ELEKTOR ELECTRONICS

THE ELECTRONICS & COMPUTER MAGAZINE

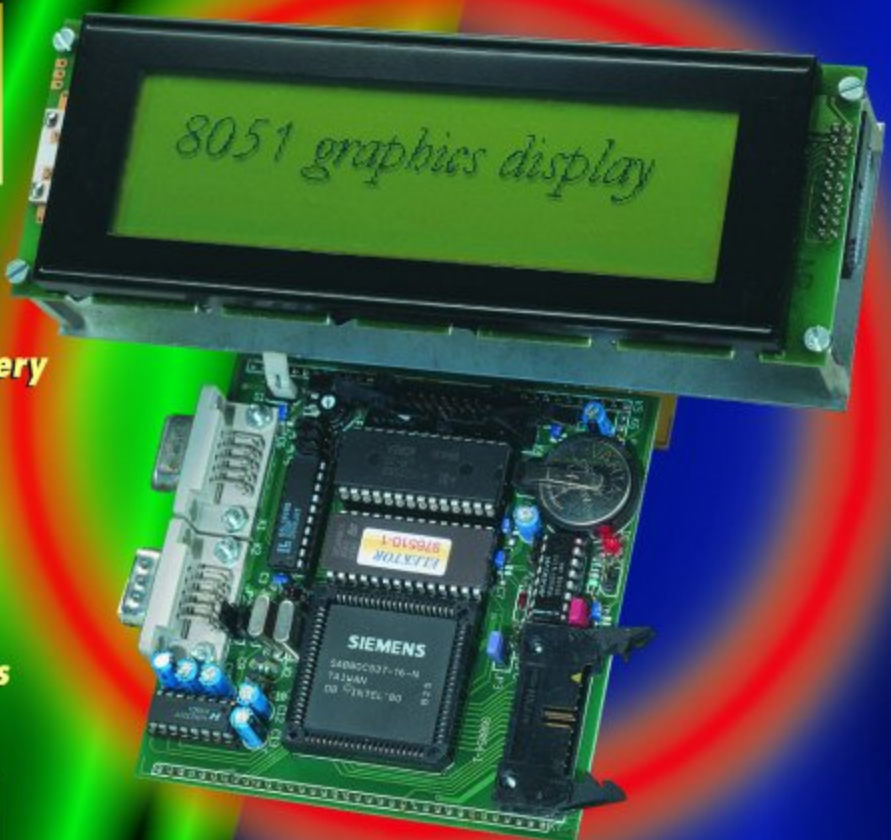
SEPTEMBER 2001
£2.95

www.elektor-electronics.co.uk

Personal Mini Webserver

Graphic LCD Module for 8051 Micros

Atmel AVR Micro Programmer



- Lead-Acid Battery Revitaliser
- MIDI-DMX Converter
- Video-SVHS Converter
- Servo on I^2C
- Li-ion Batteries

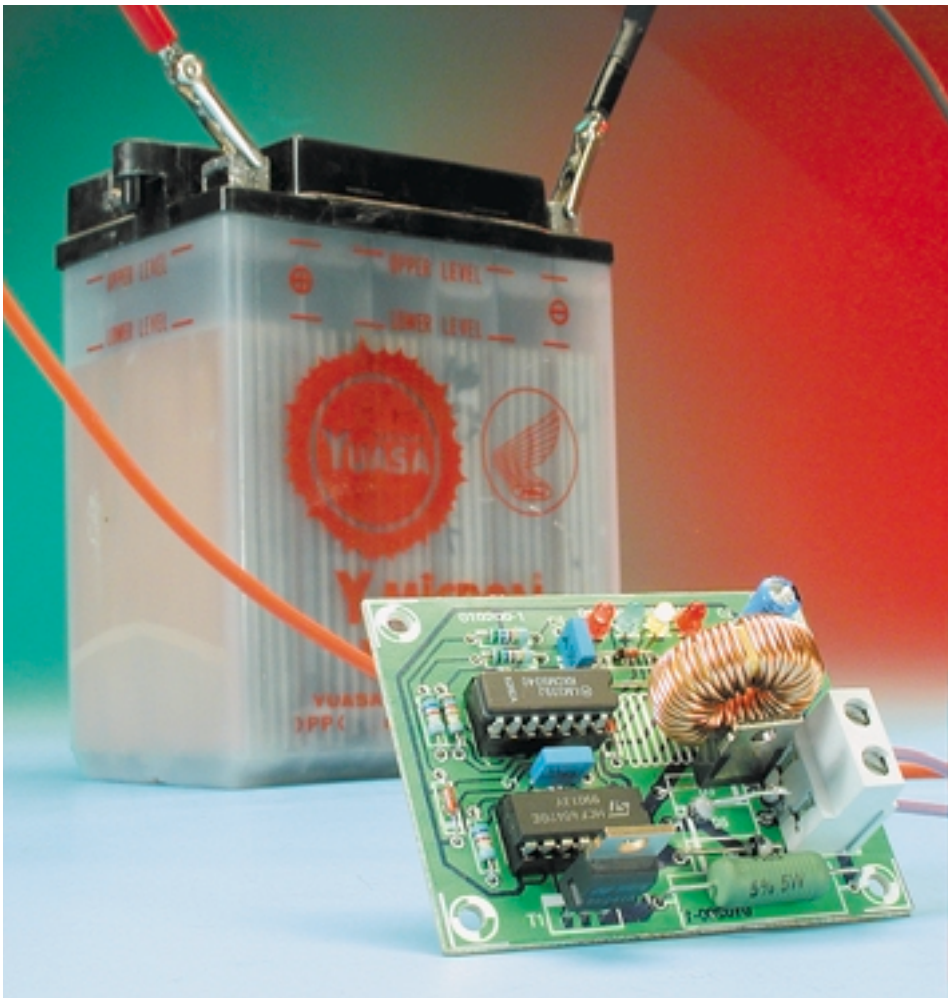
09
9 770268 451968

Lead-Acid Battery Revitaliser

a de-sulphation device for worn out batteries

Design by K. Walraven

This circuit makes it possible to do something that was previously unthinkable: reversing the effects of sulphation in tired lead-acid batteries. The circuit is also recommended for use as a conditioner for new batteries.



In a recent scientific journal it was stated that 80% of lead-acid batteries eventually fail due to sulphation. This sulphation occurs due to general old age, non-ideal charge/discharge cycles or storage in a discharged state for too long. The last happens often with batteries in motorbikes and classic cars, since these have an enforced rest during the winter months. To counter this, *Elektor Electronics* has designed a special charger a couple of years ago, which keeps the batteries in good condition during their winter sleep.

What exactly is 'sulphation'? This is a condition when the lead sulphate that is formed on the plates of a battery during its discharge changes in structure. Fairly large sulphate crystals are formed, which block the pores of the lead plates and hence reduce their active area. This causes the capacity of the battery to be reduced — it can no longer supply large currents and it can not be charged effectively in the usual way. When such a sulphated battery was charged, conducting bridges (short circuits) appeared between the plates, of which it was thought up to now that they could not be

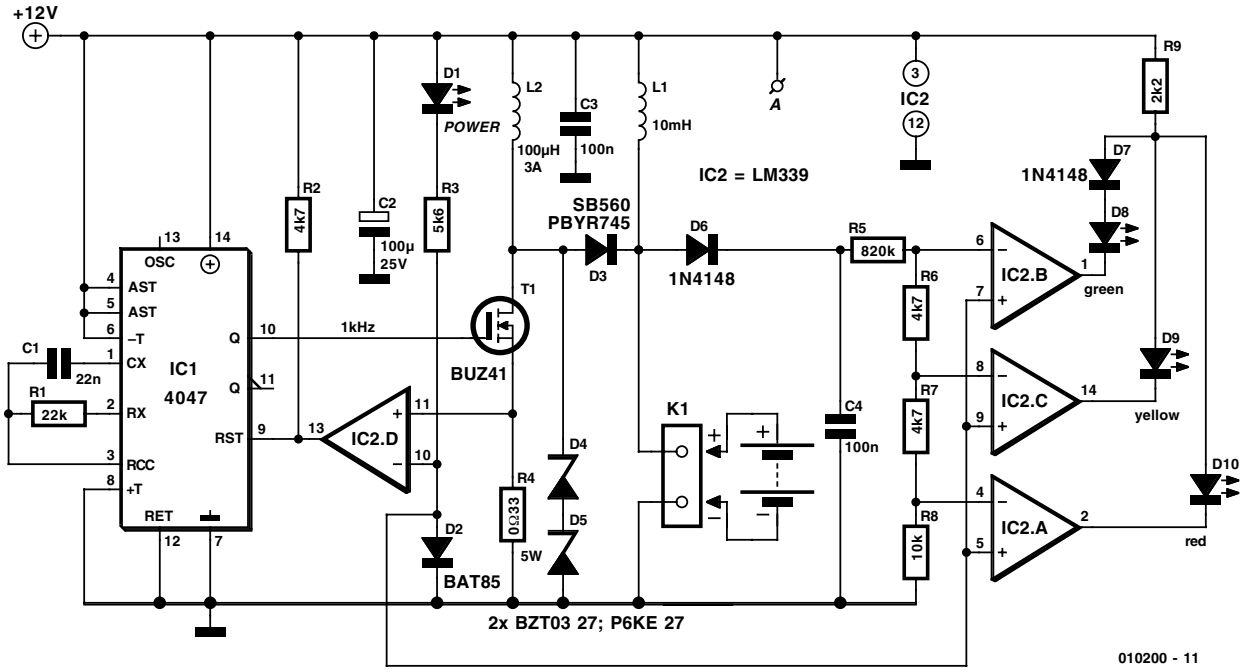


Figure 1. The circuit consists of a pulse generator and an indicator section.

removed. For the battery concerned it meant the end of its service life.

Known tricks

In the true spirit of electronics enthusiasts, you will not immediately take a tired battery to the recycling depot. After all, they aren't cheap and it is worth it to check that it really is at the end of its tether.

Insiders will undoubtedly know a few tricks that pep up a tired battery. One of the best known is to charge and discharge the battery repeatedly. This method causes a large part of the lost capacity to be restored for some reason or other. In other cases, applying large current pulses periodically seems to have some effect.

But both these method leave something to be desired in cases where badly sulphated batteries have to be brought back to life.

Cure

In recent years several manufacturers have been developing methods for the reversal of sulphation in lead-acid batteries, with varying success. The working methods seem to rely on some kind of pulsed charging. This is in contrast to normal charging procedures, which mostly use a constant voltage.

The design described here represents the latest techniques for revitalising lead-acid batteries. It is a device that periodically feeds short but fierce pulses to the battery, while discharging it slightly in between the pulses. This is, as far as known at the moment, the most effective way to break up unwanted deposits of sulphate crystals and to restore the battery plates to a reasonable condition.

Since the energy required for the charging pulses is derived from the battery itself (that may seem a bit weird at first, but the discharge of the battery is also part of this process), it is recommended to connect a charger in parallel with the battery and revitaliser when the battery has very little capacity left – but we'll go into detail of that later.

We have to be honest here and admit that our own experience of the circuit is not enough to give it an unconditional guarantee of successful operation. But since the circuit is not really expensive, its use deserves the benefit of the doubt.

Pulse generator

It can be seen from the circuit diagram in **Figure 1** that the electronics required for the revitaliser are very modest. The circuit contains two parts: a generator built round IC1, IC2d and T1, which creates the charging pulses, and an indicator circuit consisting of little more than three opamps (IC2a, b, c) and three LEDs, which show what state the battery is in.

Let's look at the pulse generator first. Just as the rest of the circuit, its supply is taken from the battery via K1. While we're talking about the supply, it should have a fairly constant voltage and be free from spikes (apart from the ones generated by the circuit itself). Suppression inductor L1 has been added to remove unwanted spikes, with C2 and C3 acting as reservoir capacitors. LED D1 lights up when the supply voltage is present.

To continue with the pulse generator, IC1 (a 4047) creates a square wave with a frequency of 1 kHz and a duty cycle that normally is 50%. As soon as the Q output of IC1 becomes high, FET T1 will turn on. This causes a (discharge) current to flow from the battery through L2, which increases linearly until the voltage across R4 is about 0.35 V; the current is then about 1 A.

At that moment comparator IC2d will switch over, causing IC1 to be reset and T1 to be turned off. The stored magnetic energy in L2 is now converted into a voltage spike,

COMPONENTS LIST

Resistors:
R1 = 22kΩ
R2,R6,R7 = 4kΩ7
R3 = 5kΩ6
R4 = 0Ω33 5W
R5 = 820kΩ
R8 = 10kΩ
R9 = 2kΩ2

Capacitors:
C1 = 22nF
C2 = 100µF 25V radial
C3,C4 = 100nF

Inductors:
L1 = 10mH
L2 = 100µH 3A suppressor coil

Semiconductors:
D1 = LED
D2 = BAT85
D3 = SB560 or PBYR745
D4,D5 = BZT03 27 or P6KE 27
D6,D7 = 1N4148
D8 = LED, green (high efficiency)
D9 = LED, yellow (high efficiency)
D10 = LED, red (high efficiency)
T1 = BUZ41
IC1 = 4047
IC2 = LM339

Miscellaneous:
K1 = 2-way PCB terminal block, lead pitch 7.5mm
PCB, order code **010200-1** (see Readers Services or Elektor Electronics website)

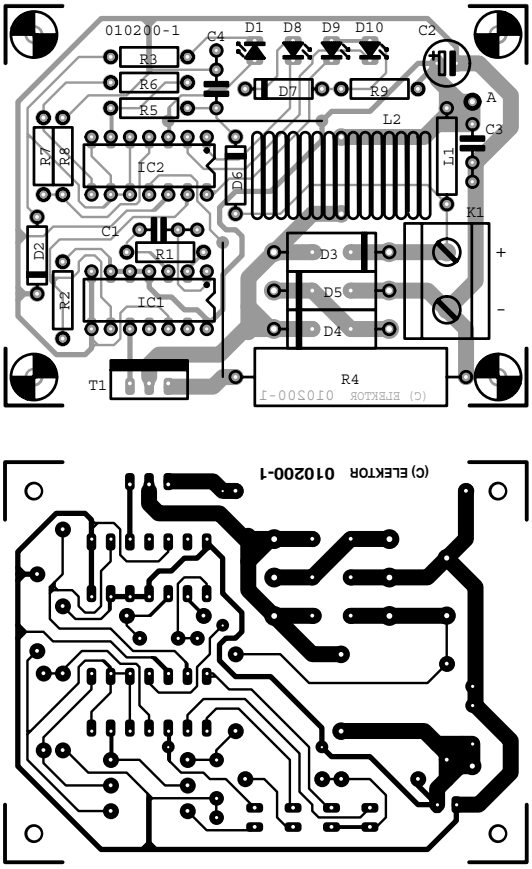


Figure 2. The use of this printed circuit board makes construction easy.

which is fed to the battery via D3.

The size of the spike is dependent on the state of the battery. When the battery is in a reasonable condition and its internal resistance is fairly low, then the peak of the spike will also be low (less than 15 V). With a high internal resistance the peak can be as big as 50 V. Its maximum value is limited by the two series connected zener diodes, D4 and D5.

Indicators

Since the condition of the battery can be determined by the size of the charging pulses, we've added a simple circuit that indicates the peak value of the pulses. The three comparators IC2a-c measure the peak value stored in C4 and switch over at voltages of 15, 20 and 30 V respectively. So when the battery is in a fairly good condition, the green LED (D8) lights up, with a mediocre battery the yellow LED (D9) and with a really poor battery the red LED (D10).

There is a detail that should be mentioned about the indicator circuit: to avoid all three LEDs from lighting up at the same time when the peak voltage is very high, they have been connected in parallel to one common series resistor (R9). Because the red LED has a lower voltage drop than the yellow LED, they will never light up at the same time. Since the yellow and green LEDs have a similar voltage drop, the same trick won't work here, which is why the green LED has an ordinary diode (D7) connected in series with it.

Construction

A compact printed circuit board (**Figure 2**) has been designed for this circuit, making the construction of this clever revitaliser a simple task even for the less experienced hobbyists. It's just a matter following the component layout and parts list carefully before construction. The battery is connected to terminal block K1 (observe the polarity!). Don't forget the wire links though; there are only two in this case, but without them the circuit will not work at all!

Because the charging pulses can cause high frequency interference, the completed printed circuit board should be mounted in a metal case.

The choice of components is not very critical. Any small Schottky diode can be used for D2. For D3 any fast power Schottky diode rated for at least 60 V/3 A is suitable.

The choice of T1 is also fairly wide, because in practice any power FET is suitable which is rated for 3 A and 100 V. The well-known BUZ10 would even be suitable, but then the zener voltage would have to be reduced to 27 V by replacing one of the zeners (D4 or D5) with a wire link. One important thing about these zeners: they can't be normal ones, but should be fast types. The zener voltage isn't critical as such, but the basic assumption is that the total voltage of zeners D4/D5 should be in the region of 40 to 50 V. In any

case, don't leave out the zener diodes as that is a sure way to destroy MOSFET T1!

For L2 a standard suppression choke is used, which is rated for at least 3 A. The inductance of the choke is not critical; any value between 50 μH and 200 μH is fine. Special inductors for switch mode supplies are also suitable; often they function even better. The value of L1 also is not critical and can be anywhere between half and double the stated 10 mH.

Usage

There are three different ways in which the revitaliser can be used.

The first is to use it in an existing system (in a car for example) to prevent sulphation from occurring in a battery with little or no sulphation. The circuit is integrated with the system by connecting it directly to the battery using as short as possible cables. Since the circuit can be left connected permanently, nothing else has to be done. The current consumption is about 20 mA, so the battery could discharge if it is not charged up occasionally.

Restoration of batteries that have already sulphated can be done in two ways. The first way is to charge the battery, remove the charger and then connect the revitaliser. Because the energy for the charging pulses is taken from the battery itself it will slowly discharge. This process has to be followed closely since a fully discharged battery has to be recharged immediately. It is likely that in practice many charge/discharge cycles will be required before a badly sulphated battery can be brought back to life.

Because the method mentioned above requires a lot of attention and carries a risk that the battery can be left in a discharged state unnecessarily (which is very bad for a lead-acid battery!), the next way is probably better.

The battery is connected to the revitaliser, with a trickle charger connected in parallel. So no chargers should be used which give a current of 7 A or more, but one which gives a maximum of 1 or 2 A. This can be left connected to the battery continuously without any problems.

As already mentioned, it is possible to connect the revitaliser in a car to the battery permanently. This does however carry the risk that pulses of around 50 V can be created in the car's wiring when the internal resistance of the battery is high, that ism when the battery is known to suffer from sulphation already. The pulses are something which electronic equipment isn't very fond of. So, if there is reason assume the battery is in a poor state, it's better to be careful and not connect the revitaliser permanently. Instead, disconnect the battery from the car before connecting the revitaliser.

Effect?

With the three LEDs it is very easy to see what effect the pulse charging has. If the sulphation does reduce, the active area of the plates increases and the internal resistance of the battery becomes less. The charge peaks supplied by the revitaliser then become smaller. This can be seen by the colour of the lit LED. With a very poor battery the red LED will be lit initially. When the pulse charging has had some effect, the red LED will go out at some stage and the yellow LED will light up. And when the green LED finally takes over from the yellow one, it is a sign that the battery can be considered to be in a fairly good state again. A final check with a voltmeter should establish that the battery voltage (without the charger of course) is in the region of its nominal value of 12 V.

This check can be extended with a discharge test. A known load is connected to the battery and you simply time how long the battery can supply the load current. The useable capacity is calculated simply by multiplying the current by the time. When a 12 V battery is loaded by a 50 W lamp, a current of about 4 A will flow. If the battery lasts for five hours then its useable capacity is 20 Ah.

When the capacity is still a long way below the nominal value given by the manufacturer, the revitalising can be continued without any objection. Any further improvement should not be expected too quickly, since the restoration can easily take from several days to several weeks, depending on the state of the battery.

(010200-1)

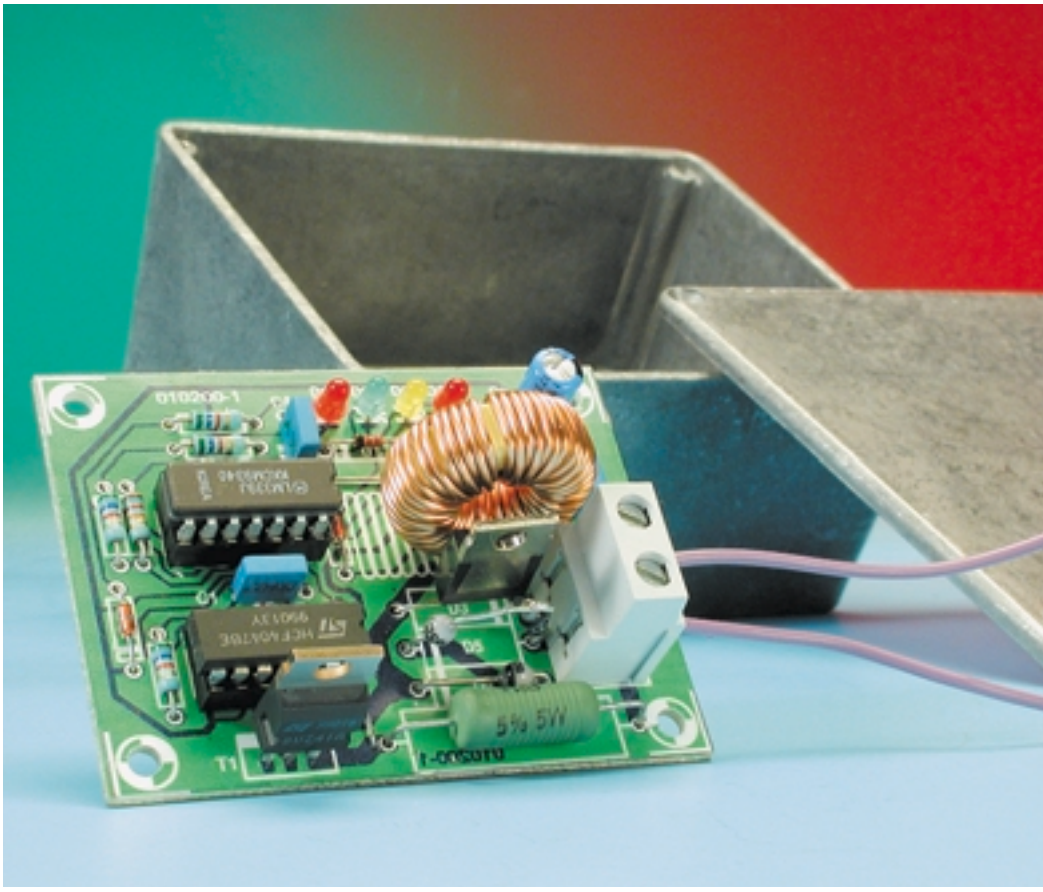


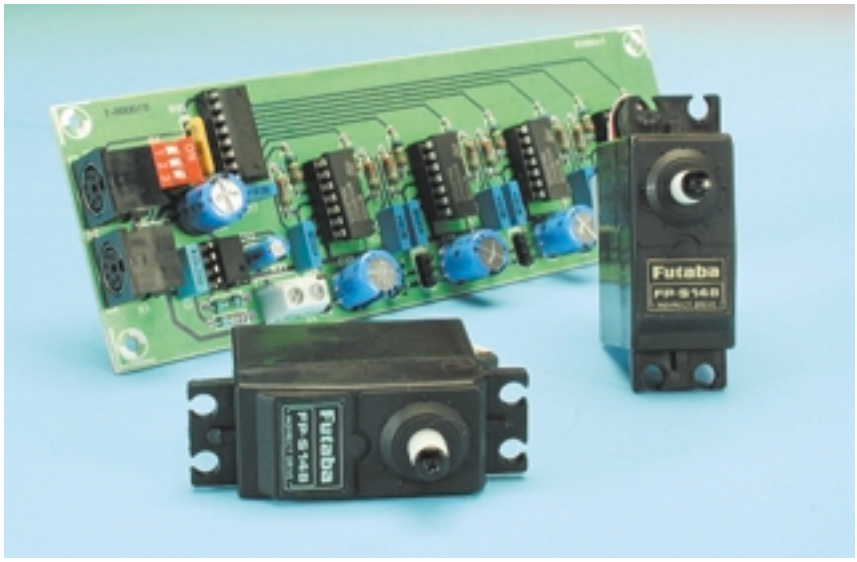
Figure 3. Keeping the possibility of interference in mind, the circuit is best mounted in a metal case.

I²C Servo Interface

control eight servos using the I2C bus

Design by C. Fuchs

Model servos have improved greatly in the past few years and now offer greater torque and speed, so much so that they are increasingly being used for other applications. The circuit described here controls up to eight servos using the well-established I²C bus. Additional cards enable expansion up to 64 servos. Software running under Win95/98 demonstrates the features of the circuit and allows simple programming on a PC using Visual Basic.



Generating an accurate Pulse Width Modulated (PWM) signal necessary to control a model servo is not an easy job for any computer particularly so if its running a multitasking operating system like Windows. Needless to say the problem gets worse when we add more servos and need to generate more control waveforms. In this situation the application of a little hardware often works wonders. The solution presented here uses a simple card driving eight servos controlled from an I²C bus on a parallel PC interface like the one presented in the March 1999 PC Topics Supplement of *Elektor Electronics*. The card allows eight servos to be controlled independently via commands sent over the I²C bus. Address selector switches on the card enable eight of these cards to be connected in parallel to the same I²C bus thereby providing a maximum of 64 servos to be controlled from the same I²C bus.

Circuit overview

An outline of the servo controller circuit is shown in **Figure 1.** The NE555 timer (IC6) is configured as an astable multivibrator, generating a pulse every 20 ms to trigger the eight monostable multivibrators (monoflops) formed by the four 556 dual timers (IC2-IC5). The pulse width output signal from these ICs will control the movement of each servo connected to that channel. A 1-ms wide pulse will move the servo arm to one end of its travel while a 2-ms pulse will move it to the other end (usually 90 degrees of movement). The control is proportional so that intermediate pulse widths will move the arm to the corresponding intermediate position. The control voltage for each of the eight monoflops is produced by an 8-channel DAC type TDA8444, which interfaces to an I²C PC port and can be controlled using software routines running in Visual basic on the PC.

