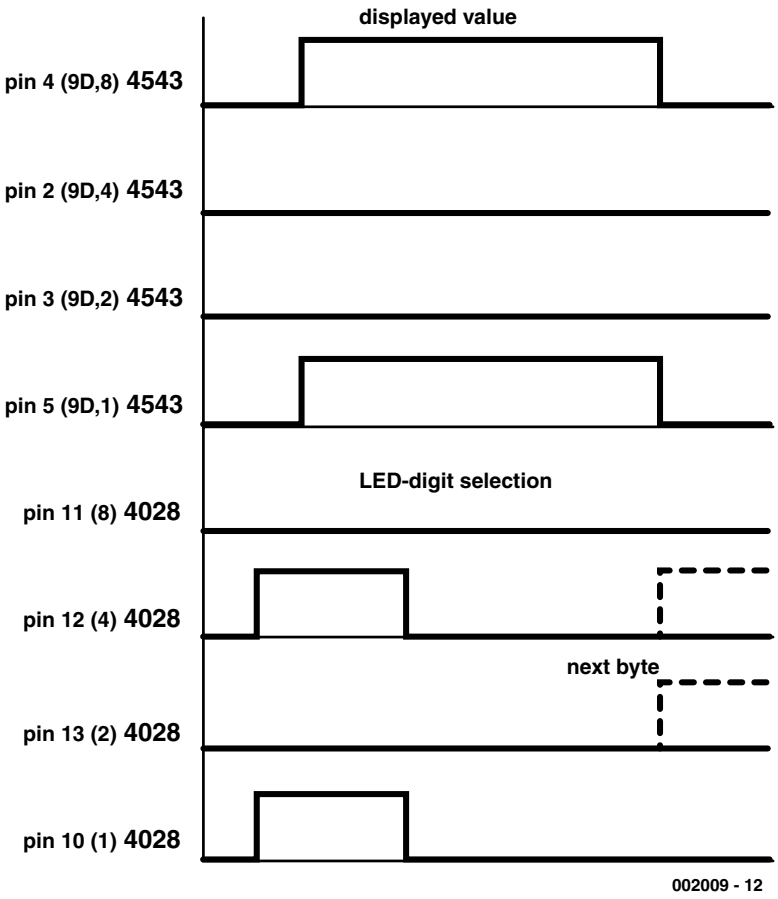


Figure 2. Timing diagram of the drive signals. The upper four signals determine the code for the display (9 in this case), while the lower four signals select the display IC (5 in this case).

The hardware

Figure 1 shows the schematic diagram of the display circuit. Only two types of IC are used in this project, both of which are inexpensively available. IC1 is a 4028, which is a simple demultiplexer. The remaining six ICs are 4543 types. This is a BCD decoder with a built-in latch and display driver.
The display components are seven-segment display ICs with common cathodes. This means that the CC pins are connected to earth. The remaining pins (a, b, … g) are directly connected to the corresponding outputs of their associated display controllers (IC2 through IC7). Finally, each display IC has one other pin, labelled 'dp'. This pin drives the decimal point of the display.
In this design, only the decimal point of the third display IC is activated, by connecting it to +5 V via a 270 Ω resistor (R43). If you want to have the decimal point displayed at some other location, it is easy to change this arrangement.
The entire display is connected to the microprocessor system via a header with two rows of five pins. Only nine of the ten pins are actually used. Since one of the pins is connected to earth, only eight processor I/O lines are needed to drive the display.
A 4-bit signal applied to pins 1 through 4 selects the desired display IC. The bit pattern on these four lines determines which output of IC1 is high. The pattern '0000' is not used, since no display IC is connected to