High torque/speed servos can generate appreciable levels of electrical interference and despite careful PCB layout and supply decoupling this has led to occasional malfunction of the TDA8444. If you plan to use many servos it is advisable to power the servos independently from an external supply and use the +5 V derived from the I²C connector for the circuitry on this interface card only.

Circuit details

IC1 is the main component of the circuit diagram shown in **Figure 2** it is an 8-channel Digital to Analogue

Converter (DAC). The voltage level of the eight analogue outputs can be adjusted with a 6-bit resolution, giving 2^6 or 64 steps. Information to the DAC is supplied from the clock and data lines (SCL and SDA) of the I²C interface over the parallel wired mini DIN connectors K1 and K2. These connectors also allow the interface card and servos to be powered from the I²C bus. Details of the command structure are given later.

The address of the DAC on the I²C bus can be set up on DIP switch S1. This corresponds to bits 2, 3 and 4 of the DAC address (0100_A2_A1_A0_0). The address switch allows up to eight interface cards to be controlled over the same I²C bus.

The VMAX (Pin 2) input of the DAC can be used to reduce its output voltage range if it is connecting to lower voltage circuitry. In this application we are using 5 V throughout so VMAX can be left open-circuit. The output voltage of each channel (pins 9-16) of the DAC are connected via resistors R3-R10 to the control inputs (CV) of the dual timer IC's. The voltage level will be approximately in the range of 0.26 V to 3.12 V.

The NE556 dual timers are connected in the classic monoflop configuration. The timing components R = 18.7 kΩ, C = 100 nF together with a control voltage input of 0.26 V on the 6.04 kΩ series resistor will give an output pulse width of 1.0 ms. Increasing the con-

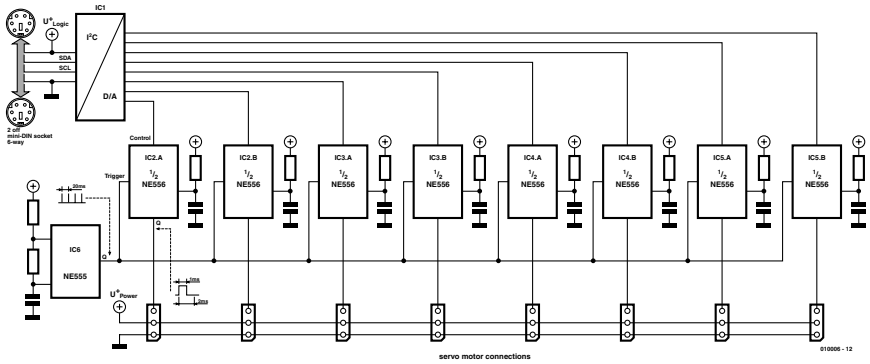


Figure 1. Servo interface block diagram.

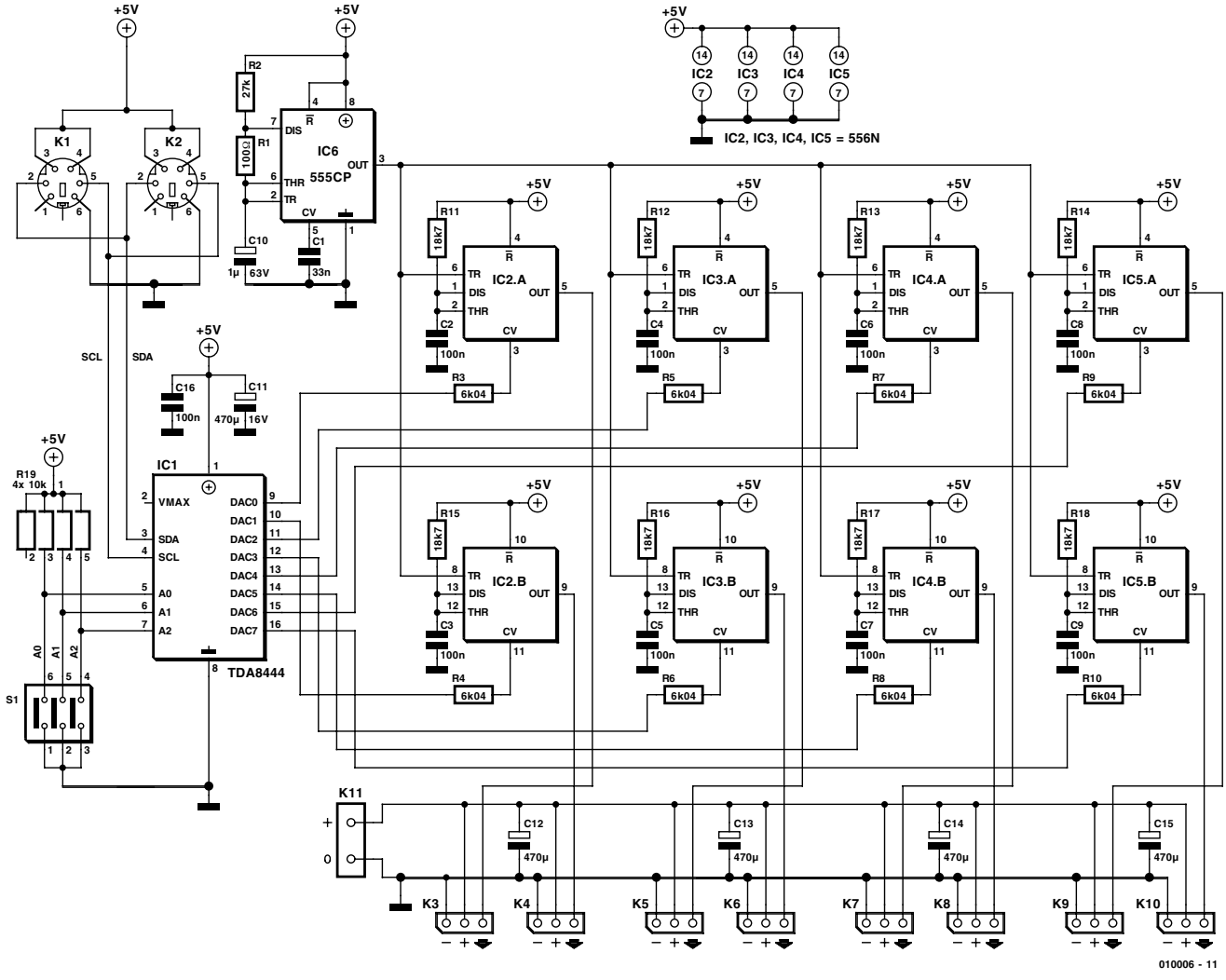


Figure 2. Servo interface circuit diagram.

trol voltage up to its maximum of 3.12 V will produce a 2.0 ms pulse. These two pulse widths represent the maximum and minimum travel of the servo and should produce a 90 degress travel of the actuating arm. In practice, component tolerances may introduce an offset or reduce the travel so it may be necessary to alter component values to achieve exactly 90 degrees if this is important for your application. The 1 ms to 2 ms pulse width signal to each servo must be repeated approximately every 20 ms. A free running astable multivibrator formed by IC6 provides this signal by generating a short 1 ms pulse every 20 ms to simultaneously trigger each of the eight monoflops. The output pulses are then taken to the three pin output connectors on the PCB along with the supply rails to drive the servos.

Different servo manufacturers in the past have always tended to use their own pin designations for the servo connectors and if you have some of these older servos the pin configurations shown in **Figure 3** should help. Be careful here, an incorrectly wired servo will be an expensive mistake. Thankfully, the situation in the UK is improving and most modern servos share a common pin configuration and are interchangeable. As mentioned earlier, servos can generate a high level of electrical noise so the PCB for this circuit is laid out with wide supply rails to the servo connectors and large supply decoupling capacitors to reduce this interference.

Construction

The PCB layout is shown in **Figure 4.** Building the card should begin by fitting and soldering the more robust components such as the wire links (9 in all), the IC sockets, connectors and pin strips to the PCB. Next, fit the passive components. Don't forget that the resistor array R19 must be fitted the correct way round. Before the IC's are fitted into their sockets, it is a good idea to connect the card to ensure that the +5 V supply is available on all the power supply pins of the IC's. Only if everything looks OK can the IC's be fitted and the card connected to the I²C interface. If many servos are connected it is important to use an external +5 V power supply and not rely on the I²C bus supply.

After power up, check that the outputs of the eight DAC channels are at 0 V (0.26 V maximum). Connect an oscilloscope to the output of IC6 (pin 3) and look for the short 1 ms pulse repeated every 20 ms. When a servo is plugged into the card, its output arm should drive to an end position. If all functions are satisfactory then the circuit up to the TDA8444 should be OK.

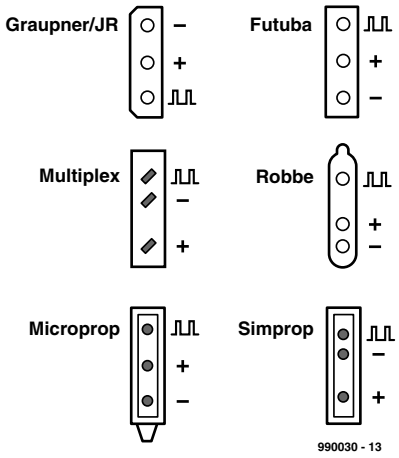


Figure 3. Pinouts of some popular servos.

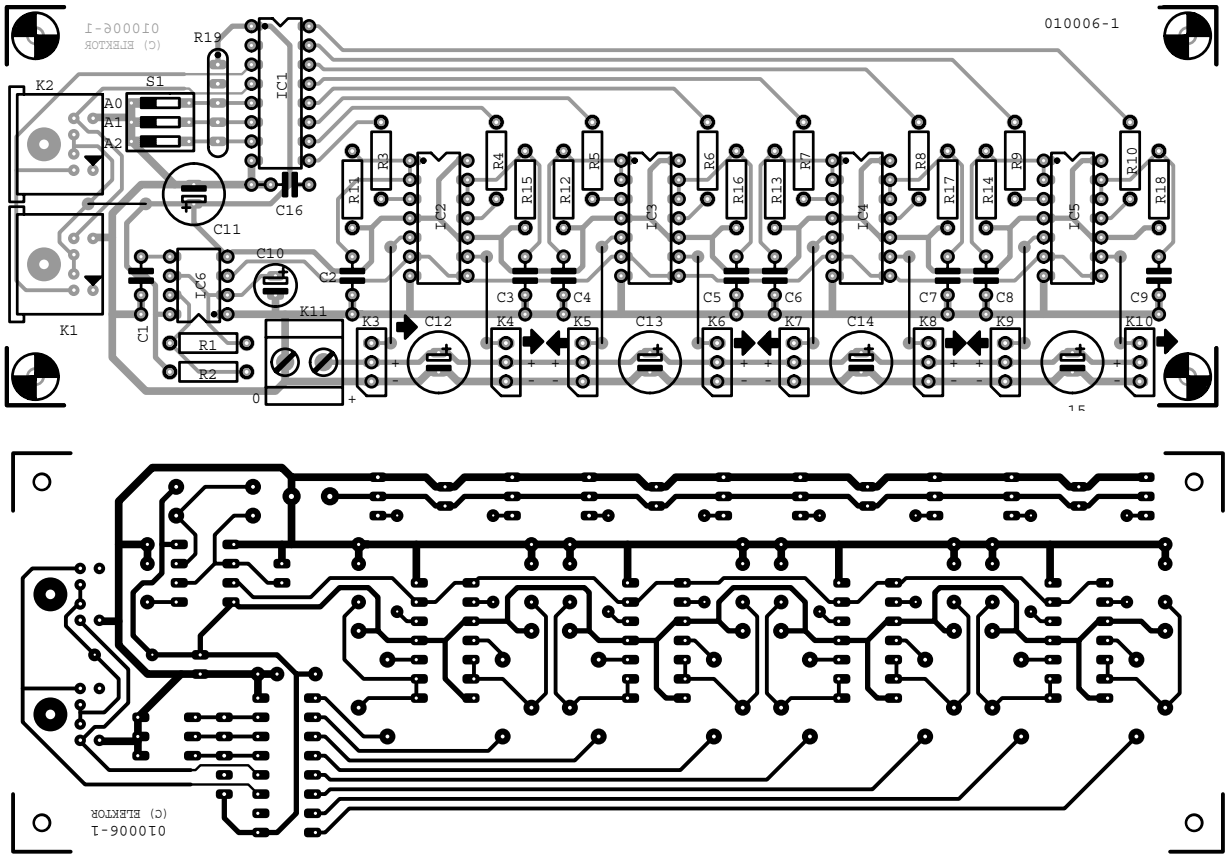


Figure 4. Simple layout using a single sided PCB (board not available ready-made).

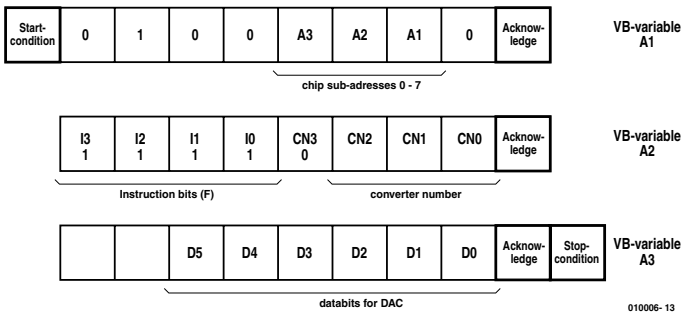


Figure 5. Format for programming the TDA8444.

The Software

The demonstration software **i2cservo.exe** is a 16-bit program and only runs under Win95/98 and not under NT. The file **iic.dll** contains software to interface to the PC hardware and must be placed in the same folder as the application or in the *system* folder in Windows. Visual Basic 3 was used to write the software. Options in the menu list allow selection of the parallel port LPT1 (378hex) or LPT2 (278hex) and also the subaddress which is decoded on the card by DIP switch S1. A 7 will talk to an interface board which has the all three switches connected to +5 V while a 0 will talk to one which has all three connected to ground. The 8 scroll bars correspond to the eight DAC output channels and therefore the servos. Clicking and dragging on one with the mouse will cause the corresponding servo to move. This simple executable demonstration programme written in Visual Basic 3 together with the source code and the PCB layout files are available as a free download from the *Elektor Electronics* website. This program is a good starting point for developers and is relatively easy to modify and expand.

When the program first starts the default chip sub-address is set to 7 (all of the address pins are switched to +5 V)

```
ChipAdress = Val(Text1.Text)
```

and the port used is set to LPT2.

```
Lpt = Lpt2
```

LPT1 and LPT2 are defined in **Modul1.bas** and have the values &H378 and &H278 respectively. Both or the settings (7 and LPT2) can of course be redefined.

```
(SubText1_Change,
SubParallel1/2_Click).
```

Next comes the check to determine if the I²C interface is connected.

```
ErrorMsg=TestConnected(LPT)
```

The corresponding failure message (if any) will be output. During operation each movement of a scroll bar

```
SubVScrollX_Change()
```

is read with X = 0 to 7 corresponding to scroll bars 1 to 8. The value of the setting is read (in the range 0 to 63) and assigned to the variable A3.

```
DacValue = VScrollX.Value
```

The maximum value MAX of the vertical scroll bars are set at 63 so there is no need to convert this value. The three least significant bits of A2 contain the code for the DAC number (0 to 7) which corresponds to that scroll bar. Bit 4 is always set to a 0 and all of the higher four bits are always set to a 1 (= 240 = Instruction F: sending data without an address will always write to the same DAC). The variable A1 contains bits 2,3 and 4 which defines the chip sub-address. The upper 4 bits contain the fixed chip address 0100 (64_d).

The three variables A1, A2 and A3 shown in **Figure 5** represent the data telegram sent over the I²C bus to the TDA8444 every time a scroll bar is moved. The routines required for this are provided in **iic.dll**. These functions are defined under GLOBAL. Sending information over the I²C bus begins with

```
ErrorMsg = StartCon( LPT)
```

To send the I²C-Start condition, and then

```
ErrorMsg=Transmit(LPT,A1..A3)
```

is used to send the three bytes. After sending the I²C bus will be released again by using the stop command

```
ErrorMsg = StopCon(LPT)
```

Any errors detected are assigned to five error variables **Er1** to **Er5** and displayed in the lower status window. See the box 'Error codes' for a list of the possible error messages. For further information on the TDA8444 a data sheet can be downloaded from the Philips Semiconductors website at http://www.semiconductors.com/acrobat/datasheets/TDA8444_3.pdf

(010006-1)

COMPONENTS LIST

Resistors:
R1 = 100Ω
R2 = 27kΩ
R3-R10 = 6kΩ04, 1%
R11-R18 = 18kΩ7, 1%
R19 = 10kΩ 4-way SIL array

Capacitors:
C1 = 33nF
C2-C9,C16 = 100nF
C10 = 1µF 16V radial
C11-C15 = 470µF 16V radial

Semiconductors:
IC1 = TDA8444
IC2-IC5 = NE556
IC6 = NE555

Miscellaneous:
K1,K2 = 6-way mini-DIN socket
K3-K10 = 3-way pinheader
K11 = 2-way PCB terminal block, lead pitch 5mm
Disk, project software, order code **010006-11**

Error codes

0	Error 0000	No failure detected
1	Error 0001	The I²C bus is not free.
2	Error 0002	No acknowledge received.
3	Error 0003	Unable to generate the stop condition.
4	Error 0004	Voltage failure on the I²C bus.
5	Error 0005	The I²C Interface is not connected.
6	Error 0006	An undefined error has occurred.

Lithium Ion Cells

Charging techniques and circuits

By G. Kleine

Where applications demand the greatest energy density and minimum weight, Lithium Ion (Li-Ion) rechargeable cells have quickly become the battery of choice despite their continued high cost. Recent advances in the cells and recharging methods are described in this article.

The ideal rechargeable battery is still a long way off. For high current application the established Nicad (Nickel Cadmium or NiCd) cell has always been preferred. Recent tests have established that the so-called memory effect (see box) is no longer a problem with this type of cell. From a cost standpoint Nicads are the cheapest cell but they do incur an environmental cost. They contain the heavy metal cadmium and should therefore not be consigned to landfill sites at the end of their life. For this reason Europe is committed to cease production of Nicads by 1998. Nickel metal Hydride (NiMH) cells are becoming increasingly seen as a better alternative. They have the advantage of not containing harmful heavy metals and offer a better energy density (cell size to stored energy ratio). We have seen a doubling of their energy density since they were first introduced with the prospect of a 2.0 Ah AA sized cell not far off.

The clear winner on the energy density front at the moment is the Lithium-Ion (Li-Ion) cell. These are relatively expensive and sensitive to mis-use but are the first choice for applications such as laptops, camcorders, mobile phones and portable equipment where weight (lithium is the lightest known metal) and capacity of the battery pack is of critical importance. Recent developments have produced high current cells suitable for powering passenger vehicles and in one case a full-size glider with motor assist! The German aircraft manufacturer Lange has taken advantage of the low weight of Li-Ion cells to power its 'Antares' electric motor glider. The prototype was developed for Ni-MH cells but replacing them with Li-Ion will give the 500 Kg aircraft a climb rate of 885 ft/min, carrying the pilot to an altitude of almost 10,000 ft with its 57 hp (42 KW) brushless electric motor.

Cell Structure and Characteristics

Returning to earth for a moment we will now delve into the internals of the cell. A significant advantage of Li-Ion is that the cell potential is 3.6 or 3.7 V. This means that two to three cells of Ni-MH or Nicad (which have a cell potential of 1.2 V) can be replaced by just a single Li-Ion battery.

The Li-Ion cell is composed of a graphite anode and a lithium cobalt oxide or lithium manganese oxide cathode in a organically fluid electrolyte containing dissolved lithium salt which supplies the lithium ions. Manganese oxide cathodes give a cell potential 3.7 V while cobalt oxide produce 3.6 V. As in all things in life there is no gain without pain. In the case of the Li-Ion cell it is its sensitivity to improper use. The recharging voltage is 4.20 V for cells with manganese oxide cathodes and 4.10 V for cobalt oxide cathodes. This voltage level must be maintained to within 50 mV if the cell is not to be permanently damaged. During discharge it is also important to ensure that the cell voltage does not fall below 2.4 or 2.5 V, otherwise cell life will be seriously compromised.

The 'Ion' part in the battery name simply refers to the fact that the Lithium should never occur in its metallic form in the battery. During charging Lithium ions (Li+) are collected on the graphite anode from the electrolyte.

Cell Chemistry (charging):

positive Pole: $LiCoO_2 \rightarrow Li1\text{-}n\ CoO_2 + n\ Li+ + n\ e-$

negative Pole: $C + x\ Li+ + x\ e- \rightarrow CLin$

The essence of rechargeable cells is that this process is reversible so as you would expect the opposite process occurs during discharge.

The consequences of improper use

Charging with too high a voltage can easily damage Li-Ion cells. If the charging voltage is increased above its optimum value of 4.1 V or 4.2 V it will lead to internal gassing, overheating and eventual explosion. A voltage increase of just 1% above this optimum level can cause the lithium ions to begin to convert to metallic lithium in the cell. This in turn reacts violently with water in the electrolyte and at this point I think we should all retire to a safe distance or risk a close encounter with the cell and its contents. On the other hand if the charging voltage is below its optimum level it leads to a significantly under-charged cell. A voltage level of just 100 mV below the optimum will result in a 7% reduction of the stored capacity. If this were not bad enough, Li-Ion is also sensitive to how low the cell voltage is allowed to fall during discharge. Deeply discharging the cell leads rapidly to an irreversible decline in cell capacity.

As you will appreciate, Li-Ion cells are not the most fault tolerant on the market. For this reason they are not currently available for general-purpose use and are not in standard battery package outlines (AAA, AA and C cell etc.). Li-Ion cells tend to be used in custom battery packs for specific equipment i.e. laptops, mobile phones, camcorders etc. where the equipment is specially designed to accept a Li-Ion pack and correct charging equipment will be guaranteed.

Battery Packs

Li-Ion battery packs are usually fitted with some form of electronic protection to prevent mis-use. **Figure 1** shows some typical circuits. One of the simplest methods (**Figure 1a**) uses an in-built NTC sensor to convey cell temperature to the connected equipment. A more sophisticated circuit giving protection against over and under charging is shown in **Figure 1b** here the cell potential is measured by a protector IC and the charge or discharge current can be interrupted by turning off MOSFET T2 or T1 respectively. In each case the inherent MOSFET body diode is used to allow current to flow through the unswitched MOSFET. Switching off both MOSFETs will effectively disconnect the battery pack.

The Protector IC prevents over-voltage during charging and deep discharge. The protector IC itself consumes less than 1 µA in standby mode.

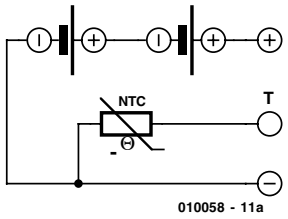


Figure 1a. Circuit diagram of a Li-Ion pack with temperature sensor.

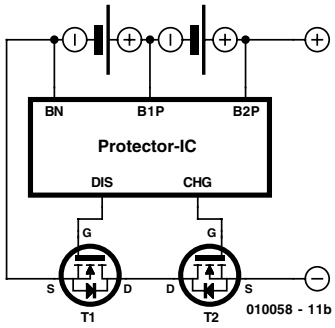


Figure 1b. Circuit diagram of a Li-Ion pack with a protector IC.

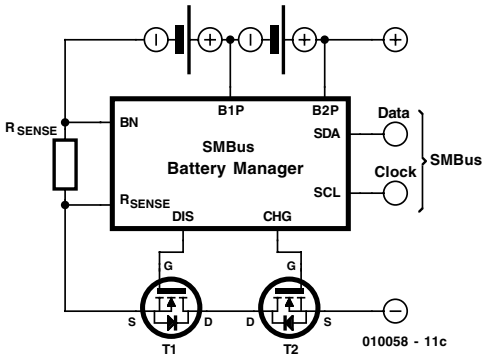


Figure 1c. Circuit diagram of a Li-Ion pack with SM-Bus battery management IC.

Figure 1c shows a battery pack with a built-in battery manager IC using a System Management Bus (SMB) interface. The IC measures each cell potential and also the current drain via R_{sense}, a low resistance (a few 10's of milliohms) sense resistor. This allows the management IC to determine the amount of charge remaining in the battery pack and send this information to the charging circuit of the equipment over the clock (SCL) and data (SDA) lines of the SMB two-wire interface. Further information of this interface is given later.

For safety reasons Li-Ion batteries have an over-pressure valve fitted to each cell. This allows pressure in the cell caused by high environmental temperature (e.g. fire) to be vented to the atmosphere. A low resistance Positive Temperature Coefficient (PTC) element is also included in the cell. If high currents are drawn, this element warms up and its resistance increases thereby reducing the short circuit current.

Li-Ion charging

Li-Ion cell recharging is carried out using a constant voltage, current limited source and requires close monitoring of the cell voltage. Incorrect charging will quickly lead to permanent loss of cell capacity or worse. A typical recharging cycle will begin with the charging equipment measuring the no-load cell potential. If this is below 2.5 V then the cell is in a deep discharge condition and requires a 'prequalification' charging phase. This is performed by trickle charging at 5 mA until the cell potential reaches 2.5 V. At this voltage level the charger will start the fast charging stage. The current is limited to a value from 1C to 2C (where C is the Ah rating of the cell) until the cell potential reaches 4.1 V (for cobalt oxide) or 4.2 V (for manganese oxide). Now the charger begins the constant voltage phase, it maintains this voltage (+/– 5 mV) and monitors the current until it falls below a pre-defined level. This top-off phase ends when the charging current falls to less than 5% of the current supplied during the constant current phase, i.e., 0.05 C to 0.1 C. The cell is now fully charged.

It is permissible as long as the cell is not fully charged to use pulsed charging with a voltage greater than 4.2 V.

For safety reasons it is necessary to use two timers to limit the charging times. One timer limits the fast charging phase whilst another limits the total charging time. If the fast charge timer runs out before the cell potential reaches 4.2 V the charging process is terminated and an error message will be sent indicating a faulty cell. The 'total time' timer will terminate the top-off phase if it is not switched off during the normal cycle.

Li-Ion cells have a very low self discharge so they do not require a maintenance charge in fact this would lead to an over-charge condition of the cell. An NTC sensing element is used to ensure that the cell temperature stays within its operating limits of +2°C to +45°C. Cell overheating will cause the controller to switch off the charging process until the temperature has returned to normal.

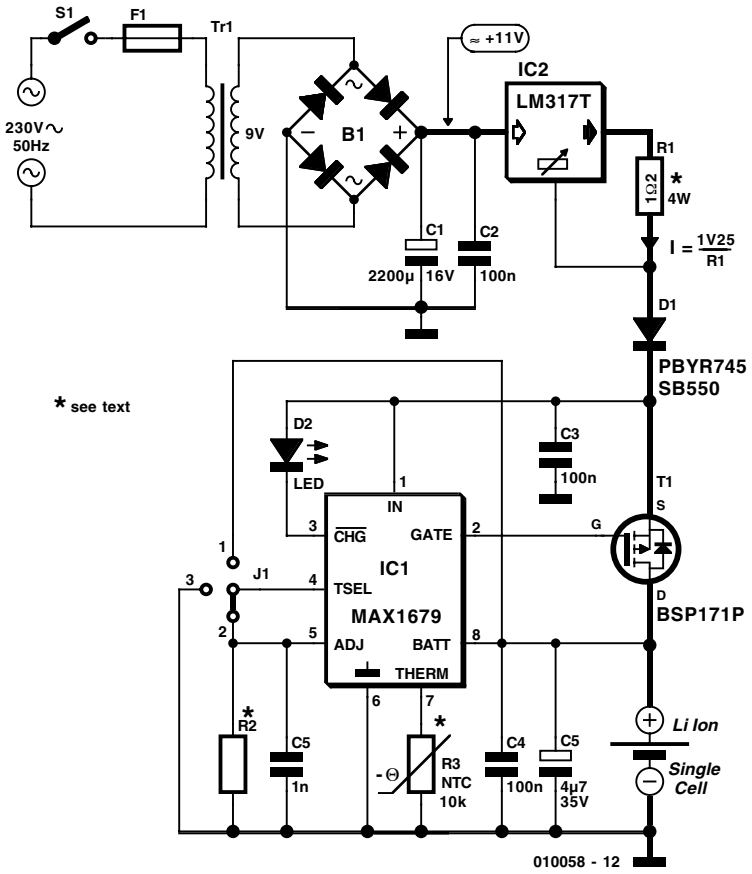


Figure 2. Simple Li-Ion charger using the MAX 1679.

Table I. LED readout with MAXI679

LED state	Function
Flashing	Qualifying (Vbatt < 2.5V)
On	Charging (fast charging or top-off charging)
Flashing	Fast charging finished
Flashes every 3.5 s	Charging finished

ICs for charging Li-Ion cells

The circuit in Figure 3 shows a simple charger suitable for re-charging single Li-Ion cells. At the heart of the circuit is the MAX 1679 (IC1) from MAXIM. IC1 measures the cell potential at its BATT input and switches MOSFET T1 using a pulsed waveform with a variable mark-space ratio the so that the charging current from the constant current source formed by IC2 and R1 can be controlled at will by IC1. During the fast charge phase T1 is on constantly

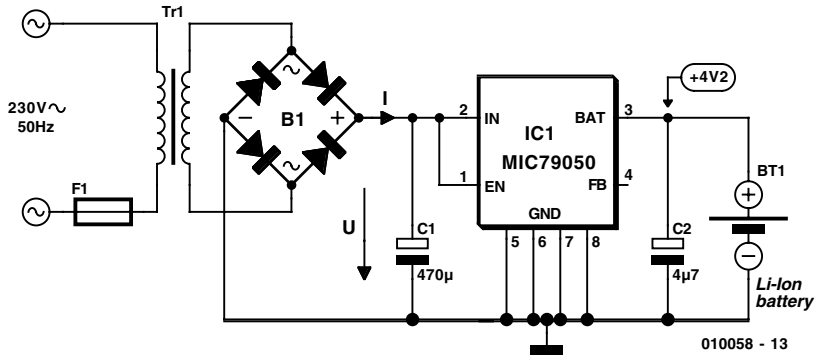


Figure 3a. Simplest Li-Ion charger using a mains adapter and the MIC 79050.

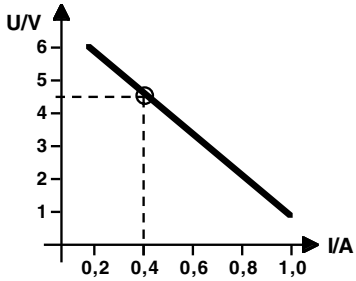


Figure 3b. Load line characteristics of an unregulated mains adapter.

Memory effect? What memory effect?

Recent tests by a German consumer magazine compared the performance of AA format rechargeable cells. Their test results can be summarised as:

Memory Effect

The testers could not find any evidence of this effect (for NiCd and NiMH cells). Cells were fully charged and then discharged to half capacity. After repeating this cycle 50 times they still delivered their full capacity! This was found true for both NiMH and NiCd cells.

Self discharge

In contrast to earlier findings, self discharge was less of a problem in NiMH cells than in NiCd. The best NiMH cell tested was given a 'good' rating with only a small loss in capacity after 80 days at 20 °C. Three other NiMH cell manufacturers scored 'satisfactory'. NiCd cells were all rated 'poor' in this category.

Conclusions:

The memory effect seems to be a thing of the past. Nicads are now 'out' for most applications. They are useful mainly for power tools and model builders but this situation is changing slowly.

The Rechargeable Alkaline (RAM) cells (distributed principally by Rayovac) were found to be a relatively poor alternative. They are expensive and can only be recharged a limited number of times (25 max). Testers swapped the RAM cells for standard alkaline cells (so called non-rechargeable primary alkaline cells) and found that they had exactly the same properties as the expensive RAM cells! But regular *Elektor Electronics* readers are already aware of this phenomenon since our article in December 1996.

and during the top-off phase T1 will be pulsed. The charging method used by this IC adheres closely to the charging method described above. The MAX 1679 can measure cell potential with an accuracy of less than 1%. An LED indicates the status of the charger (see Table 1).

Schottky diode D1 ensures that the Li-Ion cell does not discharge through the body diode of T1 after the charging process is finished. An NTC thermistor R3 is connected to the THERM input of IC1 and must be in close physical contact with the cell to sense its temperature. The ADJ input switches an internal voltage reference that allows selection of the terminal charging voltage of either 4.2 V or 4.1 V depending on the type of battery used. The jumper at input TSEL selects the maximum charging time (this time can be between 2.8 and 6.25 hours). At the end of a charging cycle the MAX 1679 will switch itself into low power mode so that the IC draws less than 1 µA from the battery.

A simpler Li-Ion charger can be constructed using a mains unit adapter. The output characteristic of a typical unregulated adapter is shown in **Figure 3b**. You would expect an ideal voltage source to have a flat load line i.e., it would produce the same voltage irrespective of load. However the transformer winding impedance of a typical unregulated mains unit causes the output voltage to drop as load current increases. The circuit shown here in Figure 3a puts this property to good use. The transformer will need to produce +4.5 V at a current of 0.5 to 1.0 times the capacity of the cell in A/hr. In figure 3b this output current is 0.4 A. During the constant current phase of charging the battery is switched directly to the output of the transformer. The charge current is limited by transformer impedance and the cell voltage gradually rises. When 4.2 V is reached the MC79050 switches to constant voltage charging and the MC79050 will monitor the cell current until it is fully charged.

Figure 4 shows a Li-Ion charging circuit using the IC LM 3622 from National Semiconductor. This circuit uses a PNP transistor as a linear regulator to maintain the voltage on the cell at 4.1 or 4.2 V. The charging current is related to the value of R_{sense} and is given by the equation:

Charge current = 0.1 V/R_{sense}

A heatsink should be used to dissipate the energy in transistor T1.

Table 2. Protection ICs for Li-ion battery packs.			
Manufacturer	**Type**	**Function**	**No. of cells**
Maxim	MAX 1665	Lithium-Ion Battery Pack Protector	2, 3 or 4 *
Maxim	MAX 1666	Advanced Lithium-Ion Battery Pack Protector	2, 3 or 4 *
ON Semiconductor	MC33348	Lithium Battery Protection Circuit	1
ON Semiconductor	MC33349	Lithium Battery Protection Circuit	1
ON Semiconductor	MC33351A	Lithium Battery Protection Circuit	3
Philips Semiconductors	SAA1502	Safety IC for Li-ion	1
TI / Unitrode	UCC3952	Li-ion Battery Protection IC	1
** version of IC dependent on number of cells*			

Table 3. Simple charger ICs for Li-ion batteries.			
Manufacturer	**Type**	**Function**	**No. of cells**
Maxim	MAX1679	Single Cell Li+ Battery Charger	1
Maxim	MAX1736	Single Cell Li+ Battery Charger for Current-Limited Supply	1
Micrel	MIC 79050	Simple Lithium-Ion Battery Charger	1
Linear Technology	LTC1730	Lithium-Ion Battery Puls Charger	1
Linear Technology	LTC1731	Lithium-Ion Linear Battery Charger Controller	1
Linear Technology	LTC1732	Linear Lithium Battery Charger Controller	1
Linear Technology	LTC1734	Lithium-Ion Linear Battery Charger	1
National Semiconductor	LM3420	Lithium-Ion Battery Charge Controller	1 ... 4 *
National Semiconductor	LM3620	Lithium-Ion Battery Charge Controller	1 ... 2 *
National Semiconductor	LM3622	Lithium-Ion Battery Charge Controller	1 ... 2 *
TI / benchmarq	bq2400	Linear Li-Ion/Li-Polymer Charger	1 ... 2
TI / benchmarq	bq2057	Li-Ion Charge Management IC	1 ... 2
** version of IC dependent on number of cells*			

Smart battery systems

With the advent of laptop PCs and mobile phones it is important to know accurately just how much life is left in the batteries. There are few things more annoying when trying to save work after a low battery warning to find that the battery has too little charge to make the back-up. Intel and Duracell are developing a so called 'intelligent' battery pack, these packs send information over a two wire interface to the equipment identifying the type of cell in the pack (NiCd, NiMH, Li-Ion etc.) and the amount of charge left. This Smart Battery System (SBS) will therefore ensure optimum cell life by providing information to the equipment about the battery and the best way to recharge it. The battery pack effectively controls its own charging! The message and command format of the SBS is well defined and can be used licence-free by all

(*www.sbs-forum.org*).

Using information from the SBS interface, equipment will at last be able to accurately predict remaining run time. ICs using the SBS are produced by MAXIM (MAX1645, MAX1647/1648 and MAX1667) and Linear Technology (LTC1759). Also noteworthy for Li-Ion cell monitoring is the DS2760 from Dallas Semiconductor – this device uses its own Dallas 1-wire bus.

Table 4. Universal charger ICs.					
Manufacturer	**Type**	**no. of cells NiMH/NiCd**	**no. of cells Li-Ion**	**cell type**	**Recognition end of charge processs**
Maxim	MAX1772	2...4	2...4	Lead-acid, Li-ion, NiCd, NiMH, universal	current and voltage setting via controller
National	LM3647	2...7	1...4	Li-ion, NiCd, NiMH, universal	-ΔU, voltage, temperature, time
Linear Technology	LTC1325	1...8	1...3	Microprocessor-Controlled Battery Management System for lead-acid, Li-ion, NiCd and NiMH batteries	voltage, temperature, time

Table 5. Lithium-Ion current-source controllers.				
Manufacturer	**Type**	**no. of cells Li-Ion**	**Function**	**Recognition end of charge process**
Maxim	MAX1737 MAX1757 MAX1758	1...4 1...3 1...4	Stand-Alone Li-Ion Charger Controller	Voltage and Current Limit, Thermistor, Max Time

New products

In the past year many new ICs have appeared on the market specifically for monitoring and supervising the recharging of Li-Ion cells. The main players are Maxim, Linear Technology, ON Semiconductor (A branch of

Web Addresses

Information on batteries:
www.ebatts.com/tips.asp
www.batteryweb.com
www.duracell.com/Fun_Learning/index.html
www.energizer.com/products/alkaline/
www.rconline.net/archiv-04/akku-faq/nimh_faq.shtml
www.rconline.net/archiv-04/akku-faq/akku.shtml
www.rconline.net/archiv-01/autobatterie/autobatterie.shtml
www.funk-akku.de (= www.akku-doktor.de, www.modellbauakku.de)
www.accu-profi.de/batterie_lexikon/index.html

Batttery Manufacturers:
www.accucell.de
www.accusonne.de
www.ansmann.de
www.durcacell.com
www.duracellusa.com
www.energizer.com,
www.energizer-eu.com
www.hitachi.com/products/material/batteries/index.html
www.panasonic.com/industrial_oem/battery/battery_home.htm
www.philipsbatteries.com
www.power-sonic.com/
www.renata.com
www.saft.alcatel.com
www.sanyo-energy-europe.com
www.sonnenschein-lithium.de
www.tdk-europe.com/
www.toshiba.com/taec
www.varta.de

IC Manufacturers:
www.maxim-ic.com
www.national.com
www.onsemi.com
www.linear-tech.com
www.micrel.com
www.dalsemi.com
www.semiconductors.philips.com

Motorola), National Semiconductor and Philips. Micrel also produce the simple MIC 79050 used in the above example. **Tables 2 to 5** give an overview of some of the current ICs and their main features. In conclusion, a visit to any of the company websites or the battery related sites listed in the web-links should provide a wealth of further information related to this topic.

(010058-1)

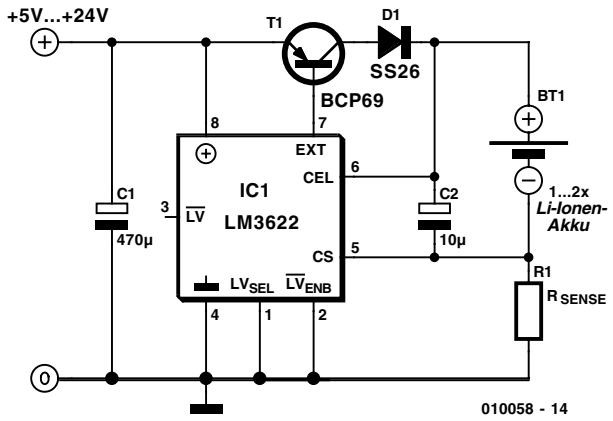


Figure 4. Li-Ion charger using a LM 3622 Linear regulator.

Literature:

1. **Battery Charge/Refresh Station,**
Elektor Electronics
October & November 1999.
2. **Application Note 64,**
Using the LTC1325 Battery Management IC, Linear Technology, August 1996.
3. **Battery Technology Today & Tomorrow,**
Elektor Electronics May 2000.
4. **Primary-Battery Refresher,**
Elektor Electronics December 1996.

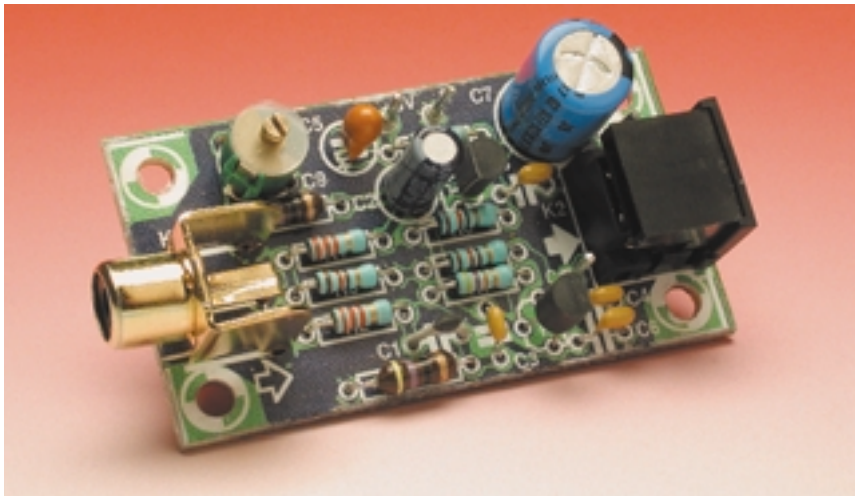
Video to S-VHS Converter

From composite (CVBS) to C/Y

Design by W. Foede

foede.koeln@web.de

Having problems connecting standard video to your S-VHS equipment? You need this neat circuit. It takes a composite video signal and converts it into the C/Y input signals used by S-VHS. To simplify construction a PCB design is supplied.



A standard video signal (also known as Composite Video Baseband Signal or CVBS) contains the black/white and picture synchronisation information together with the colour information in one signal. S-video signals (also known as Y/C) are slightly different in that these two parts of the picture signal are supplied as separate signals. Y is the luminance signal containing both black/white and sync information while C is the chrominance signal containing colour information.

Equipment using Y/C signals has the advantage of better picture resolution (greater bandwidth), less cross-colour interference and fewer filters for the signals to squeeze through.

The circuit described here allows a standard CVBS signal to be connected to the input of S-VHS equipment. It will however not improve picture quality. The circuit requires a +5 V power source supplying approximately 50 mA.

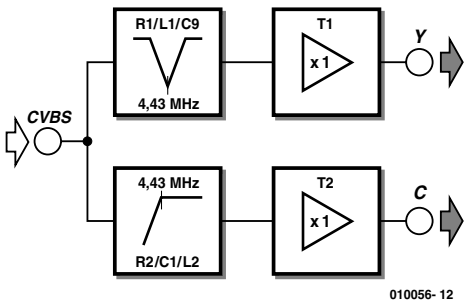


Figure 1. Block diagram showing separation of the CVBS signal into its Y and C components.

The bandwidth of a composite video signal extends from 50 Hz up 5 MHz. Colour information is mixed into the signal on a sub-carrier at 4.43 MHz with a bandwidth of –1.2 to +0.6 MHz. The job of our circuit is to try to cleanly separate the Y and C signals from each other (a standard TV needs to be able to do this as well). The method chosen here is one of the simplest. A notch (or bandstop) filter tuned to the sub-carrier frequency is used to remove the C from the Y information. A disadvantage of this method is that the picture information suffers a bandwidth reduction from 5 MHz to around 4 MHz thereby giving poorer picture resolution. The C signal is produced by passing the composite signal through a 4.43 MHz high-pass filter to remove most of the lower frequency Y components however some Y remains and causes visible cross-colour effects on the picture.

Modern TVs achieve this Y/C separation by using a comb filter, and a delay line. The advantage is a greater bandwidth of the Y signal and no cross-colour effects. S-VHS video recorders also use this tech-

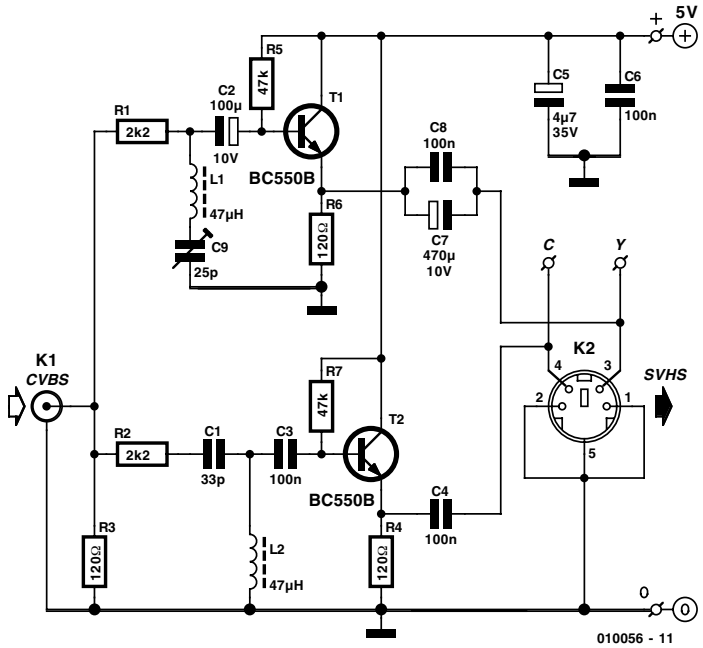


Figure 2. Circuit diagram.

nique so that the Y component of the recorded signal will have a better bandwidth.

The circuit block diagram is shown in **Figure 1**. The C part of the signal is cleanly filtered out from the composite signal using an RLC high pass filter circuit while the Y part uses an RLC notch or bandstop filter to remove the C information. Transistors T1 and T2 act as buffers to produce low impedance Y and C outputs.

The circuit diagram shown in **Figure 2** closely follows the block diagram layout. R2, C1 and L2 form a high pass filter to recover the C signal. The filter output is coupled via C3 to the base of transistor T2. This transistor is a common-emitter configuration and buffers the filter while providing a low impedance output via C4 to the S-VHS connector.

The bandstop filter used in the Y path is made up of R1, L1 and C9. L1 is again a standard 47 µH fixed inductor while C9 is an adjustable trimmer, this allows some optimisation of the picture. The Y signal contains frequency components going down to 50 Hz so the signal coupling capacitors C2 and C7 need to have a larger value than in the C signal path, otherwise low frequency parts of the signal would be attenuated. The ac-coupled low impedance signals C and Y are output on pins 4 and 3 respectively of the S-VHS connector K2. Two supply decoupling capacitors C5 and C6 complete the circuit.

The input and output impedance of this circuit deviate from the optimum value but were chosen to ensure that the output signal levels are correct. At 75 Ω the signal levels should be 0.3 V_{pp} (burst-amplitude) for C and 1.0 V_{pp} for Y.

The layout of the PCB is shown in **Figure 3**. As in all RF circuits it is important to keep the component leads as short as possible when fitting them to the PCB. An earth plane surrounds the tracks so a little extra care is required when soldering to ensure that no solder bridges are formed. In place of a genuine Hosiden S-VHS socket we have specified a standard mini DIN socket for PCB mounting. It will happily accept a Hosiden plug as long as the small plastic stop pins are first removed from the plug.

During set-up, trimmer C9 should be adjusted to produce minimum Moiré (colour patterning) effect on the picture. The circuit was successfully tested on many different S-VHS inputs. Generally the bandstop filter in the Y path was necessary but with a WinTV card it did not make any noticeable difference if the filter was in or out. In this case you could simply bypass the Y circuitry and just use the composite video signal as the Y signal.

Finally a reality check, as mentioned before, the quality of the S-VHS compatible output signal generated here cannot be any better than the composite video input signal. To get real S-VHS picture quality, you do of course need a 'genuine' S-VHS signal source connected directly to the Y/C input of the monitor, TV or recorder.

(010056-1)

COMPONENTS LIST

Resistors:
R1,R2 = 2kΩ2
R3,R4,R6 = 120Ω
R5,R7 = 47kΩ

Capacitors:
C1 = 33pF
C2 = 100µF 10V radial
C3,C4,C6,C8 = 100nF ceramic
C5 = 4µF7 35V tantalum bead
C7 = 470µF 10V radial
C9 = 25pF trimmer capacitor (3-25 p)

Inductors:
L1,L2 = 47µH

Semiconductors:
T1,T2 = BC550B

Miscellaneous:
K1 = Cinch socket, PCB mount, e.g., T-709G (Monacor/Monarch)
K2 = 6-way Mini-DIN socket, PCB mount, angled pins

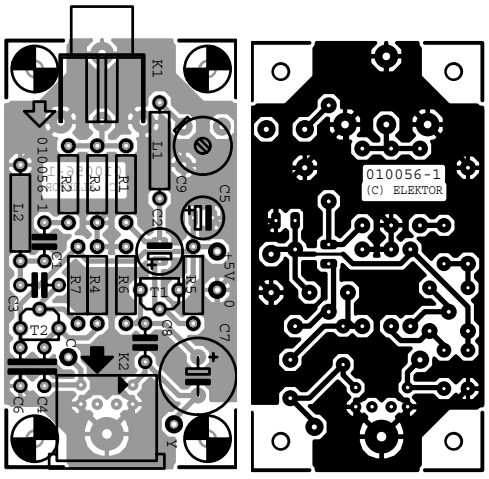


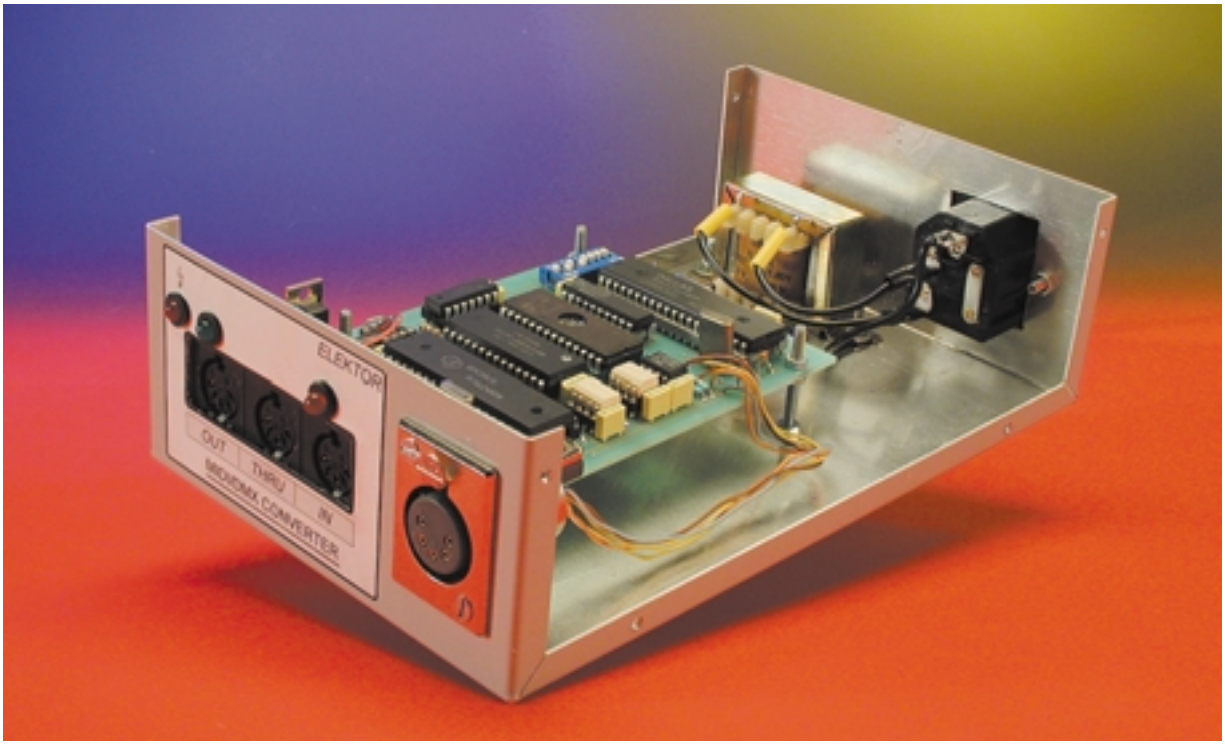
Figure 3. PCB layout. The groundplane is typical of an RF layout (board not available ready-made).