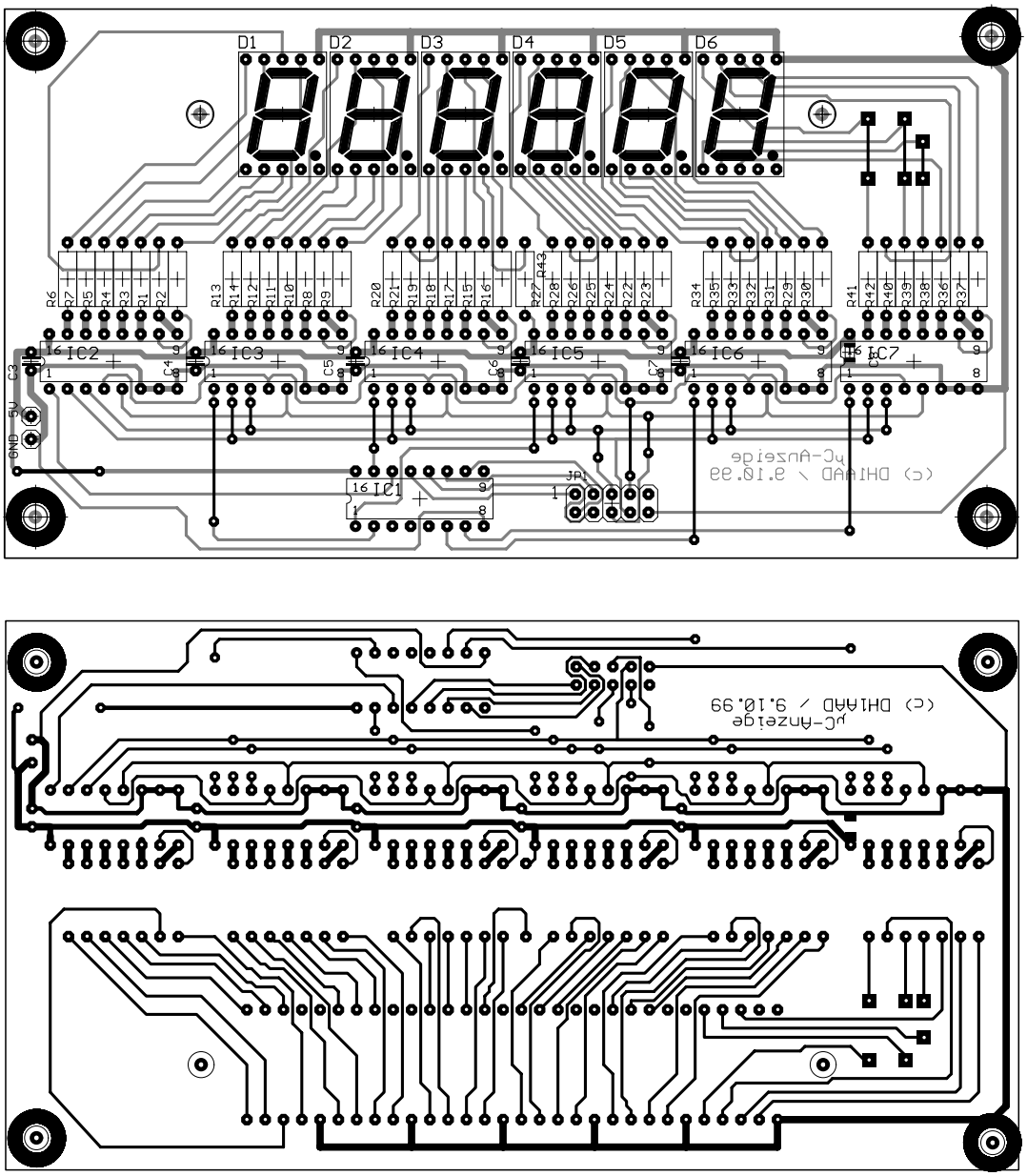


Figure 3. The copper track layout and component layout of a printed circuit board that can be used to build the LED display.

pin 3 (Q0) of IC1. The other four signal lines (pins 5 through 8) are used to generate the BCD codes that drive the display. As soon as the proper BCD code has been applied to these lines, the load input (LD, pin 1) of the controller for the desired display IC is set high. On the falling edge (from high to low) of this signal, the 4543 stores the code present on its inputs 1A through 1D (pins 2 through 5). The pattern of output pins (AS through G) that go high for a particular input code is determined by a matrix integrated into the decoder IC. The associated segments of the display IC are then illuminated.