MIDI-DMX Interface

control your DMX network from a sequencer !

Design by B. Bouchez

We've recently described the principle of DMX512-based remote control, which allows automated projectors or flood lights to be controlled on up to 512 channels, via an RS485 link. Every DMX system comprises a master and several slave devices responding to DMX commands.



Actually, there are two approaches for the realisation of a DMX512 master: either a dedicated controller is used (such as the MARTIN2518, for example), or software running on a PC.

The main advantage of the first approach lies in its simplicity and low cost. On the down side, the capabilities, especially in programming and memory, are generally limited. Moreover, the user interface of some of these controllers is sometimes original if not odd, to put it mildly.

As always, the use of a PC with a DMX interface allows you to do almost everything, while offering unbeatable capabilities. Actually, there is almost no concert, public event or TV show without some form of PC-controlled (stage) lighting.

However, the PC solution also has two disadvantages: first, the need of

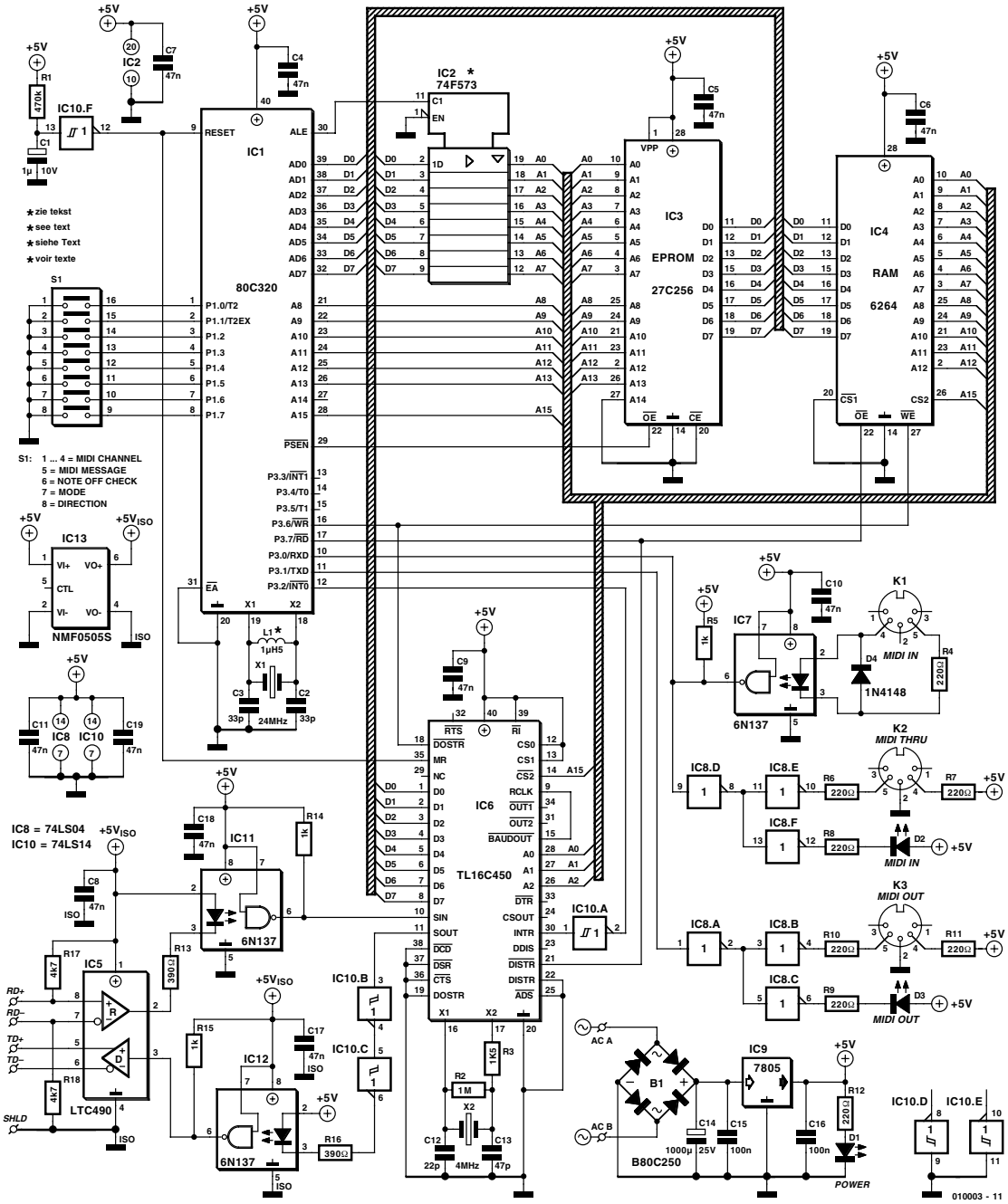


Figure 1. This circuit diagram of the DMX-to-MIDI interface effectively shows a stand-alone microcontroller system.

a specific interface, since the PC's serial port is not compatible with the DMX512 standard; second, suitable software to support these interfaces is costly an difficult to use for non-specialists.

The interface described here allows, at a small outlay, to control any DMX system from any MIDI sequencer, on PC, Macintosh or stand-alone. Rather than controlling sounds, the sequencer controls DMX levels, colour changes, gobo calls, etc.

This interface integrates another very useful function: a DMX to MIDI converter that enables you to look what happens on your DMX control and identify the main problems that can arise. Moreover, this function can be used to capture sequences generated by DMX controller in your own sequence software.

Lastly, a suite of software utilities has been designed especially for this interface, including a driver library for easy creation of your own software under Windows. This toolkit also includes two test programs.

Finally, the author of this project has written two powerful but easy to use light control programs employing the interface described here.

The circuit

As you could imagine, the interface is built around a microcontroller in charge of various dataflows. Initially, the well-known 80C32 was chosen, but following a field test of the

prototype it transpired that the software load on the CPU was rather too high for reliable operation. In the end the 80C320 was elected, which, being pin compatible as well as software compatible with the 80C32, runs three to four times faster thanks to a completely redesigned kernel. None the less, as we will see a bit further on, it is possible to use a 80C32 in a 'light' version of the interface. To support the fast data transfers (250 kbits/s on DMX, 31.25 kbits/s on MIDI), the micro-controller is clocked at 24 MHz.

Although the current version of the software works perfectly, it should be able to improve or extended it in the future. Consequently a external EPROM (IC3) is used to hold the program. Don't be surprised to see an address line tied to ground rather than to the 80C320 address bus. Although the software may also be loaded into a 27C64 or 27C128, these EPROMs are generally slower and more costly than the 27C256. Because not all memory is actually used, unused pins have been tied to ground to simplify the PCB drawing. Please note that the EPROM speed must be 120 ns or less, due to the processor speed. For this same reason, the data/address demultiplexer, IC2, should normally be a 74**F**573. The use of a 74HCT573 is only required in a few rare cases, see the section on testing further on.

Because the amount of data to be handled exceeds the processor's on-chip memory capacity, an external RAM is used (IC4). As will be explained a bit further on, it is possible to omit this memory device at the cost of some interface capabilities. As for with EPROM, a smaller RAM device could be used, but these 'small' memories may be hard to find, especially devices with a fast access time. A relatively large memory is used here, which should be easy to obtain at a small additional cost. Again, it is mandatory to use a 120 ns or faster memory.

The MIDI interface is built directly around the microcontroller, using its internal serial port. The physical interface around IC7 and IC8 is classical and fully complies with the MIDI specification. Two signalling LEDs were added to visualize the data flow on MIDI IN and MIDI OUT. This kind of extension costs next to nothing but is really practical in actual use when problems arise between the interface and the sequencer.

At the DMX side, the interface is built around IC6, a Type 16C450 fast UART (Universal Asynchronous Receiver Transmitter). This chip is the evolution of the 'good old' 8250, which has been part and parcel of the PC's serial port since the beginning. The 16C450 and 16C550 can easily achieve data throughput in excess of 250 kbits/s, where the 8250 is limited to 19,200 bits/s (even it can reach 115.2 kbits/s in practice).

This UART is a 'must have' in this circuit because it greatly simplifies the job of the microcontroller. The UART circuit integrates a Break detector, which is used for DMX synchronisation. On the 80C320's serial port, such detection would have to be implemented in software.

The UART integrates its own crystal oscillator and a programmable divider that allows almost any baud rate to be set. Note that a clock signal, which is 16?, the baudrate may be found on the RCLK pin of the chip. On this board, you can find a 4-MHz (16?250 kbits/) frequency. This is very practical when testing the board.

You will notice that IC10 is a 7414 Schmitt trigger. This IC is used to create the reset pulse for the UART and microcontroller, which must start simultaneously for software reasons. A 7404, which has the same pinout, but no specified trigger behaviour, can lead to serious problems when the board starts up. For the record, the author lost several days on one of the prototypes chasing a non-existing software bug, only to discover that the UART started after the micro-controller.

An LTC490 converter with separate send and receive buffers is employed to transform TTL levels (used by UART) into RS485 levels. Resistors R17 and R18 are biasing devices on the DMX lines and serve to avoid false 'Break' detections when a DMX input is not connected. Without these resistors, a false Break condition would cause erroneous synchronisation.

Especially in long distance DMX networks, there is a real risk of earth loops being created. To prevent problems, the DMX side is optically isolated from the rest of the circuit. Since DMX is quite fast, it is mandatory to use high speed 6N137 opto-couplers.

This part of the circuit is powered by a small DC/DC converter. It is possible to lower the cost of the interface by omitting the converter and the opto-couplers. In that case, you have to connect the SOUT pin of the UART directly to DIN (pin 3) input of LTC490, and the SIN pin of the UART to DOUT (pin 2) of the LTC490. Remember, however, that this configuration greatly increases the risk of damage to the interface in case of energy return on DMX512 lines. From our experience, it happens more frequently that you could think.

Last thing, the interface is configured by an octal (8-way) DIP switch, S1, directly connected to the processor's P1 port. There is no need for pull-up resistors, since these are integrated in the P1 port.

The software

The value of a microcontroller-driven board is totally dependent on the software it hosts. In the case of the present interface, a lot of software is involved! To give you an idea, DMX-receive processing can use up to 60% of processor's computing capabilities. On the MIDI side, it can reach 45% for a heavily loaded (= very busy) MIDI line. Moreover, receiving data is only a part of the functions carried out by the CPU, transmission in the background being the other.

Before detailing the different modes of the interface, let's take a look to some generic things about it.

On the MIDI side, our interface recognises two message types: Note On (90h) and Control Change (B0h). The message type used for the MIDI-to-DMX as well as the DMX-to-MIDI direction is selected by means of switch **S1-5**. With the switch in ON position, the interface employs Control Change messages. In the other case (OFF position), the interface uses Note On messages.

These two MIDI messages were chosen because they comprise two data bytes, the first for note or control number, the second for the value of velocity or control.

Since a MIDI channel can transmit 128 notes or controls, the first byte will be used to identify the corresponding DMX channel.

The second byte of data will be used to set the DMX value for the channel selected by the previous byte. As MIDI supports only 128 values (DMX supporting 256), a scaling operation is performed by truncating the least-significant bit and subsequent rotation. The correlation between MIDI and DMX values is

Table 1. MIDI/DMX values correspondence

MIDI Value	◄►	DMX Value
0		0
0		1
1		2
1		3
2		4
2		5
3		6
3		7
.		.
.		.
126		252
126		253
127		254
127		255

Table 2. Mode selection by S1-7 and S1-8

S1-7 (Basic Mode/ 4 channels)	S1-8 (Direction)	Mode
OFF	OFF	1
OFF	ON	2
ON	OFF	3
ON	ON	4

given in **Table 1**.

Switch **S1-6** allows or disallows recognition of Note Off messages (to make sense to this switch, the interface should be in Note On mode, not Control Change). If S1-6 is closed, the interface will treat any Note Off received on configured MIDI channel(s) as a blackout command for the corresponding DMX channel. Thanks to this function, programming MIDI sequencers is generally easier. In Modes 2 and 4, S1-6 has a different function, allowing a special mode to be activated, called 'Delta'. In Delta Mode, only those MIDI values that have changed since the previous cycle are returned to the MIDI. Obviously, this considerably lightens the traffic on the MIDI link. More importantly though, it allows DMX sequences to be registered on a MIDI sequencer.

The protocol converter of the interface can be used under four modes, selected by **S1-7** and **S1-8,** see **Table 2.**

Mode 1: the converter transforms MIDI data on the channel selected by S1-1 to S1-4 (see table 3) into 128 DMX values.

Mode 2: the converter transforms the first 128 DMX channels received into MIDI messages and send them on MIDI OUT to the MIDI channel selected by S1-1 to S1-4. MIDI message content is defined by S1-5 (see below). Note that MIDI transmission is continuous to insure that data are regularly refreshed.

Mode 3: the converter controls 512 DMX channels on the DMX output (this is called a 'DMX world'). Since it is not possible to send 512 values on a single MIDI channel, the interface will use 4 consecutive MIDI channels, each of these controlling 128 values. In this mode, only S1-3 and S1-4 give the base MIDI channel, the three others being the following channels. Possible values for the MIDI set-up are given in **Table 4**.

Mode 4: in this mode, the converter transforms the 512 received DMX channels into MIDI messages and sends them continuously to MIDI OUT. As in previous mode, four consecutive MIDI channels will be used, selected by S1-3 and S1-4 (see Table 4). As with Mode 2, the MIDI message type is selected by switch S1-5.

Note that, in all cases, the interface recognises and transmits DMX frames with 'Start Code' equal to 0 only.

Table 4. Selection of MIDI channel by S1-3 and S1-4 for Modes 3 and 4

S1-3	S1-4	MIDI Channels
OFF	OFF	1, 2, 3, 4
ON	OFF	5, 6, 7, 8
OFF	ON	9, 10, 11, 12
ON	ON	13, 14, 15, 16

Building the interface

Although the schematics may appear quite complex, building this interface is relatively simple, mainly thanks to the double-sided PCB specially designed for the project. As always, it is highly recommended to use high quality sockets for all ICs, especially the microcontroller and its peripheral devices.

Table 3. MIDI Channel selection by S1-1 through S1-4 for Modes 1 and 2

S1-1	S1-2	S1-3	S1-4	MIDI channel
OFF	OFF	OFF	OFF	1
ON	OFF	OFF	OFF	2
OFF	ON	OFF	OFF	3
ON	ON	OFF	OFF	4
OFF	OFF	ON	OFF	5
ON	OFF	ON	OFF	6
OFF	ON	ON	OFF	7
ON	ON	ON	OFF	8
OFF	OFF	OFF	ON	9
ON	OFF	OFF	ON	10
OFF	ON	OFF	ON	11
ON	ON	OFF	ON	12
OFF	OFF	ON	ON	13
ON	OFF	ON	ON	14
OFF	ON	ON	ON	15
ON	ON	ON	ON	16

As already mentioned, it is possible to build a 'light' version of the interface. In that case, you can only use Mode 1 (which is the most common and useful mode). For this version, there is no need for SRAM IC4, which consequently need not mounted. Since DMX-receive is inactive in Mode 1, there is no need to mount IC11, R13 and C18. Last but not least, it is possible to use the good old 80C32, which will be cheaper than the 80C320.

While on the subject of these controllers, some retailers are unable to specify the maximum frequency of the ICs they sell. Worse, they will not hesitate to sell 16 or 20 MHz devices processors and claim compatibility with a 24MHz clock. We even surprised one day a retailer supplying an 8052 (which is NMOS and limited to 12 MHz) as a 80C32 (CMOS), saying that it will only run a little warmer to the touch. You have to know that a processor specified at 20 MHz may seem to work correctly at 24MHz, but will cause problems one day. We expericed the case of a 80C32 whose serial port was totally 'crazy' during interrupt processing, when the rest of the chip was running correctly.

For this project, if you want to build the 'light' version, we recommend to use Philips 80C32 -IBPN, -UBPN or -IFPN versions, which are guaranteed at 24 MHz. As far as we know, Intel processors of this family are limited to 20 MHz, but that may nedd to be verified. If you want to make the 'full' version, you have to use a Dallas Semiconductor 80C320MCG (this is the most frequently seen version).

For the UART, you can use the Texas Instruments TL16C450 or National Semiconductor NS16C450. You can also use TL16C550

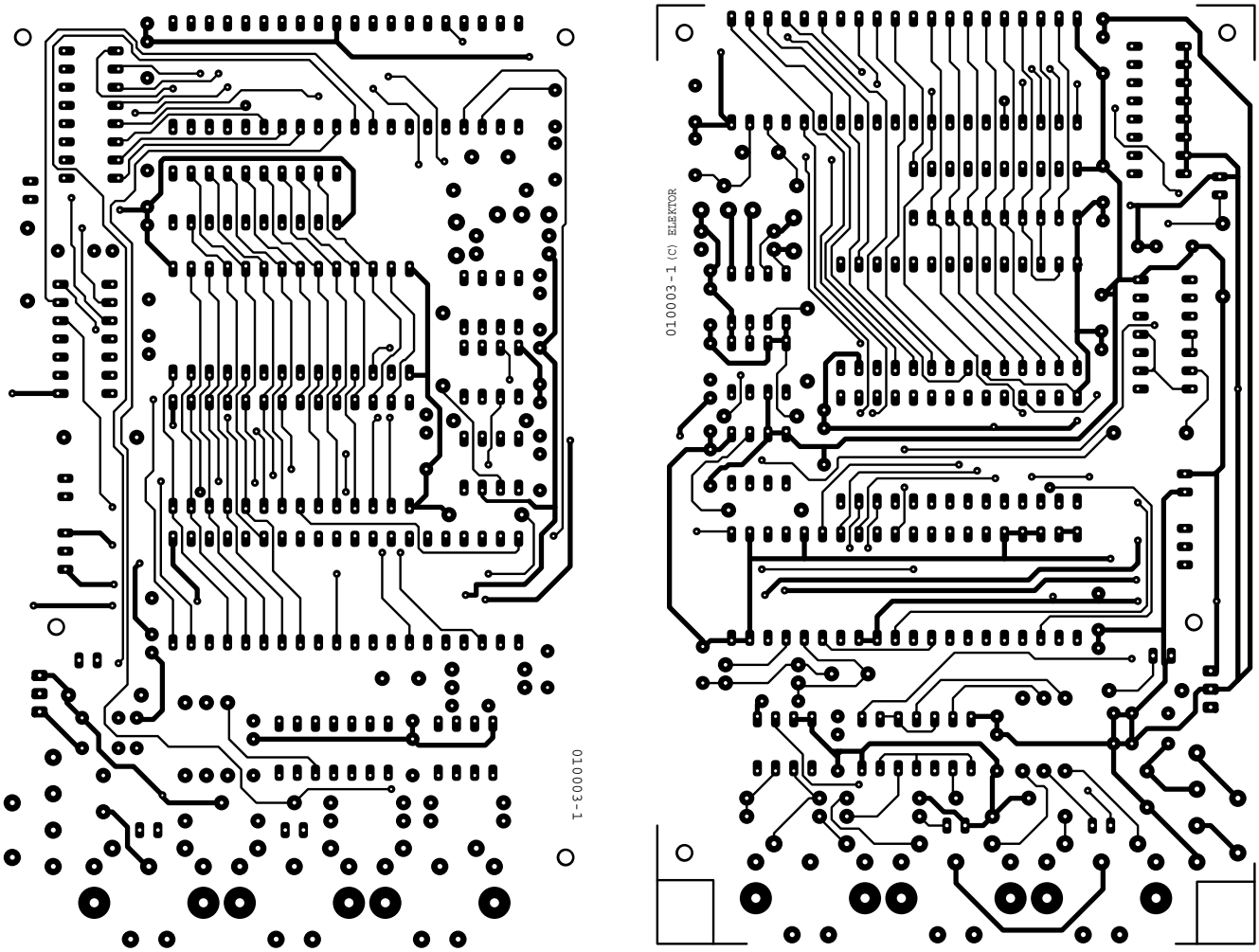


Figure 2. Copper track layout and component mounting plan of the PCB designed for the interface (board available ready-made).

or equivalent without any problem. These components may be found with more generic type numbers like 16450, 16C450, 16550 or 16C550. Note that you may find such components on serial port add-on boards for PCs (such boards may be picked up cheaper than the component alone). Do not use 82C50 or 8250, since these chips are normally not compatible with 4-MHz crystals.

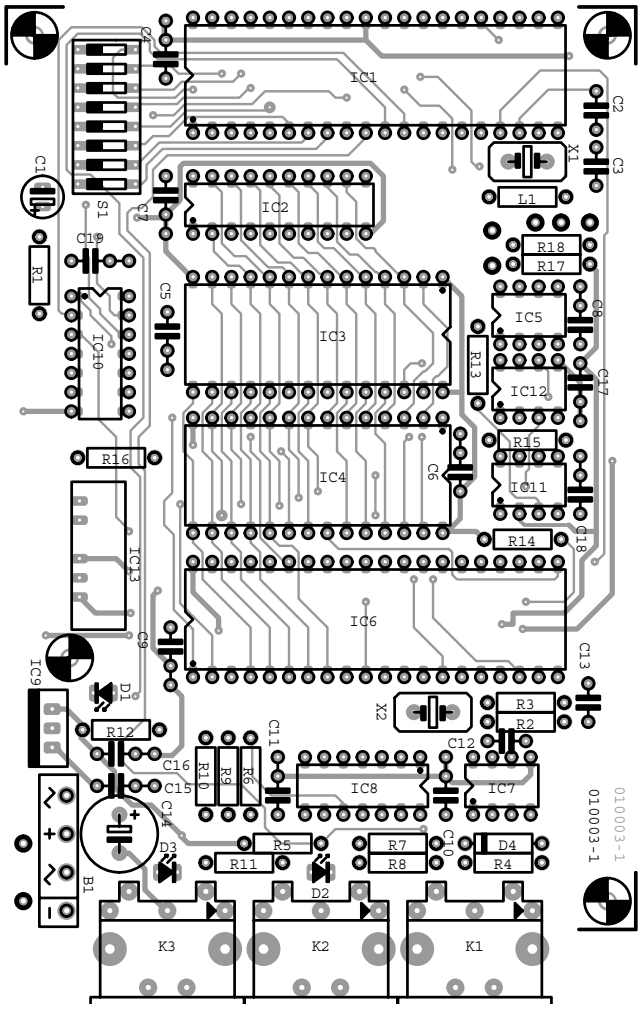
The only problematic component (as far as availability is concerned) is the DC/DC converter IC13, a NMF0505S. Any compatible **insulated** DC/DC converter does the job, but have a good look at the pinout which may differ between devices.

Concerning the two crystals on the board, you may notice that the soldering pads are larger than the insulating rings of most of crystals, which can lead to unwanted short-circuits. We recommend mounting the crystals either 1 mm above the PCB, or to put a small piece of plastic between crystal and PCB.

The microcontroller crystal is used with a small **optional** inductor, which serves to force the crystal into overtone mode. Always test your board for the first time without the inductor. If you find 24 MHz on the X2 pin of the processor (with a 'scope or a frequency meter), everything is okay. If you find only 12 MHz, you have to add the inductor. If the crystal is found to oscillate at the wrong frequency, the DMX output will operate correctly, but not the MIDI input or output. In case you have neither an oscilloscope nor a frequency meter, you may test the board in Mode 2 or 4 using the DMX Tester software on the project disk. If you can get some (correct) MIDI data, you can be sure that the crystal is oscillating at the right frequency.

Table 5 : DMX connectors pinout

	Female connector	Male connector	Female connector
Pin	DMX OUT	DMX IN	DMX OUT/IN
1	Ground (SHLD)	Ground (SHLD)	Ground (SHLD)
2	Transmit DMX- (TD-)	Receive DMX- (RD-)	Transmit DMX- (TD–)
3	Transmit DMX+ (TD+)	Receive DMX+ (RD+)	Transmit DMX+ (TD+)
4	Not connected	Not connected	Receive DMX- (RD–)
5	Not connected	Not connected	Receive DMX+ (RD+)



COMPONENTS LIST

Resistors:
R1 = 470kΩ
R2 = 1MΩ
R3 = 1kΩ5
R5,R14,R15 = 1kΩ
R4,R6-R12 = 220Ω 0.5W
R13,R16 = 390Ω 0.5W
R17,R18 = 4kΩ7

Capacitors:
C1 = 1µF 10V
C2,C3 = 33pF
C4-C11,C17,C18,C19 = 47nF
C12 = 22pF
C13 = 47pF
C14 = 1000µF 25V radial
C15,C16 = 100nF

Inductors:
L1 = 1µH5

Semiconductors:
D1,D2,D3 = LED
D4 = 1N4148
IC1 = DS80C320MCG (Dallas Semiconductor)
IC2 = 74HCT573 or 74F573
IC3 = 27C256 (programmed, order code **010003-21**)
IC4 = 6264 (RAM)
IC5 = LTC490 (Linear Technology)
IC6 = TL16C450 (Texas Instruments)
IC7,IC11,IC12 = 6N137 (
IC8 = 74LS04
IC9 = 7805
IC10 = 74LS14
IC13 = NMF0505S

Miscellaneous:
B1 = B80C250
K1,K2,K3 = 5-way DIN socket, PCB mount, pins at 180 degrees
PC1- PC7 = solder pin
S1 = 8-way DIP switch block
X1 = 24MHz quartz crystal
X2 = 4MHz quartz crystal
PCB, order code **010003-1**
Disk, project software, order code **010003-11**

The heatsink on IC9 regulator is almost optional, since the board has only modest power consumption. A small piece of aluminium of a few square centimetres is sufficient. If you choose a metal box (as we recommend you should), you may use one of the sides as a heatsink. Do not forget that the case of IC9 is connected to ground. It is better to isolate it from the box to avoid unexpected ground loops.

The power supply can be either a small external mains adapter (in this case, you can omit rectifier B1 and capacitor C14), or a 9 V, 5 VA mains transformer, with adequate fuse protection.

The link to DMX world is by way of a standard 5-pin XLR connector, as prescribed by the DMX specification. Since there are a lot of different pin configurations for PCB-mount XLR connectors, we decided to use chassis-mounted versions with wires to and from PCB.

It may be difficult to find male XLR 5-pin connectors for chassis mounting. Fortunately, you may avail yourself of an unclear area of the DMX specification: since pins 4 and 5 are reserved for 'optional DMX data', you may use them for DMX input. The DMX pin-out, according to the DMX specification is given in **Table 5**.

Concerning the housing, we highly recommend the use of a **metal box with a protective earth connection**. In fact, this circuit is made for professional use in 'harsh' RFI environments. No greater embarrassment than the lights system 'hanging' during a show, only because the processor is locked up as a result of RFI.

The simplest solution is to integrate the transformer inside the box, and to use a standard mains appliance socket, with an earth connection linked to box. If you want to make a 'high-end' version, you can even put a filter between the mains socket and the transformer.

One of our prototypes, we should not forget to add, was capricious when starting, giving erroneous UART operation. The only solution to get it to work properly was to add a push-button in parallel with capacitor C1 so we were able to reset the board manually, if necessary. Since fitting this button, the pro-

totype has always worked correctly. Even if this phenomenon appeared only on one of our prototype, we should mention this workaround. In the end we discovered that the BAUDOUT pin of IC6 was stuck at 66 Hz, when it should go to 4 MHz.

With the board fully populated, it is ready for its final test.

The critical moment: testing

To test this interface, you need a DMX receiver, such as the one inside a DMX floodlight or projector. You can also hire a DMX tester with a DMX values display (light equipment rental companies will be a able to help you).

For MIDI, the easiest thing is to use a PC with a MIDI interface and running Windows 95/98, the two purpose-designed software tools will prove useful.

Connect the PC's MIDI OUT to the MIDI IN connector of interface, and connect the DMX OUT link of the interface to the DMX IN input of the floodlight (or the DMX tester). **Do not forget the DMX terminator**, especially if the DMX line is long. Configure the DMX slave (light or tester) on any DMX address, and select Mode 1 or 3 on the interface. Also select a MIDI channel and the MIDI message type on it.

Power up the interface. The POWER LED should be the only one to light up. If not, disconnect the interface and find the problem.

Launch a program named DMX TESTER, which is on the project disk. Select the MIDI interface used to connect our interface, the selected MIDI channel and the mode (1 or 3). With the 'Start' combo box, select a DMX channel corresponding to the floodlight or projector, then move the slider. The DMX slave should respond to DMX commands. If you want, you can select the 'Group' option, to send simultaneously the DMX level given by the slider on all DMX channels between 'Start' and 'Stop'.

To test the other direction (DMX/MIDI), you should have a DMX generator, like a stage lighting console. In a small number of cases you may be able to avoid it, but the test will only be on the MIDI section (note that if the first test was okay, there is little chance that the DMX section is not working properly for that direction).

Set Mode 2 (or 4) on the interface and select a MIDI channel with the associated MIDI message to send. Connect the MIDI OUT connector of the interface to the MIDI IN input on the PC. Connect the DMX generator to DMX IN on interface, if possible with a DMX terminator.

Start the DMX VIEWER program (on disk), then give it the MIDI channel and the MIDI message selected on the board. Do not forget to select a MIDI interface if necessary. The moment the interface is powered, the MIDI OUT LED of the interface should light. If you have a DMX generator, you can change DMX values and verify their evolution on the screen. If you do not, this test should be run with Delta Mode de-activated (S1-6 OFF), else the absence of changing values on DMX will cause the MIDI link to remain 'stalled'.

If, during the tests, you notice anomalies in DMX values received or transmitted (using VIEWER, you'll notice that certain DMX addresses are blocked to erroneous values), you need to consider replacing the 74F573 by a 74HCT573. The reasomn is that some RAM ICs do not mix very well with 74F ICs. In these rare cases, changing to HCT solves the problem.

How to use the interface?

This interface is suitable for many applications, including, of course, lightshows but also servicing, maintenance and lights testing.

In this last case, the easiest solution is to use DMX TESTER and DMX VIEWER on the disk (order code **010003-11**).

For lightshow use, the basic tool associated with the interface will be a MIDI sequencer. This interface is in principle compatible with all existing sequencers on PC, Mac or standalone machines. The only difference with the established use of a sequencer is that note velocities (or control values) do not control synthesizer sounds, but light intensity levels, mirror positions, gobos types and speed, etc., depending on the type of DMX device.

To control something on DMX, just record notes (or control values), with velocities (or values) corresponding to the desired DMX levels, then play the sequence and see how the DMX units behave. For the first experiments, we recommend not using the Note Off recognition or long note durations, to ensure that the DMX control has enough time to send the message. As long you 'feel' effects of MIDI to DMX, you can adapt the programming of the sequencer to your needs.

Possibilities given by a sequencer are almost unlimited, although it must be saisd that you have to be thoroughly familiar with your sequencer software or machine to rapidly arrive at a good result.

Our 'field' experience proved beyond doubt that programming and using a sequencer was a too heavy a job for a number of small and medium-size club installations, where the DJ typically doubles as the LJ (Light Jockey). For these cases, the author has developed two programs, called SoftController I and SoftController II, which allow easy and quasi-automatic control of a DMX system.

Describing these programs here would take us too far away from the goal of this article. You should know however that these software utilities represent a stage light control panel, whose look is something of a standard in the world of stage lighting. We have deleted almost all menus. To arrive at almost intuitive operation, you have to click on the buttons

On project disk 010003-11

copyright.txtl	author's copyright statements
dmx_tester.exe	DMX interface test program for Windows
midi_dll.dll	.dll file
midi_dmx.asm	EPROM contents (assembler file)
Midi_dmx.hex	EPROM contents (hexadecimal file)
mode1.asm	assembly code file for Mode 1
mode2.asm	assembly code file for Mode 2
mode3.asm	assembly code file for Mode 3
mode4.asm	assembly code file for Mode 4
viewer.exe	DMX interface viewer program for Windows

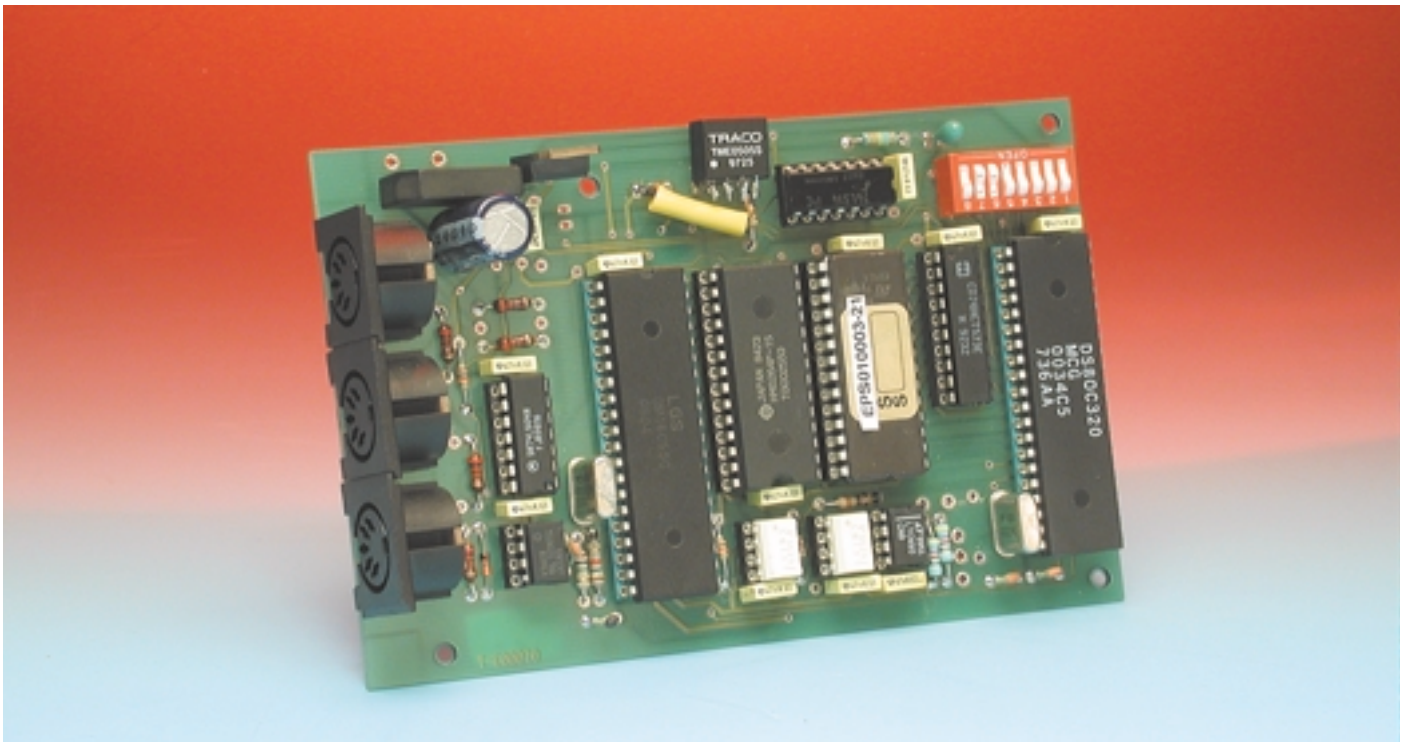


Figure 3. An early prototype of the interface board. Some improvisation was necessary around the 0505S due to a small design error which was corrected later.

to activate them. Similarly, sliders are controlled by mouse motion, etcetera.

The software is actually employed in Belgium, to control professional lighting systems.

Finally, a bit of advice: do read the user manual (after printing it for example), before attempting to use the programs, since there are a lot of possibilities.

The software on the diskette has taken many hours of intensive work to develop and test. There is no copy protection, or evaluation version to pay afterwards, since that would only disturb 'honest' users. In spite of this, the software utilities related to the MIDI-to-DMX Interface is not 'copy-free' — by buying the project disk, you obtain only one user licence for all the software. Please do not 'give away' the software or use copies, but buy a corresponding number for your interfaces. In view of the above, the project software is not available as a free download from the *Elektor Electronics* website.

(010003-1)

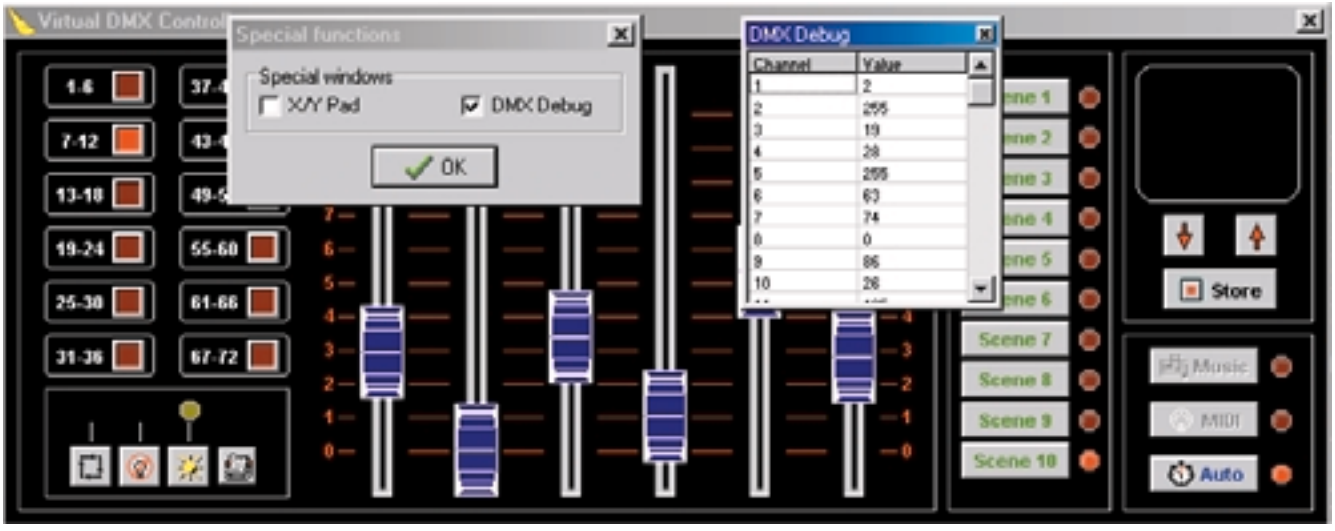


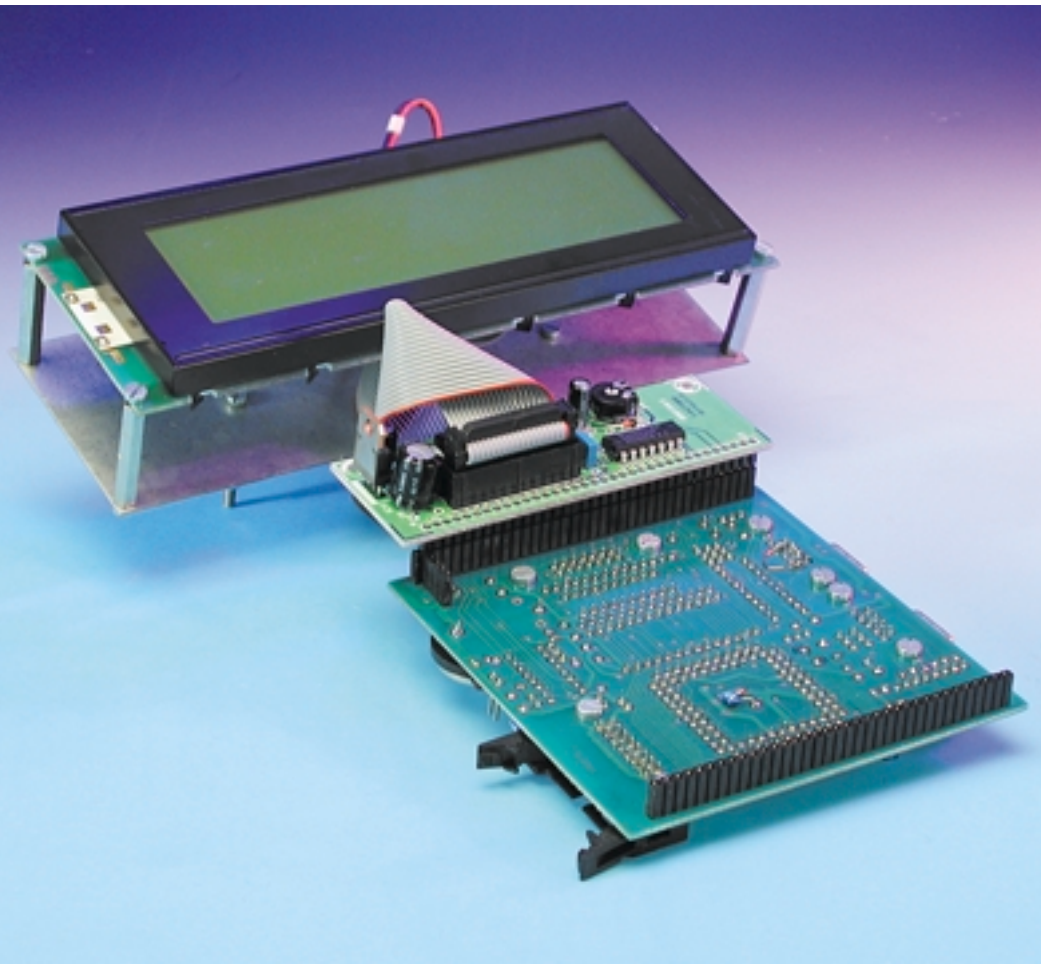
Figure 4. Scvreendump showing the SoftController program in action.

Graphic LCD Module for 8051 Micros

playing around with pixels and bytes

Design by Luc Lemmens

In this article, we describe how a display using the popular Toshiba T6963 controller can be driven by an 8051 (or compatible) microcontroller. Read on to find out how to use the full display for a graphic image and how to convert an image on the PC into the pixels needed by the display.



For the familiar alphanumeric displays used in many *Elektor Electronics* projects, Hitachi has set the industry standard with its HD44780 controller. Nearly all manufacturers use this controller in their LCD modules, with the result that in practice it does not matter which display you attach to your circuit, although the connector may vary from one brand to the next.

The situation with monochrome graphic modules is similar, although here we unfortunately find that there is much less standardisation on a single controller. One controller that is certainly used quite often is the Toshiba T6963, so we decided to see how a display fitted with this controller could be driven using an 8051 or compatible microcontroller. Our starting point was the compact 80C597 board described the January and February 2000 issues, along with its associated EEPROM (976510-1), but in principle any other microcontroller system could be used just as well.

The graphic display that served as the guinea pig for our experiments was a Sharp LM24014 module with 240 × 64 pixels, but any other display

with a T6963 controller could be used for the experiments in this article without any problems. For displays with a different pixel ratio, small modifications will be needed in the software described in this article.
Naturally, a lot more could be said about driving this type of display than we can fit into the confines of this article. For other applications and options, such as using the standard text mode (comparable with an alphanumeric display), we refer you to the data sheet for the T6963 controller or the documentation for whatever display module you may have that is fitted with this controller.

The LCD module

Three registers in the graphic module are used to drive the LM24014: the data register, the command register and the status register. Communications takes place by first writing one, two or no values to the data register and then sending an instruction to the command register. After this, a value can be read back from the data register if this is supported by the instruction in question. The status register must be polled prior to each individual read or write task, in order to check whether the display has finished processing the previous task and is ready to receive and process new data.
The command and data registers can be directly accessed, but not the status register. If we want to read the status, we must first send a Read Status instruction to the command register, following which the status byte can be read from the data register. As can be seen from **Figure 1**, there are various status bits that indicate which processes the display can handle. In some cases, it is necessary to check several status bits and verify that they have the proper values before instructing the display to process new information.

Interface

The 20 connection pins of the display include 8 bi-directional data lines, a reset line and 4 control signals ($\overline{CE}$, $\overline{RD}$, $\overline{WR}$ en C/$\overline{D}$, which we will discuss later). The LCD module also has an internal character ROM, just like the better-known alphanumeric dis-

Status Bit	Description		Function
STA0 (BUSY1)	Busy flag to indicate whether T6963C is ready to accept instruction	"0""1"	=NOT READY =READY
STA1 (BUSY2)	Busy flag to indicate whether T693C is ready to accept data read or write	"0" "1"	=NOT READY =READY
STA2 (DARRDY)	Data Auto Read Ready flag (Only valid in Data Auto Read/Write modes (section 8.6))	"0""1"	=NOT READY=READY
STA3(DAWRDY)	Data Auto Read Ready flag (Only valid in Data Auto Read/Write modes (section 8.6))	"0""1"	=NOT READY =READY
STA4	-	-	-
STA5 (CLR)	Clear flag indicating operation of the T6963C	"0""1"	=NOT CLEARED =CLEARED & Operating
STA6 (ERROR)	Error flag for Screen Peeking and Screen Copy commands (section8.8 & 8.9)	"0""1"	=Address Pointer Valid = Address Pointer out of Graphic Area
STA7 (BLINK)	Blink flag to indicate status of Blink condition	"0" "1"	=Display OFF =Normal (Display ON)

Figure 1. Contents of the status register.

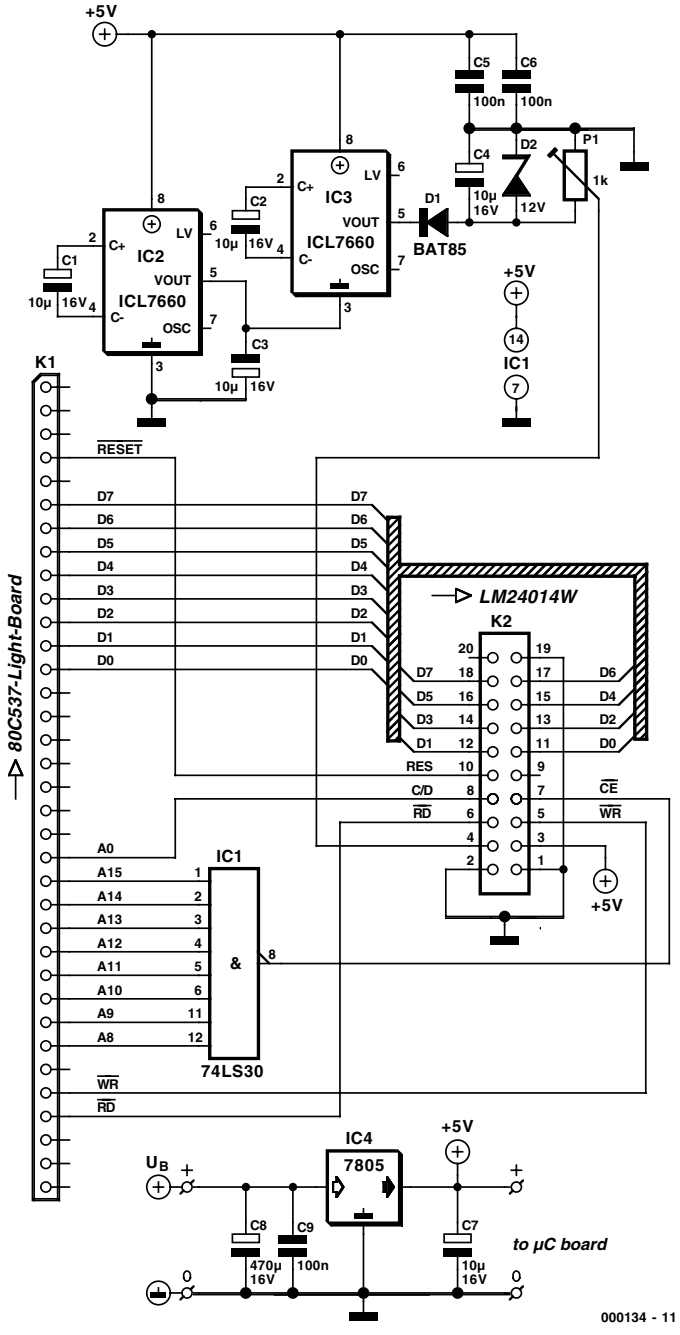


Figure 2. Adapter circuit for connecting the display to the 80C537 board.

plays, and pin 19 allows you to choose a character matrix with either 6 × 8 or 8 × 10 pixels. The displays works on +5 V, but for contrast adjustment it needs a relatively large supplementary negative voltage (between –12 V and –6 V, according to the data sheet). Many more modern displays can manage with one diode voltage drop below ground level or even 0 V. Check the data sheet if you use a different display module.

The display is connected to the 80C537 board via the simple adapter circuit shown in **Figure 2**. The data bus of the display is connected directly to the data bus of the microcontroller board. IC1 is an address decoder that allows the display to be addressed at 0FFxxh. The decoder output corresponding to this address is connected to the $\overline{CE}$ (Chip Enable) line of the LCD module. Address line A0 is connected to C/$\overline{D}$, which is the control line that looks after the selection of the internal command or data register of the display. This means that the data register can be read or written at address 0FF00h, while the command register is located at 0FF01h. The Reset, $\overline{RD}$ en $\overline{WR}$ (Read and Write) lines of the display are connected to the signal lines with the same names on the microcontroller board. For completeness, the interface board is also fitted with a 7805 voltage regulator to provide the operating voltage to the display. This voltage can also be used to power the processor board via pins 5 and 9 (GND and +5 V, respectively) of K1 on the printed circuit board.

The printed circuit board layout for the interface circuit is shown in **Figure 3**. Since this board is available ready-made from Readers Services, assembling the circuit should not present any problems.

Driving the display using software

The flowchart shown in **Figure 4** shows the sequence in which the data must reach the display. Naturally, the first thing that must be done is to switch on the power and allow the display to receive a power-on reset. As can be seen, in our case this signal is provided by the Reset signal from the microcontroller board.

Before the display module is ready to receive images, it is necessary to make a few parameter settings. This involves (in arbitrary order) defining the width of the graphic area on the display, specifying the starting address of the graphic area, selecting the display mode and enabling the graphic display mode via software. In contrast to an alphanumeric display, therefore, there is not any true initialisation. The parameters settings that we make here can certainly be modified later on.

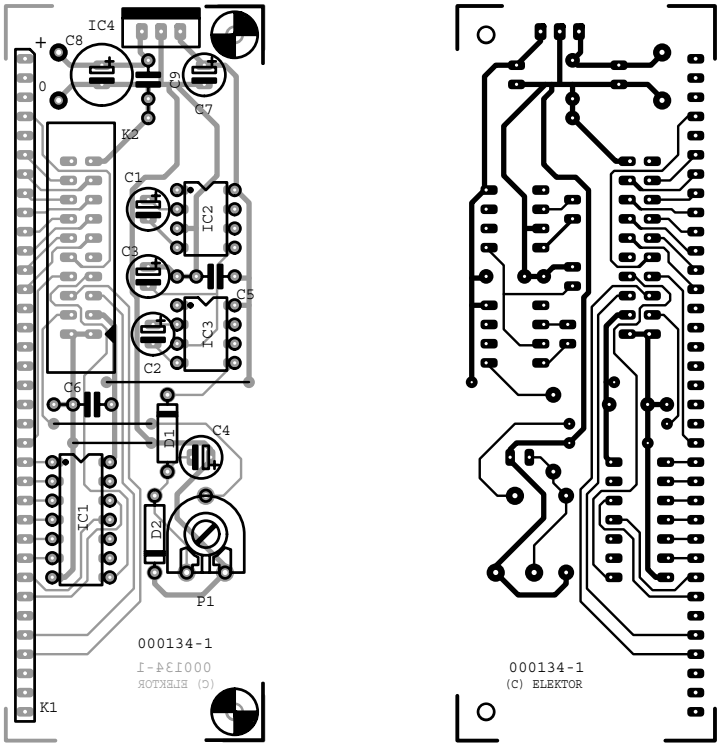


Figure 3. Printed circuit board track layout and component layout for the circuit of Figure 2.

Graphic Area

The width of the graphic area (GA) depends on the level at pin 19 of the display module, which selects a 6 × 8 or 8 × 8 character matrix. As will be seen, the screen is filled from the top left to the bottom right, always using strips that are 1 pixel high and (in our case) 8 pixels wide. If we select 6 × 8 (pin 19 at +5 V), the data bytes that we send to the display are actually only six pixels, since the two most significant bits will simply be ignored. With 8 × 8 (pin 19 at 0 V), the two uppermost bits are shown on the display. Our display is 240 pixels wide, so at 8 × 8 the width of the graphic area is 240 ÷ 8 = 30 (hexadecimal 0E1h), and at 6 × 8 it is 240 ÷ 6 = 40 (hexadecimal 028h).