The whole process is illustrated in the timing diagram shown in **Figure 2**. The four upper bits show that a BCD code of '1001' is applied to the inputs of the display controllers. After this, IC5 is activated by the 4028 decoding the selection code "0101' (decimal 5).

The approach taken here has two important advantages for the user. First, as long as the displayed value does not have to be changed, no new control signals have to be sent the display. Second, it is possible to modify the value displayed by a single display IC. These two features mean that the load on the processor is kept to a minimum.

There is little to say about the rest of the circuit. A simple 5-volt power supply is used. Each IC has a decoupling capacitor. The series resistors for the display segments are well chosen for the supply voltage used. If you want to make the display brighter, the value of these resistors (R1 through R43) can be reduced, for example to 180 Ω. If on the other hand you want less brightness, in the interest of reduced energy consumption, you can increase the value of these resistors.

Printed circuit board

In order to simplify the construction of the display, the author has designed a printed circuit board that is shown in **Figure 3**. This board is not available from Readers Services, so you will have to make it yourself. This should not be difficult, since the layout is simple and

the board is single-sided.
When assembling the board, start with the wire bridges. After this you can insert the remaining components. You should preferably use IC sockets, and solder carefully. For IC2 through IC7, the last resistor position always has no component number. This is the resistor for driving the decimal point (R43 in the schematic diagram). First decide if you want to have a decimal point, and then decide where it should be located. After this, all you have to do to activate the decimal point is to solder a 270-Ω series resistor in the proper location.

Sample software

In order to use the display with a microcontroller, you will naturally need some software. Consequently, we have included a sample program in the form of a machine-language routine that is suitable for Atmel AT90S1200 through AT90S8515 processors. We have chosen to use a separate driver to keep the application as simple as possible. This makes a flexible approach possible, and this driver can be called from an application. This means that the control software only has to be written once, after which it can be used in a variety of systems. **Listing 1** shows the design of the driver. Its has a quite straightforward construction, which

```
;*************************************************
;* File Name    :LEDDisp.inc
;* Title              :Driver for LED Display
;* Date         :Ingo Gerlach / 10.10.99
;* Version            :1.0 / 11.10.99
;* Version            :
;* µC                 :AT90S1200...8515
;*                    :
;* Changes            :
;*
;*************************************************
;
;
; Main program register variables
;-----------------------------------------
;.def   temp    = r16
; Registers  / LED
;-----------------------------------------
;.def   cntr    = r20   ; counter
;.def   dly     = r21   ; delay loop variable
;.def   pos     = r23   ; position
;.def   byte    = r24   ; byte

; Equates
;-----------------------------------------
;.equ   LED_qty = 6             ; number of LEDs
;.equ   LED_Del = 45            ; delay
;.equ   OutPort = PortB

; Functions
; LED_Blank     : switch display on
; LED_Null      : reset display 0 ( Null)
; LED_Show      : show bytes , transport byte (R24),
position (R23)

; **** Switch display off
;*************************
LED_Blank:      ldi     cntr,LED_QTY
LedLoop1:       ldi     temp,192
                add     temp,cntr
                out     OutPort,temp
                Rcall   Led_Delay
                dec     cntr
                brne    LedLoop1
                Ret

;*************************
; **** Reset display
;*************************
LED_Null: ldi   cntr,LED_QTY    ; load number of LEDs
LedLoop2: out   OutPort,cntr
                Rcall   Led_Delay
                dec     cntr
                brne    LedLoop2
                Out     OutPort,cntr
                Ret
;*************************
; **** Show byte
;*************************
LED_Show:   mov   temp,pos ; position in register
                out       OutPort,temp; activate BCD-
to-decimal decoder, LD 4543
                Rcall     Led_Delay ; short delay
                mov       temp,byte ; value in
                                      register
                swap      temp      ; value high
                                      nibble
                add       temp,pos  ; goto position
                out       OutPort,temp; 4028 / 4543
                          move value to display
                Rcall     Led_Delay ; short delay
                sub       temp,pos
                out       OutPort,temp; LD signal
off, store value
                Rcall     Led_Delay
                Ret
;*************************
; * Internal functions !!!

; *** Delay LED display
;**********************
LED_Delay:      ldi dly,LED_Del
LedLoop:        dec dly
                brne LEDLoop
                ret
```