Graphic Home Address

The memory in a display module is generally larger that what can be shown on the display. Consequently, we have to specify which memory address will be displayed at the top left corner of the screen. This is called the Graphic Home Address (GHA). The content of this memory location contains the first eight pixels (small horizontal strip) on the display. The next address contains the eight pixels located immediately to the right of the first eight pixels. In our case, the second line of the display begins thirty addresses past the starting address, and each successive *n*th line starts at the address GHA + *n* (GA). After an image has been placed in the graphic memory, we can cause it to scroll vertically over the display by increasing or decreasing the value of GHA by integer multiples of GA.

Display Mode

Here, in addition to things such as switching the curser on and off, we can specify what the display will do if text pixels and image pixels overlap. In the OR mode, a pixel will be switched on if it is driven either as text or as an image pixel. The other modes are AND (the pixel is on if it is driven in both forms) and XOR (the pixel is on only if it is driven in only of the two forms).

Data for the graphic display

After these settings have been made, data can be written to the

COMPONENTS LIST

Resistor:
P1 = 1kΩ preset H

Capacitors:
C1-C4,C7 = 10µF 16V
C5,C6,C9 = 100nF
C8 = 470µF 16V

Semiconductors:
D1 = BAT85
IC1 = 74LS30
IC2,IC3 = ICL7660
IC4 = 7805

Miscellaneous:
K1 = 35-way SIL connector
K2 = 20-way boxheader
PCB order code **000134-1** (see Readers Services pages)
Disk, project software, order code **000134-11** (see Readers Services pages)

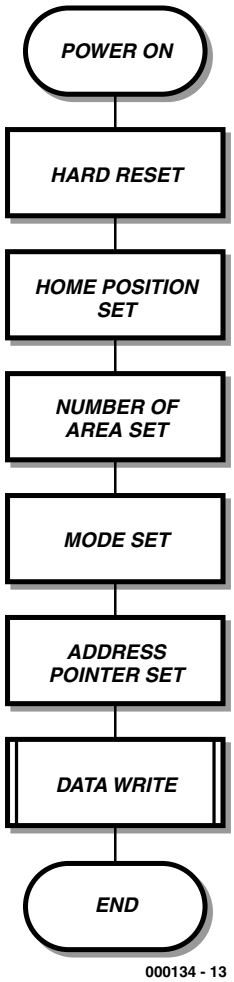


Figure 4. This flow chart shows the order in which the data should arrive at the display.

graphic memory of the LCD module. With the GHA setting, we have already specified which memory address contains that data that will appear at the top left of the display (the first horizontal strip of 8 pixels). We can address each byte on the display by setting the Address Pointer of the display to the value corresponding to the associated address and then sending a data byte to the module. However, it is highly cumbersome to always have to first send two bytes for the address followed by the data byte in order to address eight pixels, which is why the display also has an Auto Write mode. This requires the starting address to be supplied only one time, followed by all the data bytes. The display itself then looks after incrementing (or decrementing!) the Address Pointer after processing each individual byte. Following the final data byte, this mode must be switched off.
Note that it is also possible to alter a single pixel. This is due to the fact that the graphic memory can also be read out, We can thus read the byte containing the pixel in question, switch that bit on or off using AND or OR instructions and then write the modified byte back to the same address in the display module (naturally, it is also possible to change several bits at the same time).

Communicating with the display is thus not all that complicated, but how do you translate an image into bytes? In the old days, this was done with grid paper by first colouring in individual squares to form the image in large and then repeatedly counting off blocks of eight squares to form bytes (with a black square being a '1' and a white one a '0'). This is the technique that we will use, but now with the aid of modern tools!

From pictures to bytes

We will use the PC to prepare an image for reproduction on the graphic display. The picture can be generated in Paint (the standard drawing program provided with Windows), or an existing image can be enlarged or reduced so that it nicely fits on the display and then stored in bit-map format (.BMP). Next, we use an application written in Delphi (version 2.0) to translate the pixels in the image into data bytes for the display. These bytes are automatically added to an assembly language listing, which in turn must be translated into an Intel HEX file using an 8051 assembler. A terminal emulator program is then used to load this file into the program memory of the 80C537 board via the serial port, and the application is started. Following this, we can admire the result of our efforts and artistic inspiration on the graphic display.

Making pictures with Paint

In order to make an image (or make it suitable) for reproduction on the display, first start Paint (in the Start menu under Accessories). The click on the Images/Attributes menu and make the following settings (see **Figure 5**):

Units:	pixels
Colors:	black and white
Width:	width of the display in pixels (in our case, 240)
Height:	height of the display in pixels (in our case, 64)

After this, you can go to work with the drawing tools and make a new image, or you can take an existing image in any desired graphic format (BMP, GIF etc.) that is supported by Paint and enlarge or shrink it so that it nicely fits in the window. When closing the file, you must make sure that it is in fact saved as a monochrome bitmap!

Converting the bitmap

In addition to the image, BMP files contain a lot of other information (such as the number of colours, dimensions and so on) that are not necessary for our application. Here we will not devote any more attention to this 'extra' information, but will just use Delphi to extract

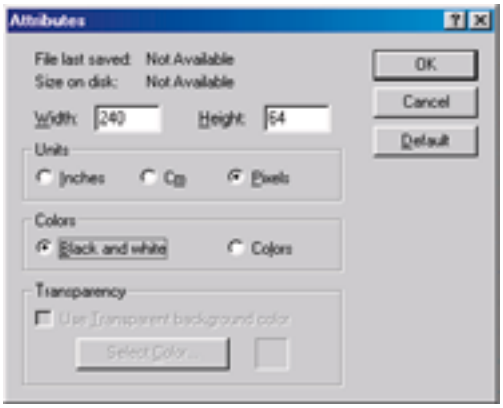


Figure 5. These settings should be made when starting the Paint program.

the pure pixel data from the bitmap file. In Delphi, we make use of a 'canvas' component, which is a sort of backdrop onto which a picture can be placed. The program looks after converting the BMP file to a graphic presentation, and the canvas is then a two-dimensional array of pixels: the modern version of our old-fashioned grid paper. The program is not dependent on the on the dimensions of the image. The canvas component automatically adapts itself to the size of the image, so the program does not have to be modified to be used with other aspect ratios or a different model of display.

Using a double FOR loop, we examine each pixel in turn to see whether it is black or white. We always put eight pixels at a time into each byte, since that is the representation that we will need later on for our graphic module.

The Delphi program directly opens the assembler file TEST.A51 and pastes the data for the display in this file using DB (Define Byte) instructions. The picture is stored in the program memory of the target system as a table, which allows it to be quickly reproduced on the display.

Reproducing the picture

Following the previous steps, all that is left is the assembly-language program that looks after the reproduction of the image on the display. This program is fairly simple. First, logical names are assigned to the addresses of three registers in the 8051 and the data and command registers of the display. Next, the display parameters are set (GHA, GA, Mode Set, Address Pointer Set) and the display is put into the Auto Data Write mode, so that the address pointer is automatically incremented after each data byte is received. We the see the same structure steadily reappear: first (if necessary) data is sent to the data register of the display, and then an instruction is sent to the command register so that the LCD module knows what should be done with the data. Note in particular the status check **before** each write instruction!

Next comes the part we've all been waiting for: writing the data into the graphic memory of the display. The

Assembly code listing

```
ORG 0100H
; Definition of special function registers in the 8051
; data pointer DPTR, 16 bit register represented by DPL and DPH
DPL  EQU 082H
DPH  EQU 083H
; accumulator
ACC  EQU 0E0H

; two registers of the graphic module: data and command register
data  EQU 0FF00H
command EQU data+1


   ACALL chk01    ; check if the display is ready to receive data

; the Graphic Home Address will be set. First we'll send the LSB of
; this address (0H)
   MOV DPTR,#data   ; DPTR points to LCD data register
   MOV A,#0H
   MOVX @DPTR,A

   ACALL chk01    ; check if the display is ready to receive data
; then the MSB (0H)
   MOV DPTR,#data
   MOV A,#0H
   MOVX @DPTR,A
   ACALL chk01    ; check if the display is ready to receive command
; and instruction Graphic Home Address Set (42H)
   MOV DPTR,#command
   MOV A,#42H
   MOVX @DPTR,A
   ACALL chk01    ; check if the display is ready to receive data

; The following instructions will set the Graphic Area (GA). The display is
; 240 pixels wide, 240/8 = 30 = 001EH
   MOV DPTR,#data
   MOV A,#1EH    ; LSB of GA
   MOVX @DPTR,A
   ACALL chk01


   MOV DPTR,#data
   MOV A,#0     ; MSB of GA
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#43H    ; Graphic Area Set = instruction 43H
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#80H    ; Mode Set: logical OR
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#98H    ; Display Mode Set: text off, graphics on
   MOVX @DPTR,A
   ACALL chk01

; The following instructions will set the LCD graphic Address Pointer to
; address 00H.
   MOV DPTR,#data
   MOV A,#00H    ; LSB of the Address Pointer
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#data
   MOV A,#0     ; MSB of the Address Pointer
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#24H    ; Address Pointer Set = instruction 24H
```

```
   MOVX @DPTR,A
   ACALL chk01

; The display is put into Auto Increment mode, i.e. the module itself will
; increment the address pointer after receiving a data byte
   MOV DPTR,#command
   MOV A,#0B0H    ; Data Auto Write Set  = instruction 0B0H
   MOVX @DPTR,A
   ACALL chk01


; Write data to LCD module

; First read the end of the data block, LSB is stored in R0, MSB in R1
   MOV DPTR,#enddata
   MOV R0,DPL
   MOV R1,DPH
   MOV DPTR,#block                  ; start of data block
   MOV ACC,#0
loop  MOVC A, @A+DPTR               ; read data byte from program memory
   PUSH ACC
; check if the last byte has been sent
   MOV A,R1
   CJNE A,DPH,next
   MOV A,R0
   CJNE A,DPL,next
; end of data block, jump out, first turn Auto Write off
   MOV DPTR,#command
   MOV ACC,#0B2H           ; Auto Mode Reset = instruction 0B2H
   LJMP stop
next  POP ACC
   PUSH DPH
   PUSH DPL
   PUSH ACC
   ACALL check2            ; check display status, ready to receive next byte?
   POP ACC

   MOV DPTR,#data          ; address of LCD data register
   MOVX @DPTR,A            ; write byte to display module
   POP DPL
   POP DPH

   INC DPTR                ; points to next byte in data block
   MOV ACC,#0
   SJMP loop

; This subroutine check if the display is ready to receive new data or a new
instruction
chk01

   MOV DPTR,#command
   MOVX A,@DPTR
   ANL A,#3
   CJNE A,#3,chk01
   RET

; This subroutine check if the display is ready to receive new data in Auto Write
mode
check2

   MOV DPTR,#command
   MOVX A,@DPTR
   ANL A,#8
   CJNE A,#8,check2

   RET
  ORG 200H
block
  DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
   .
;continued for several lines. By coincidence, the first and last bytes are all 0.
   .
  DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
enddata
stop  SJMP stop
  END
```

block of data starting at the label 'block' was added to the listing by the Delphi program. The label 'enddata' marks the address of the first byte following this block. With this technique, we make the assembly-language program independent of the size of the display. You should also bear in mind that the image could certainly be larger than the physical dimensions of the display, since the graphic memory is larger than the display area (one critical comment: the size of the graphic memory is not stated anywhere in the LM24014 data sheet!). It would certainly be possible to write some extra lines into the memory and then have the image scroll over the display by incrementing the Graphic Home Address by multiples of the Graphics Area (GA).

If a display with different dimensions is used, the value of GA in the assembler listing must be modified accordingly.

Note: there are also displays in which the addresses do not progress linearly. For example, we have a Toshiba display for which there is a jump in the memory address halfway through the display! For such a display, the assembly-language program would have to be modified.

And finally…

This source code file must next be assembled using the 8051 assembler of your choice; one possible candidate is the assembler for the 80C535 programming course published in the January–April 1994 issues of *Elektor Electronics* and available from Readers Services under order number **1811**. After this file has been translated into a standard Intel HEX file, it can be sent to the 80C537 board via the serial port with the aid of any desired terminal emulator program (such as HyperTerminal, which is provided with Windows). The required settings for the serial port are: 9600 baud, 8 bits, 1 stop bit and no flow control. Finally, the application can be started, following which we at last have a chance to admire the image on the display that cost us so much blood, sweat and tears to produce.

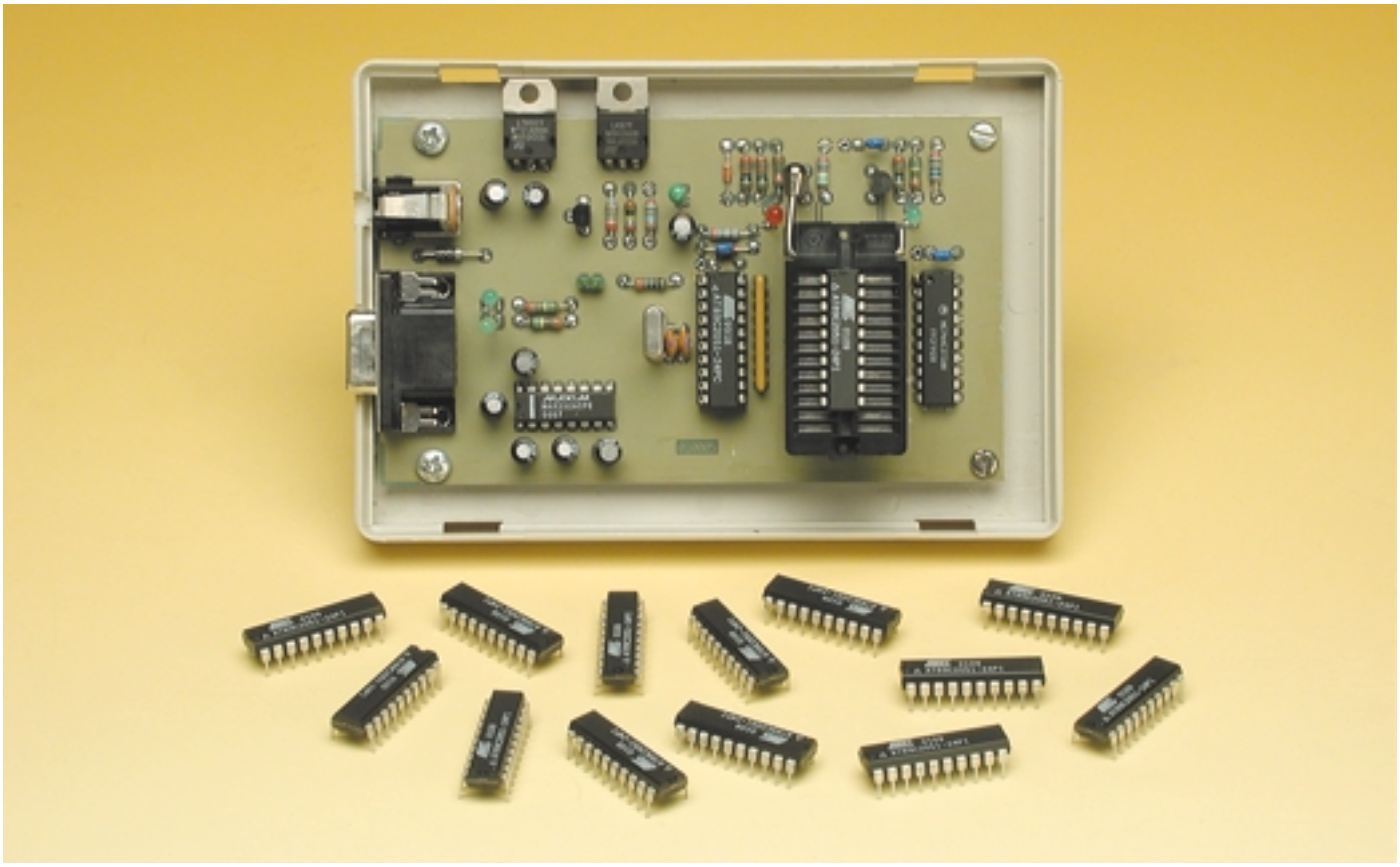
(000134-1)

Atmel AVR Micro Programmer

for 89Cx051 with up to 4k of program memory

Design by A. Oyrer

This programmer is suitable for programming the three popular 20-pin Atmel processors, the 89C1051, 89C2051 and the new 89C4051 with 4 k of program memory.



There is a new arrival in the Atmel family of powerful yet inexpensive MCS51-compatible microcontrollers with flash memory (PEROM): the 89C4051 boasts 4 kBytes of program memory. Besides the program memory capacity all members of the family have the following features:

– 128 bytes RAM
– 15 I/O port bits
– 2 16-bit timer/counters
– Interrupt architecture with two priorities and five vectors
– Programmable full-duplex serial port
– Precision analogue comparator

– Integrated clock/oscillator circuit

The AT89Cx051 family operates at a clock frequency between 0 (static) and 24 MHz and features two software-selectable power-saving modes: *Idle* (CPU only stopped) and *Power down* (RAM contents are preserved, but all functions are disabled).

Programming the flash

The programming characteristics of the flash memory are of course crucial to the design of a programmer. When the Atmel microcontroller leaves the factory the program memory array is filled with FFs and is ready to be written, byte by byte. Once the array is programmed, a byte that has been written can only be rewritten after first erasing the entire memory. The AT89Cx051 includes a PEROM **address counter**, which is reset to 00H on a positive-going edge of RST and incremented on each clock pulse on XTAL1.
The logical values on port pins P3.1 to P3.7 determine the programming mode, as shown in **Table 1**.

In order to write a program into the PEROM, Atmel recommends the following **programming algorithm**.

1. Power up by applying the supply voltage to VCC and GND, and hold RST/VPP and XTAL1 to ground.

2. Take RST/VPP and P3.2 high.

3. Set port pins P3.3 through P3.7 according to the table.

To write and read programming data the next steps are as follows:

4. Apply the first byte to be programmed (into memory address 00H) to pins P1.0 through P1.7.

5. Take RST to 12 V to being the programming process.

6. Apply a negative-going pulse, with a duration of between 1 and 100 µs, to P3.2 in order to write the byte into the PEROM array or into the lock bits. The write cycle completes automatically after about 1.2 ms.

Flash memory programming modes

Modus	RST/VPP	P3.2/$\overline{\text{PROG}}$	P3.3	P3.4	P3.5	P3.7
Write program data	12 V	1.2 ms negative-going pulse	L	H	H	H
Read program data	H	H	L	L	H	H
Write lockbit 1	12 V	1.2 ms negative-going pulse	H	H	H	H
Write lockbit 2	12 V	1.2 ms negative-going pulse	H	H	L	L
Erase memory	12 V	10 ms negative-going pulse	H	L	L	L
Read signature bits	H	H	L	L	L	L

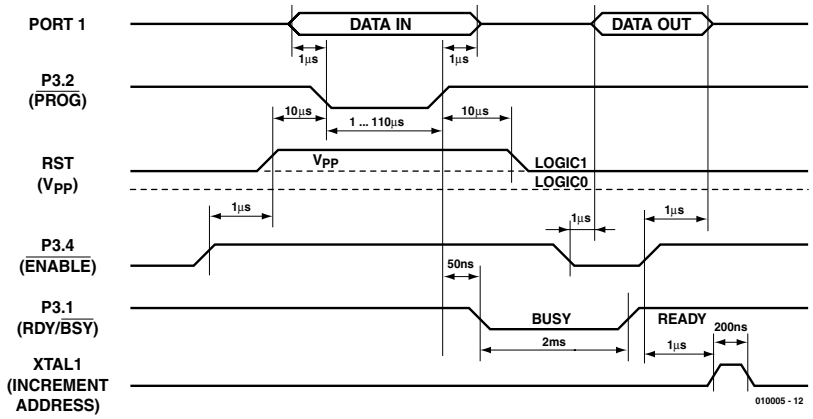


Figure 1. Timing diagram for a write/read cycle.

7. To read data, the voltage on RST is taken down to a normal logic high level and port pins P3.3 are set to the appropriate levels. The byte stored in the current address appears on port P1.

8. When the byte has been written or read, the address counter is incremented by a rising edge on XTAL1. The new byte is put on P1.

9. Steps 5 to 8 are repeated until the whole memory is programmed or until the end of the programming data is reached.

10. To finish the programming process, set XTAL1 and RST low and remove the supply voltage from VCC and GND.

A complete write/read cycle is illustrated in **Figure 1**. Port pin P3.1, which signals **READY/$\overline{\text{BUSY}}$**, is not shown. The data sheet gives the duration of the programming pulse at between 1 and 100 µs. Since a programming cycle lasts about 1.2 ms and it is not permitted to begin a new cycle before the end of the previous one, pin P3.1 is taken low (BUSY) for a maximum of 2 ms, while writing is taking place. Only when P3.1 returns high (READY) can a new write cycle be initiated.

As can be seen from the timing diagram, **verification** of the write process can be carried out either byte by byte during the BUSY period during write (compare) or all in one go, as long as the lock bits have not been set (read). The latter is carried out as follows:

1. Take RST from low to high to reset the internal address counter to 000H.

2. Set up the appropriate control signals on pins P3.3 through P3.7 and the data byte appears on P1.

3. Apply a pulse to XTAL1 to increment the address counter by one place, and the next data byte appears on P1.

4. Repeat step 3 until the entire memory is read.

The **lock bits** cannot be directly verified, but they can be identified by the effect they produce.

The entire memory, along with the lock bits,

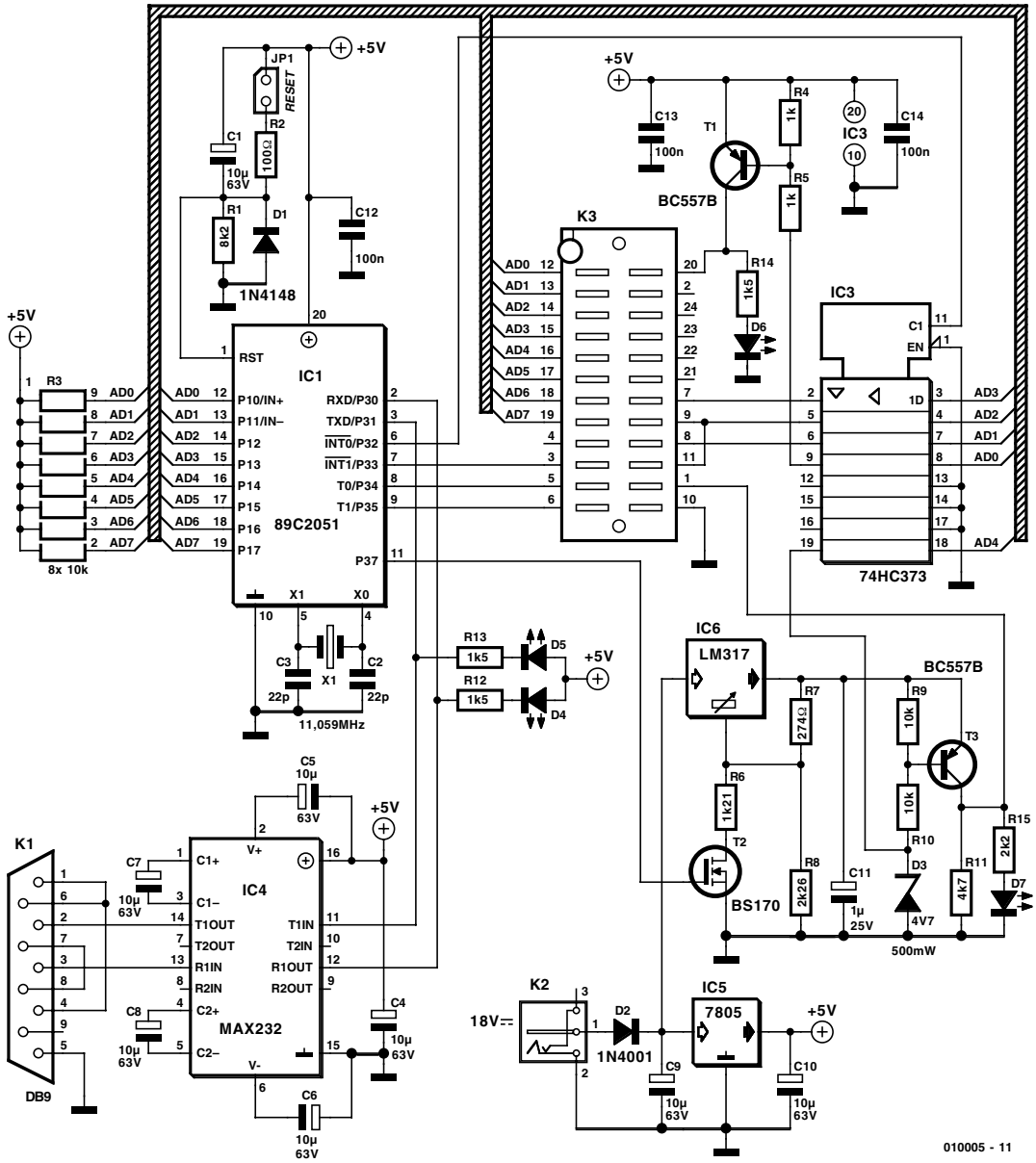


Figure 2. Circuit diagram of the Atmel Programmer with master controller and ZIF socket for the device to be programmed.

is electrically erased when the correct combination is applied to P3.3-P3.7, and P3.2 is held low for at least 10 ms. All memory locations are then filled with ones. This **chip erase** procedure is required before reprogramming an already-programmed memory or in order to erase the lock bits.

The signature bytes give manufacturer and device type information. They can be read in the same way as the program memory locations, but with P3.5 and P3.7 held low. Possible values are as follows:

000H 1EH: manufacturer code for Atmel
001H 11H: 89C1051
21H: 89C2051
41H: 89C4051

Programmer hardware

Since, apart from the size of the program memory, all the functions and features of the three microcontrollers are identical, the programmer is not complicated. It must simply read the signature bytes at the start of any operation to identify the type of microcontroller.

The programmer presented here implements all the programming modes described:

– Write
– Read
– Compare
– Program lock bits
– Erase
– Read signature

In write mode the signature bytes are checked to ensure that the amount of data to be programmed will fit in the processor. In read mode, the data file generated can be in Intel Hex or in binary format. Reading the signature bytes is optional and can if required (for example in the case of a faulty microcontroller) be disabled via the menu. In this case the programmer assumes that it is dealing with an 89C4051.

As can be seen from **Figure 2**, the programmer hardware is relatively straightforward and consists essentially of a master microcontroller, which needs to contain just 865 bytes of program code. IC1, naturally enough an 89C2051, carries out everything required for programming: it sets up the programming mode requested by the user, converts into parallel form the program code sent from the PC over the RS232 interface, sends the contents of an already-programmed memory (and other information) back to the PC, and switches the voltages (0 V, 5 V and 12 V) as required by the timing diagrams. However, even with the best will in the world, all this is not so easy to achieve with just the 15 I/O port bits offered by the 89C2051.

Communication with the PC is handled via port pins P3.0 and P3.1. Data transfer is indicated by LEDs D4 and D5 flashing. Port pin P3.3 (interrupt 1) is used as an input to monitor the BUSY/READY signal of the microcontroller being programmed ('processor under programming', or PUP). The individual programming pulses are provided by a further port pin, P3.5, while P3.4 is used to provide the edges on XTAL1 to increment the address counter of the PUP.

Now, the remaining ten port pins must provide five control signals and eight data bits, and here some ingenuity is required. The whole of port 1 of IC1 is connected to its counterpart on the PUP, but the five port pins P1.0-P1.4 are also connected to IC3, an 8-bit latch type 74HC373. The microcontroller first applies the control signals to the five latch inputs. The latch enable input (pin 11) is taken high and so the latches are transparent: control signal outputs AD1, AD2, AD3 go directly to the PUP (as can be seen from Table 1, P3.5 and P3.7 always have the same value), AD0 turns on the +5 V supply to the microcontroller via T1 (indicated by LED D6), and AD4 controls the application of the programming voltage to RST/VPP. Once these values are set up, the latch enable input is taken low again, and the values are stored.

The Atmel programmer uses an adjustable voltage regulator to provide the voltage on RST/VPP. For reading, when 5 V is required, FET T2 conducts and the two resistors R6 and R8 are effectively in parallel. If 12 V is required, the FET is turned off and only R8 is in circuit. The BS170 is switched by a separate port pin of IC1, P3.7.

In order that the voltage on VPP/RST is at ground potential after the programming or reading process (and so the PUP can safely be removed from its socket) the LM317 is first switched to 5 V. Then AD4 (from the latch) is set, to make the potential at the junction of R10 and D3 equal to 5 V. This turns off transistor T3 and R11 pulls RST/VPP to ground.

This accounts for all the port pins of the 89Cx051.

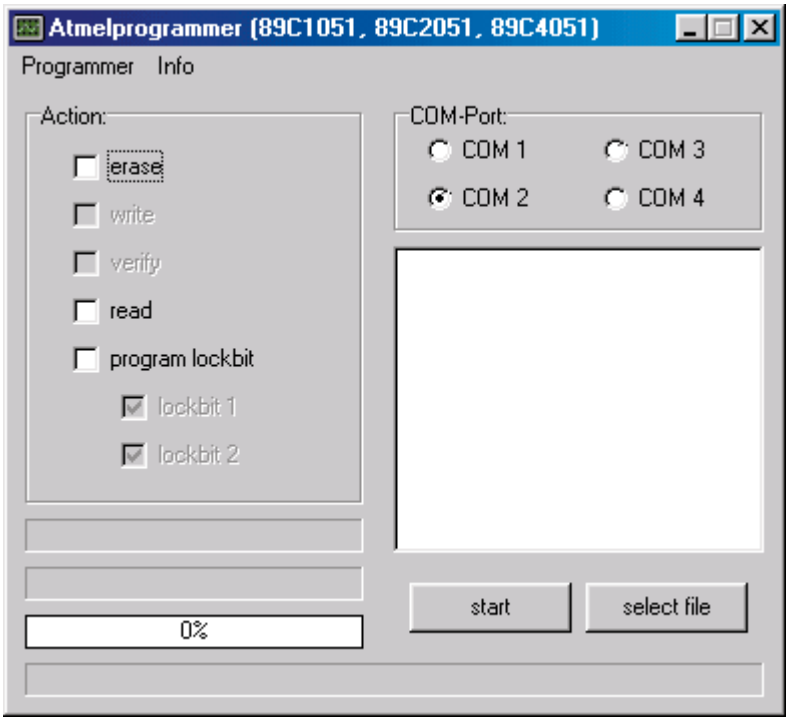


Figure 3. Control panel for the PC program *Programmer_eng.exe*.

Software

The companion diskette for this project, order code **010005-11**, which is available from Readers' Services, includes an assembler program called *Atmelpr.asm* for the master microcontroller, as well as the microcontroller object code in the form of a *Atmelpr.hex* file. Ready-programmed microcontrollers are also available (order code **010005-41**). This software is complemented by

Programmer_eng.exe,

a program written in Delphi which controls all the operations of the Atmel programmer such as reading, writing, and erasing. The Delphi source code *source_eng.zip* is also available.

The printed circuit board layout is available for free download from our Internet site www.elektor-electronics.co.uk. There is no download for the above mentioned software because of contractual agreements with the authors.

Serial port COM2 is used by default. When the program starts up a window appears, divided into two (**Figure 3**). On the left the operations are selected from a menu while on the right the operations are confirmed, for example as follows:

```
reading signature byte  ok
programming lock bit 1  ok
error: file too large!
```

and so on.

If *Read* is selected, all other operations are disabled and a window appears to allow the user to choose the name of the file in which the contents of the processor's memory are to be written out. If the file extension is given as *hex*, then an Intel Hex file is automatically generated; if *bin* is specified, a binary file is read.

The *Select file* button allows the user to choose the name of the file that will be written. Here also **hex** and **bin** are possible.

The file size is shown below the microcontroller type. Below that a bar indicates progress of the individual actions. In the area

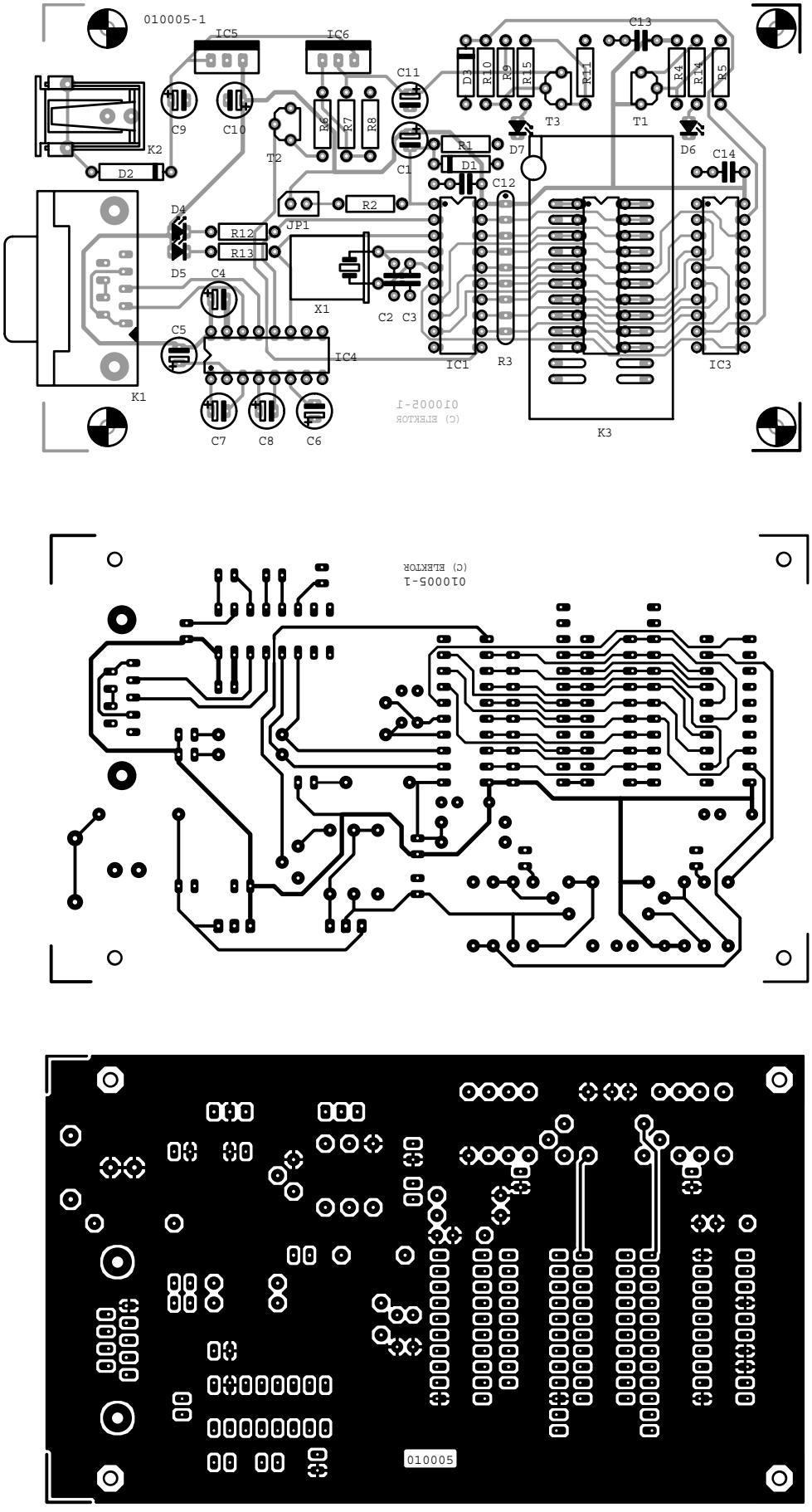


Figure 4. Layout and component mounting plan for the double-sided printed circuit board.

COMPONENTS LIST

Resistors:
R1 = 8kΩ2
R2 = 100Ω
R3 = 8-way 10kΩ SIL array
R4,R5 = 1kΩ
R6 = 1210Ω
R7 = 274Ω
R8 = 2260Ω
R9,R10 = 10kΩ
R11 = 4kΩ7
R12,R13,R14 = 1kΩ5
R15 = 2kΩ2

Capacitors:
C1,C4-C10 = 10µF 63V radial
C2,C3 = 22pF
C11 = 1µF 25V radial
C12,C13,C14 = 100nF

Semiconductors:
D1 = 1N4148
D2 = 1N4001
D3 = zener diode 4V7, 500 mW
D4,D5,D6 = LED, green, high efficiency
T1,T3 = BC557B
T2 = BS170
IC1 = AT89C2051-12PC (order code **010005-41**)*
IC2 (K3) = 24-way zero-insertion force (ZIF) socket
IC3 = 74HC373
IC4 = MAX232
IC5 = 7805
IC6 = LM317T

Miscellaneous:
JP1 = jumper
K1 = 9-way Sub-D socket (female), PCB mount, angled pins
K2 = mains adaptor socket
X1 = 11.0592MHz quartz crystal
Enclosure, size 137x95x25 mm (e.g., Pactec type WM46)
PCB, order code **010005-1***
Disk, project software, order code **010005-11***

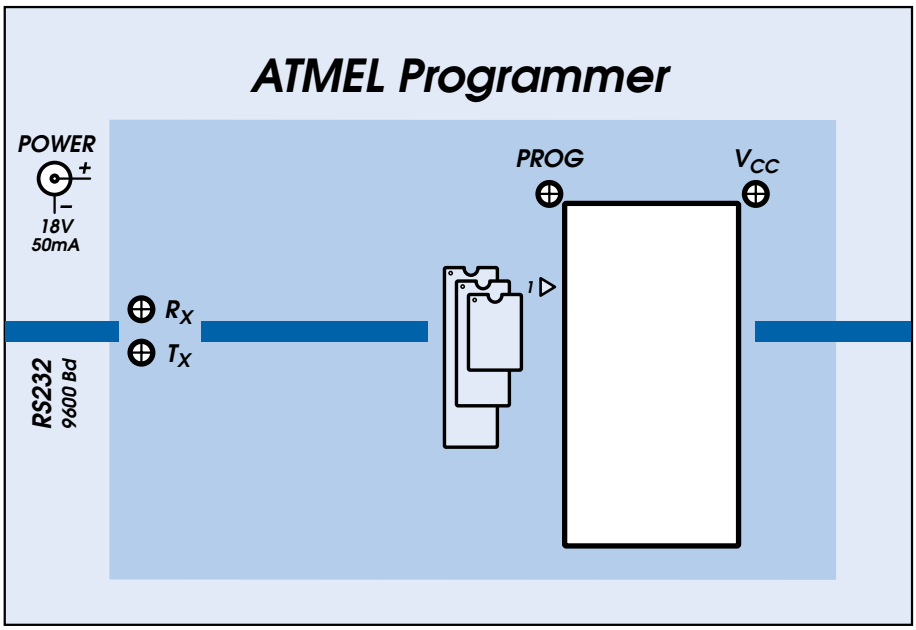
* (see Readers Services page or www.elektor-electronics.co.uk)

at the bottom the filename and path are shown.

In the menu under *Read signature byte* the user can choose whether the signature bytes are initially read or not. This option should always be activated unless the programmer is unable to read the signature bytes. The information in the signature bytes is shown below the microcon-

troller type select box. The *Write, Read* and *Verify* commands require this information, since otherwise they assume that the device to be programmed is an 89C4051.

If the programmer is not connected, the error message *Check programmer or COM port* appears. Check the serial connections (RxD, TxD and GND) are connected correctly and not crossed over, and check that the programmer is switched on. An indication is given if the file is too large for the processor to be programmed. An error message also appears if an error is detected in the input file.



010005-F

Figure 5. Front panel design for that professional look.

Construction and adjustment

The Atmel programmer is constructed on a small double-sided printed circuit board. The layout and component mounting plan are shown in **Figure 4**. The programming socket needs to stand a little higher than the other components so that it can protrude through the lid of the enclosure and the PUP can easily be removed. This is most easily achieved using a wire-wrap socket, and enough space has been allowed on the circuit board for such a socket.

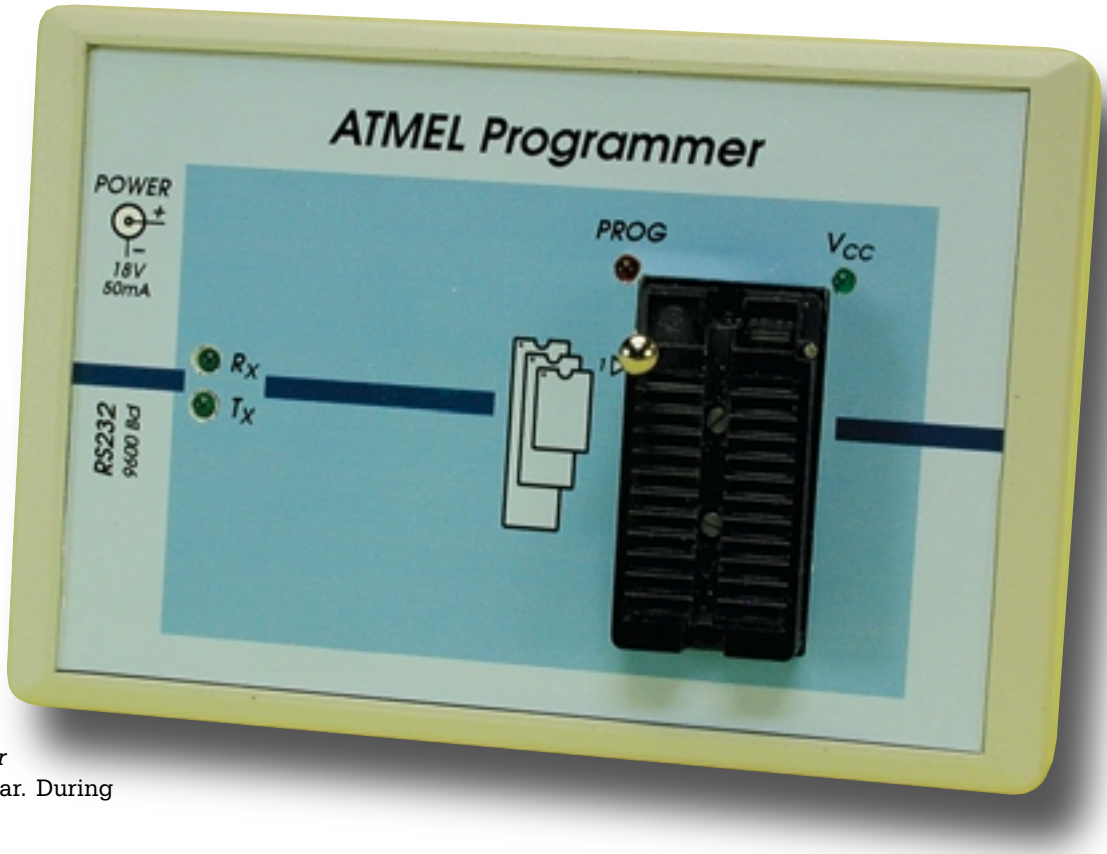
The 9-pin sub-D socket fits comfortably in the recommended enclosure, but the power supply jack projects a little and must be filed down.

When the circuit board has been assembled, checked and fitted into the enclosure suggested in the parts list (onto which the front panel shown in **Figure 5** fits snugly), only one thing remains before programming your first device: checking the programming voltage.

To do this, connect the programmer to the PC and choose any desired file to write. The *Read Signature Byte* option must be deactivated, since otherwise the *No Processor* error message will appear. During the write process the RST pin is switched to 12 V. If this voltage is not correct, R8 can be increased slightly (which will increase the voltage) or reduced slightly (to reduce the voltage).

In exactly the same way the 5 V supply can be adjusted via R6. The *Read File* option should be selected, and the resistance in parallel with R6 adjusted appropriately. This completes the adjustments, and you are ready to program your first microcontroller.

(010005-1)



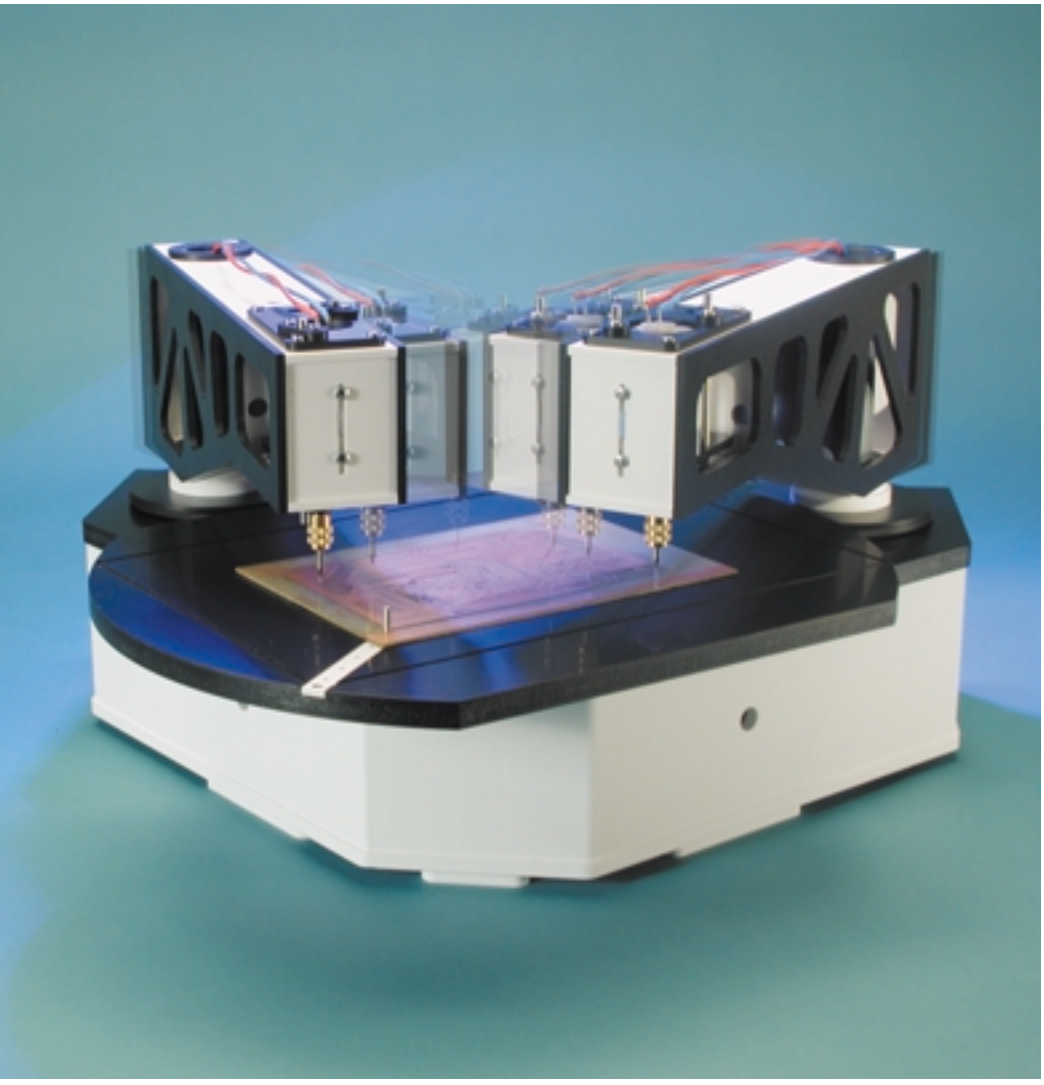
PCB Drilling Machine (5)

part 5 (final): ready to go!

Design by T. Müller (Radix GmbH)

www.radixgmbh.de

In this final part on the construction of the PCB drilling machine, all its functions are tested and the machine is calibrated. The machine is then ready for its first job.



By the time you read this, the first PCB drilling machines have already been delivered — and perhaps also assembled. It is therefore high time that our description of the machine be completed. All that remains is the calibration of the machine. The machine must be calibrated before the driver software is able to determine the reference point, calculate drilling coordinates correctly, position the arm(s) and turntable and raise and lower the drilling head. A number of position sensing switches play a key role in initialising the machine. In this final part we describe a complete calibration procedure for the arms and turntable with the aid of the TanBoTest software, and the use of the TanBoDrive driver software for drilling PCBs. Please note that the German working title of the project, 'Tanbo' (for Tangential Bohrer) is retained in the names of the files and programs developed for the project.

Position sensing swiches

In the machine enclosure there is an optical switch situated just between the two drive shafts for the tool arms, which operates on the reflective principle. **The position of the turntable** and of tool arms 1

and 2 can be determined using this switch. The sensor consists of a light-emitting diode (LED) which emits infrared light vertically upwards, and a phototransistor, also pointing upwards, which detects when a reflective object passes in front of the LED. For this reason there is a reflector made of metallised foil on the pointer attached to the tool arm drive shaft, as well as on the underside of the turntable.

This reflective light switch operates using unmodulated infrared light and is therefore highly prone to interference from other IR sources such as daylight, artificial light, and even the light from cigarette lighters! While under the circular part of the turntable the switch is completely covered, but the turntable's two flattened sides, which are opposite one another, can let stray light in. And why is the turntable not a perfect circle? It saves a little material, but more importantly this is the only way that access can be provided to the sensor to allow for inspection and cleaning.

If stray light should interfere with the sensor, the turntable need only turn until the sensor is again in the dark, and then continue to turn until the sensor sees light again **but only over a narrow angle**. In this way, the position of the reflective foil can be found. Upon subsequent rotation in the opposite direction the transition from light to dark must be found at the same point. Once the system has determined the position of the turntable, it then immediately knows where the flattened sides are that give rise to the risk of light interference: the software takes these into account.

The process for **initialising the tool arms** is simpler, since in each case the sensor is covered by the turntable and is therefore protected from stray light. The system looks for (and finds!) the reflecting position of the arm.

If two arms are in use simultaneously, there is a problem that must not be overlooked. If both arms attempt to travel to their end positions at the same time, there is a risk of collision. This difficulty is overcome elegantly as follows: the Windows software notes the position of the two arms at the end of a run and stores the relevant information in a temporary file. When the program is next run, the file is loaded, and so a collision can be avoided. If the program terminates abnormally (which can be the rule rather than the exception under Windows!), this file will be found to be missing the next time the program is run. In this case the program enters manual control mode. A graphical representation of the drilling machine appears on the monitor: the tool arms can be dragged into position on the screen using the mouse so as to represent the actual state of the machine. This need not be absolutely accurate, although the machine will rely on your input.

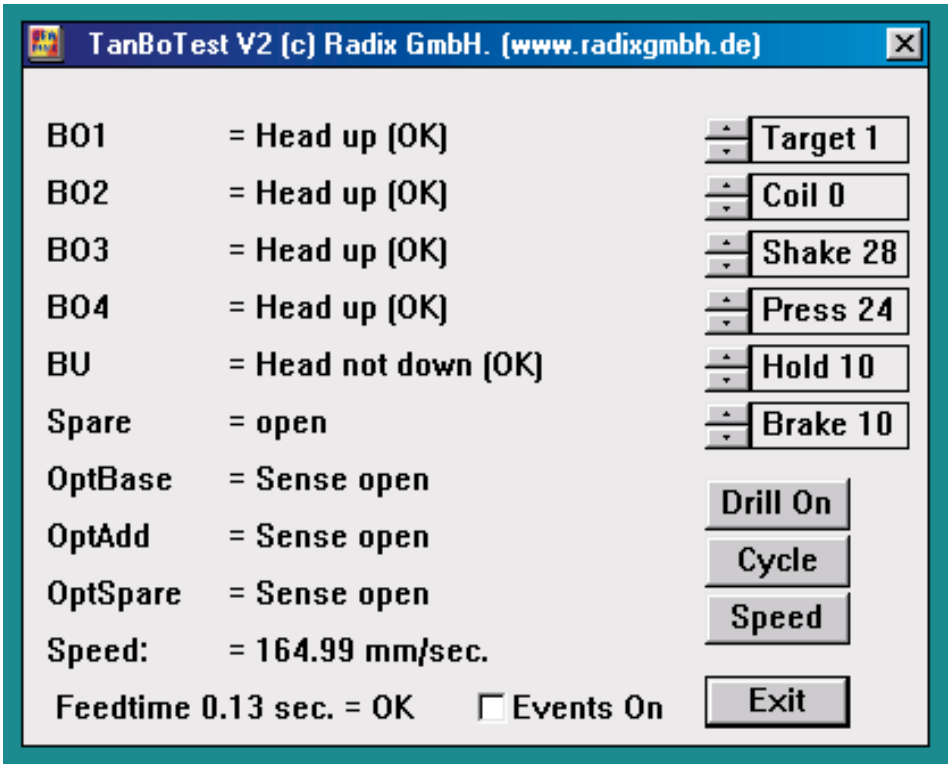


Figure 1. The TanBoTest utility tests the various microswitches and determines the operating parameters to be stored in the configuration file.