Listing 1. A machine-language driver for the LED display. This can be integrated into various applications.

makes its operation easy to follow.
Listing 2 contains a short demonstration program. This program first sets the display to all zeros and then displays the number '145675'. If the decimal point is set after the third segment, the value to be read from the display is thus 145.675.
You can easily experiment with this flexible LED display. If the code has to be translated for a different processor family, this can be easily done based on the information provided in the listings.

(002009-1)

Text (Dutch original): H. Steeman

```
;***************************************************
;* File Name    :LED.asm
;* Title              :Test program for LED display
;* Date         :Ingo Gerlach / 10.10.99
;* Version            :1.0 / 10.10.99
;* Version            :
;* µC                 :AT90S1200...8515
;*                    :
;* Changes :
;*
;***************************************************
;

;***** Directives


.device AT90S1200              ;device type
.NOLIST

.include "1200def.inc"


.list
.listmac



; Show data
; Structure of data

;       MSB                LSB
;       7 6 5 4        3 2 1 0
;       0 0 0 0        0 0 0 0
;
;       Select position 1 = 1. 2 = 2. etc
;       Number in BCD code
;
;e.g.   10010010b = 146d = pos. 2, value 9


;
; Main program register variables
;-----------------------------------------
.def    temp           = r16


; Registers  / LED
;-----------------------------------------
.def    cntr           = r20   ; counter
.def    dly            = r21   ; delay loop variable
.def    pos            = r23   ; position
.def    byte           = r24   ; byte


; Equates
;-----------------------------------------
.equ    LED_qty = 6            ; number of LEDs
.equ    LED_Del = 40           ; delay
.equ    OutPort = PortB


;***** Interrupt vector table

reset:
    rjmp  main  ; main routine
    reti        ; external interrupt0 handle
    reti        ; T/C0 overflow interrupt handle
    reti        ; analogue comparator interrupt handle

;***** Functions
***************************************************


;****************************************
;***** Main ******************************

main:
;   ldi   temp, LOW(RAMEND)    ; setup StackPointer
                                 for > 90S1200
;   out   SPL, temp            ; initialize SPL
;   ldi   temp, HIGH(RAMEND)
;   out   SPH, temp            ; initialize SPH


    ldi   temp,255             ; temp = 255
    out   ddrb,temp            ; port B output

    Rcall LED_Null             ; reset display

mainloop:

; Show  145.675

                ldi     Pos,1
                ldi     Byte,1
                Rcall   Led_Show

                ldi     Pos,2
                ldi     Byte,4
                Rcall   Led_Show


                ldi     Pos,3
                ldi     Byte,5
                Rcall   Led_Show

                ldi     Pos,4
                ldi     Byte,6
                Rcall   Led_Show

                ldi     Pos,5
                ldi     Byte,7
                Rcall   Led_Show

                ldi     Pos,6
                ldi     Byte,5
                Rcall   Led_Show


forever:        rjmp forever

; ***** End of main program ******************

; ***   Include Files  ***
.include "LEDDisp.inc"
```

Listing 2. A demonstration program that shows how the driver can be integrated with an application.