Two microswitches are provided in the tool arms to determine which of three possible states the drill head lift mechanism is in.

State 1: Drill head lift mechanism at the top of its travel, in rest position. If no current flows in the solenoid, the force of the spring built into the lower part of the arm must push the drill head to the top of its travel and thus open switch BO (BO is a normally-closed contact). If the arm is not fitted, there is no BO switch, and the corresponding switch inputs (BO1-BO4) on the circuit board are open. The software can check each arm individually to determine whether the corresponding drill lift mechanism is in the up position.

State 2: Drill head lift mechanism is lower position, determined via BU. All BU contacts of all arms are normally open and are connected in parallel. This can work because only one head is allowed to be in the down position at a time. Safety logic in the GAL prevents multiple drill head lift mechanisms from being actuated at once. And if (because of a mechanical failure) more than one drill head should be down at the same time, this can be detected via the BO contacts.

State 3: This state is between the other two: neither up nor down. Assuming nothing has gone wrong, this can only happen during motion of the drill head. If this state occurs while the head is not in motion, the drill lift guide is probably jammed.

During drilling the Windows software can measure the time between the closing of the upper switch and the closing of the lower switch, which gives the time taken to drill the hole. A broken drill can be detected by this

time being too short, and an overly worn drill can be detected by this time being too long. If it takes longer than five seconds to drill a hole, the controller automatically switches off the drill lift mechanism and the drill motor.

The first test

The *TanBoDrive* driver software requires that the drilling machine be fully functional, both electrically and mechanically. The input data — in this case drilling data in Excellon format — are specially processed and transformed into the required commands, in the form of motion commands and drilling commands, to be sent to the microcontroller. More than this *TanBoDrive* cannot do. Linear interpolation, as required for interpreting HPGL-format files, is included (otherwise the machine could not even move from hole to hole), but HPGL import is disabled and will only be enabled when the previously-mentioned milling arm has been thoroughly tested. *TanBoDrive* should be compared to a printer driver: if the controller board or the mechanics are not in order, the program simply gives up.

First, then, servicing and diagnostic software is required to test all the functions of the controller board and the connected units. The various switches are particularly important, since they must be correctly wired as well as correctly fixed. All components must first be made to work as intended and all tests must be passed before we can move on to drilling a circuit board.

This requirement is met by the *TanBoTest* test software (**Figure 1**). When the software is started a dialogue box appears showing the states of all the switches, updated every 300 ms. The optical switches can be tested with a finger or any reflective object. The drill motors can also be turned on and off, and the lift solenoids can be supplied with a selectable current, in order to help check (and, if necessary, improve) the smooth running of the guides. The program, freely available from this url

www.radixgmbh.de/deutsch/menu_bestellung.html

is self-explanatory and shows the states of all the switches. It is essential to check the messages in the dialogue box and to test the three positions of the drill head lift mechanism. Test the optical switches with a reflective surface and with stray light. Set *Target* to 1 and push the drill head down: you should see activity on BO1 and BU (but not BO2); likewise for the other target. Using the box marked *Coil*, adjust the solenoid current to find the value required to drive the drill head down. The numbering of the targets is clearly defined: looking at the machine from the longer side and above, the left-hand arm is tool arm 1 and the right-hand arm is tool arm 2.

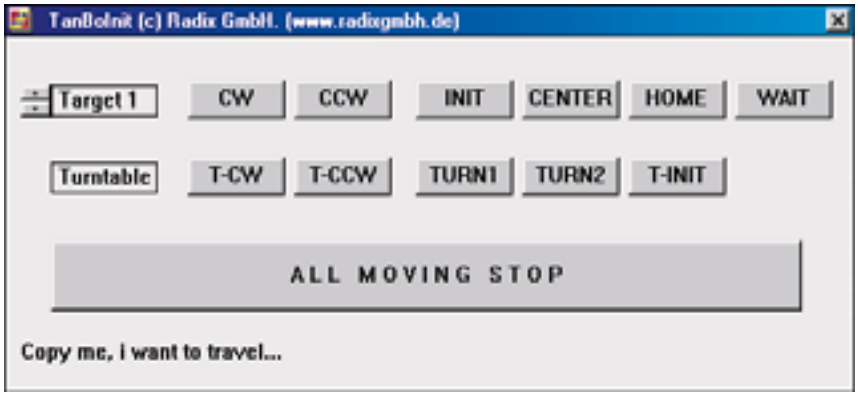


Figure 2. Initialisation program for calibrating the arms and turntable.

The switches are so vital to the operation of the machine that the switch inputs to the controller board are not fitted with a socket, but rather soldered directly. Although a little more tedious to assemble, the connection is more secure. A length of black PVC insulating tape should be used as a strain relief on the cable which comes from the drive shafts, so that the connecting pins are not under mechanical stress. Ensure also that the cable is free to move.

Here is a summary of all the functions of the test software:

The *Target* box selects between output drive stages 0-3 for drill and coil.

The *Coil* box can be set from 0-65 to control the solenoid current from 0 (off) to 65 (100 %) for the output stage selected under *Target*. Beware that the original solenoids are overloaded at the 100 % setting and cannot run continuously at this current. A duty cycle of about 30 % is possible. Continuous operation is possible at a setting of 26. The software switches the coil off if the setting is left unchanged for 5 s.

The *Drill On* button switches the output drive for the drill motor of the selected target on and off.

The next four boxes are to do with the drill cycle. Each target has its own press and brake values, while the shake and hold values apply to all four targets. Each click on the Cycle button initiates a drill cycle with the selected parameters.

The *Press* box sets the force with which the drill is pushed down on the circuit board. This value — to a certain extent — affects the speed of drilling. If the value is too small, the solenoid will not exert enough force to drive the drill down. If it is too big, the drill can in extreme circumstances be damaged when it hits the circuit board. The value should be set to suit the drill: a value between 24 and 40 is normal. If satisfactory operation can be obtained with a value below 25, it is a credit to the conscientiousness and care with which the assembly has been built: congratulations!

The *Brake* box sets the braking value needed to cause the drilling head to come to a gentle stop at the end of its upwards travel. If the value is too high, the head guide will rebound down before finally coming to a halt; if on the other hand the value is too low, the upper switch will be (ab)used as a mechanical endstop, which is definitely something to be avoided. Find the setting where the head does not rebound and then add one or two to the value.

The *Hold* box sets the time (in units of 52 ms) for which the drill remains down after the circuit board is drilled through. Too short a time leads to swarf not being fully cleared from around the hole on the underside of the board, while if the time is set too long, it is simply time wasted. The default value of 10 (i.e. 0.52 s) is a

good typical value for clean holes of all sizes.

The *Shake* box is aimed at the automatic clearing of faults. If for example the guides are very dirty and not running smoothly, the force of the spring may not be enough to lift the head fully up and operate the upper switch. This fault can be simulated by pushing the guide down a couple of millimetres so that the switch BO indicates *'Head not up'*. If the head stays in this position when it is carefully released, then the fault condition has been replicated. The shake value controls how the head is driven down a small amount and then released, without braking, so that it springs back. The force with which this is done must of course not be so great that the drill hits the circuit board when it is not running.

The *Cycle* button initiates a complete drilling cycle on the selected target. The feed time is measured and displayed (minus the Hold value). If the cycle completes without error, *OK* is displayed; otherwise, *Error* is displayed. If there is an error with a feed time of around 4.5 to 5 s, this indicates that the head never reached the lower position: this condition can be simulated by setting Press=0. With a feed time of around 0.5 to 1 s, the fault is that the upper end position was not reached: this condition can be simulated as described above with Shake=0.

The values established for the drill cycle using the *TanBoTest* program are recorded in the configuration file.

The *Speed* button sends data at a steadily increasing rate to the controller until the PC's processor can no longer keep up with the data stream and the controller's FIFO empties.

The *Speed* indication in mm/s refers to the speed of interpolation on two axes, for example to simultaneous motion of the turntable and the tool arm at the given speed. The mechanical maximum speed limit is around 80 mm/s. If the speed test gives better results then there is a surplus of processing power, and further axes can be made to move simultaneously. When trying the speed test, the stepper motors should not be connected!

The *EventsOn* check box, when checked, gives up some processor time during the speed test to the operating system, and hence to other processes. This is done to allow other tasks to make progress. The *TanBoDrive* control program uses a dynamic algorithm to determine how much processor time to allow the operating system, ensuring as necessary that the system does not come to a complete halt.

At start-up, the software must be informed of the address of the LPT port to which the controller is attached. This setting can be obtained from the Windows system information, and is generally 0378-037F$_{HEX}$. Open an ASCII text editor with an empty file and enter on the first line the first of these numbers (here 0378). Now press Return twice and save the file as *lptaddr.txt* in the same directory as *TanBoTest*. This file is processed by all the drilling machine programs and must be placed in the same directory as all those programs.

An important part of the test program is the measurement of the transfer rate for data streams over the Centronics interface. See the text box for further details.

Adjusting the arms

After you have checked the operation of the hardware — and, if necessary, corrected it — using the *TanBoTest* program, the machine must be adjusted and calibrated. Calibration — also known as setting the zero point — presents no difficulty with a linear machine, but how can we adjust something that is circular and does not have a start or an end? Further, several of the components of the drilling machine can be assembled in different ways: for example, the drilling axis can be moved to and fro by several millimetres before being firmly screwed down. In order to calculate the X/Y coordinates exactly, the length of the arm must be known absolutely precisely. It is all done, as we said in the first article in this series, without precision measuring instruments of any kind.

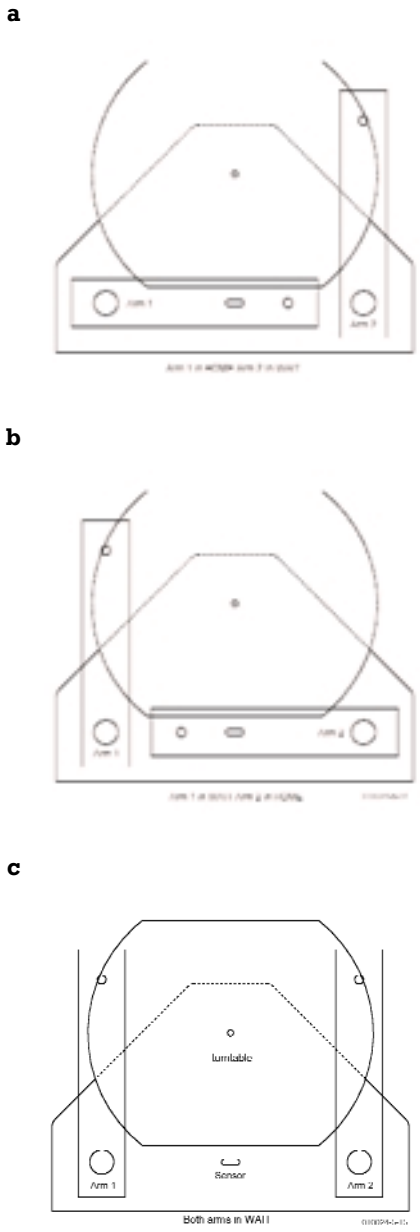


Figure 3. The arm positions for calibrating the turntable: home/wait (a), wait/home(b) and wait/wait (c).

All that is required for exact adjustment is the *TanBoInit* software **(Figure 2)**, which also uses the *lptaddr.txt* file. Note: if at any time during the adjustment procedure anything untoward should happen, or if you think from the way the arms are moving that a collision is imminent, click on the 'emergency brake' ALL MOVING STOP: the machine will come to an immediate halt.

First we calibrate the arms, and then the turntable. The other way round is not possible, because the turntable can only be set up

with properly calibrated arms. Each arm can be moved at will using the buttons CW (clockwise) and CCW (counterclockwise). In *TanBoInit*, as before, which of the arms 1-4 is moved can be set via the *Target box*.

First check that all the tool arms turn appropriately in response to the CW and CCW buttons. Select also the targets for which an arm is not fitted, and press CW and CCW. In these cases nothing should move.

The ordering of the target numbers has already been checked in the test program. The four BO switches are shown in a column above one another. If you push down the head on arm 1, you should see activity on BO1 and not on BO2. Check this very carefully, since otherwise the collision detection will not work. The drives are extremely powerful and are easily capable of destroying one another!

The turntable has its own CW and CCW buttons, while the *Target* box remains always assigned to a tool arm. Before an arm can be calibrated, the turntable must be rotated so that the reflective light switch is off.

Begin with the calibration of arm 1 (target 1). Move arm 2 (if fitted) to the WAIT position, as shown in **Figure 3a**. In the kit of parts which contained the brass chuck, you will find a short polished metal pin, 20 mm long, left over. This is the only calibration tool required. Push this pin as far as possible, without using excessive force, into the hole in the aluminium shaft at the centre of the

Buffer Underrun?

The drilling machine controller software forces Windows to operate in real time. Although Windows can output data very quickly, it cannot coordinate the timing of the output exactly, and for this reason the controller includes a FIFO (first in, first out) memory which allows the transmitted data items to be brought back into step with one another.

This works perfectly as long as we can ensure that the data stream from Windows never stalls for so long that the FIFO becomes completely empty. The FIFO has 35 slots, from which we can make the following example calculation.

Let us suppose that a stepper motor is to be turned at the rate of 500 steps per second. The motors used in the drilling machine have an angular resolution of 1.8°, and so this corresponds to a rotation rate of 2.5 rotations per second. Since the following gearbox has a ratio of 200:1, the output shaft turns through an angle of 4.5° per second. The circumference of the circle swept out by the machine is 1510.6 mm, and so the head moves at a speed of 18.88 mm/s.

What data rate is required for this? The specification of the Centronics interface allows for at least 20000 bytes/s. On an older PC (150 MHz Pentium) the bandwidth was measured at around 46000 bytes/s.

Each motor step requires exactly one transfer, and so the required 500 transfers per second occupy a mere 2.5 % of the available bandwidth. Bandwidth is therefore not a problem.

At 500 steps per seconds each step lasts 2 ms. The FIFO, with its 35 slots, can hold 70 ms of data. In these 70 ms the processor we used runs for 10.5 million cycles. If these cycles are consumed by Windows and the FIFO cannot be refilled with data in time, then there will be a break in the data stream.

For this reason it is not possible to run programs in parallel with the drilling machine that load the system heavily or which consume practically all its processing power.

How can we check that the data stream is in fact continuous? A control signal on the controller board indicates when the FIFO is empty.

This signal is taken to pin 12 of the Centronics interface (paper empty) and is asserted when all the bytes in the FIFO have been processed.

So why is it so important to have a continuous data stream? Stepper motors that are to be used at a high stepping rate must be brought gradually up to the desired target speed using a specially-designed ramp function. The moment of inertia of the rotor is so great that it cannot follow a rapid speed change and drops steps. This effect is much more noticeable when starting than when stopping, and depends on the individual motor and its driver stage. So, if the pulses are suddenly stopped while the motor is still turning, the rotor will not come to an immediate halt, but rather jump on one or two positions because of its moment of inertia. That does not matter, or course, if the system has feedback, for example in the form of an encoder on the drive shaft.

And the solution? The best answer is not to allow interruptions in the data stream. Before issuing each motor pulse command, we can check whether the controller FIFO signal is asserted, indicating that the FIFO has emptied. Then — if we know the characteristics of the motor — we can decide on the basis of the previously-set command rate whether the motor might have got out of step (this can only happen above a certain, relatively high, rotation speed), and whether the drive in question should return to the limit switch position to recalibrate itself. If this behaviour recurs—that is, if you notice that the machine is returning to its reference points too frequently—then either you will need to use a faster computer, reduce the load on your system, or reduce the pulse rate for the motors.

None of this is really a problem, but the unnecessary recalibration movements do affect the average speed of the machine and hamper the system somewhat. A reduced motor step rate can increase the overall speed of the system, if it reduces the number of recalibration movements.

And there is an even better solution: the FIFO memory can be expanded, as used to be done with print spoolers. Exactly the same principle can be applied to the drilling machine. With a small static RAM of for example 8 kbyte the capacity of the FIFO can be increased to 8.25 seconds of data at the frequency of 500/s mentioned above.

This extra FIFO would take the form of an additional unit; it is not possible simply to change the microcontroller. Microcontrollers with a large amount of on-chip RAM are significantly more expensive. The additional module could be supplied with power via the OptSpare pin (5 V DC). Another possibility would be a small module on a printed circuit board to fit in the microcontroller socket which includes the extra buffering.

If such an idea is implemented, you will be able to read about it either in Elektor Electronics or on the drilling machine website.

turntable. This hole is machined extremely accurately: on pulling out the pin you should hear a satisfying 'plop'.

Fit the largest insert into the chuck on arm 1 and tighten the nut loosely. Loosen the eight screws in the head guide bearing block. There are eight oval places where the 4 mm shafts for the head guide assembly are fitted. Do not remove the screws completely, but only loosen them so that the head guide assembly can be moved laterally to and fro.

Using the CW and CCW buttons, bring arm 1 to the centre of the turntable until the chuck can be pushed down exactly over the pin. The head guide assembly must have enough play that no force is required to push the arm into the correct position. The precision obtained by this procedure is improved if the last command given is always CCW.

Now press the head assembly fully down and tighten the nut fully. The entire assembly is held vertically in the centre of the table by the steel bolt and so the bearings are forced into their position. Now, with particular care, tighten up first the lower, and then the upper bearing screws, initially gently, and then tightly. The nut can now be removed. The head guide now glides (we hope) smoothly up. The smoother this motion, the faster the machine will be able to drill.

Two important things have been done here: the head guides are now perfectly vertical and the arm length is set to exactly the desired value. One of the polar coordinates — the length — is now done. The coordinate that remains, the angle, is handled by the software. Press the *INIT* button for the arm that has just been calibrated. The arm automatically moves out until its pointer is located over the optical sensor. The software now knows exactly the number of angular steps between the end point and the mid-point. The mid-point corresponds to an angle of 45°.

Now, to test the middle position, remove the bolt from the central hole and fit it directly in the chuck. In the software, press the *CENTER* button, and the arm should move to the middle position. When the axle is pushed down, the pin should sit perfectly in the hole. Release the chuck, allow the head to slide up, move the arm away and then remove the pin from the hole.

If the positioning is not perfect in this experiment, the arm angle can be adjusted using the CW and CCW buttons. The last command should, as before, always be CCW. After each adjustment the INIT button must be pressed again to store the new reference values. After the button is released, the arm travels immediately back to the sensor position. Once the mid-point positioning is to your satisfaction, press HOME. Target 1 now travels to its home position. The procedure with the second arm is the same: first, as shown in **Figure 3b**, bring arm 1 out of the danger area and click on *WAIT*. When calibration is finished, be sure to keep the bolt in case you wish to recalibrate the machine after modifying or expanding it.

Turntable calibration

If you have calibrated two arms, then arm 2 will be over the sensor and arm 1 will be in the wait position. Press *WAIT* also for arm 2 so that both arms are vertical and parallel to one another. If you have only calibrated one arm and this is in the home position, then you should also press *WAIT*. To calibrate the turntable put the approximately 8 mm long registration pin from the bag which contained the chuck into position 1 of the registration guide: this is the outermost hole on the rim of the turntable. Set target to 1 and press *TURN1*. Arm 1 makes a small movement from its waiting position towards the turntable.

With the *T-CCW* button (NOT *T-CW*!) turn the turntable so that the bolt in the registration guide lies under the chuck. Now with the *T-CCW* and *T-CW* buttons you can carry out a fine adjustment so that the chuck slides over the bolt when the head guide assembly is pushed down. The last command should again be a *T-CCW*. When you are happy with the positioning, press *T-INIT* and then the button *TURN2*. The turntable turns about 140° counterclockwise. At the same time the tool arm also moves and the two come to the second possible intersection point. The mathematical background to the positioning scheme for the drilling machine is shown in a Flash animation at this url

www.radixgmbh.de/deutsch/menu_link.html

When the turntable and the tool arm come to a halt, the test pin should once again slide into the chuck smoothly and without undue force on either part when the head guide assembly is pushed down. Once the calibration of the two turntable points is complete, press *T-INIT* again: the turntable will turn to the point where the optical switch is activated, the *HOME* position, and then the tool arm moves to the *WAIT* position.

If a second arm is fitted, the calibration procedure should be repeated. Although this is unnecessary from a mathematical point of view, it helps to compensate for tolerances in the system.

When the program is exited by clicking on the cross at the right-hand end of the title bar, the file *TanBo.def* is written to the hard disk. This file contains all the measured parameters which are used by the control programs. Without *TanBo.def* the *TanBoDrive* program cannot be started.

Setting up the machine also requires checking the planarity of the turntable. The more level the turntable, the less packing material is required between circuit board and turntable. The turntable is fixed to the drive axle via a generously-dimensioned fixing flange in the form of a circular steel disc by three M6 screws, easily accessible from above. By turning the turntable you can test to see if the surface moves up and down. Mark the 'deepest' point and dismantle the turntable. On the marked point, on the flange, lay a scrap of paper wetted with salad oil. The reason for the oil is that it makes the paper more dimensionally stable so that the thickness of the paper is not affected by humidity. Screw the turntable down onto the flange, tightening the screws fully. After at most two or three trials you should be able to turn the turntable without measurable vertical wobble.

The packing material — a thin piece of card or a couple of sheets of paper — is needed because drill points are in general not cylindrical but rather conical in shape. The card must therefore be thick enough for the point of the drill. The maximum feed distance, and so the lowest position of the drill, can be set by an adjustment screw at the front of the tool arm.

Drill!

The machine is now calibrated and ready to drill your first printed circuit board. The basic requirement (apart, of course, from an etched but as yet undrilled circuit board) is a drilling file in Excellon format. Practically every printed circuit board layout program is capable of exporting a drilling file in this format. Among other things, the Excellon file contains the Cartesian (or X/Y) coordinates of each hole specified in the layout. The file also contains control information and (possibly in a second file) information specifying the diameter of the hole to be drilled at each given coordinate pair. This file can be opened, examined and even modified using a normal ASCII text editor.

The *IMPORT* button in *TanBoDrive* (**Figure 4**) opens a standard Windows file selection box and waits while the Excellon file is selected. The file is searched for the two points used in the reference system; the diameters specified for these holes are not used. Once the two coordinate pairs are found by *TanBoDrive* they are used as reference points, but, of course, not drilled. It is necessary to specify in which of the two reference points the fixed pin is fitted: for this reason a thumbnail of the drilling pattern is shown in the dialogue box.

Place the ready-etched printed circuit board on the turntable and select one of the twelve holes in the registration system. Fit the circuit board over the short metal pin and slide the free pin so that it fits in the second hole in the printed circuit board. Push the free pin outwards a little so that the circuit board is better centred. If the circuit board is badly warped, it is necessary to hold the corners down with adhesive tape.

In the program select in the *FixPoint* box the number of the registration hole to which the circuit board is fitted. The positions are numbered from outside (greatest radius) to inside from 1 to 12. There is no need to count: with each click the drilling pattern picture is updated so as to show the selected pin position. You can therefore check at a glance that the number is set correctly.

At this point there are two things that might go wrong. First, the circuit board might be rotated 180°: the hole shown over the fixed pin in the dialog box is in fact over the moving pin, and vice versa. Click in the check box marked *ROTATED* and the circuit board will be shown correctly on the screen.

Some programs produce drilling information as viewed from the reverse of the board (as the copper foil patterns are shown in the component mounting plans for *Elektor Electronics* printed circuit boards). Since the drill tends to wander if the board is not drilled from the copper side, the *REFLECTED* check box must be clicked to correct the situation.

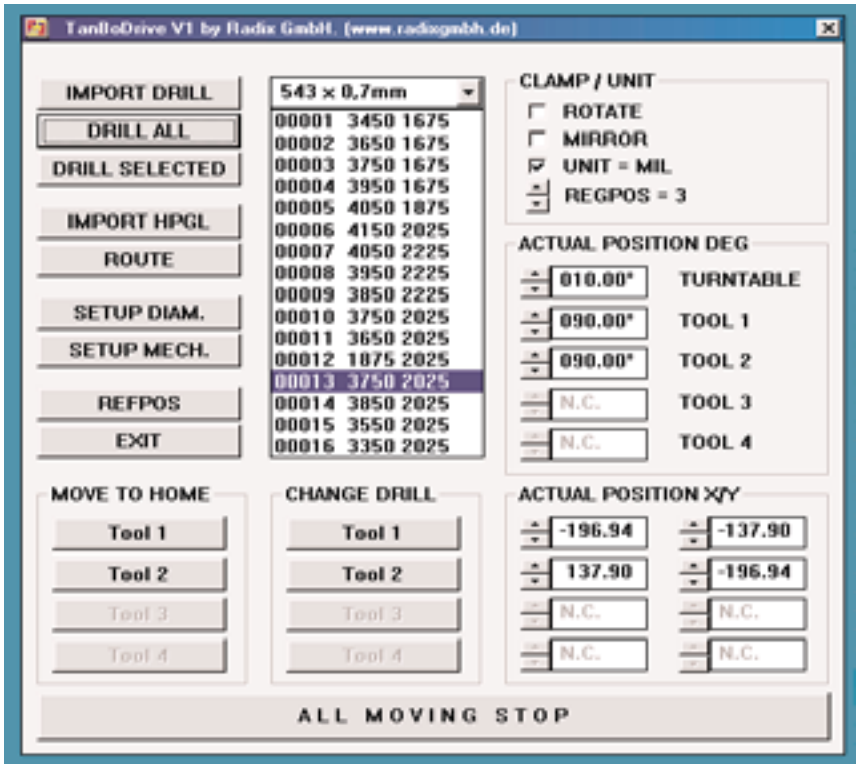


Figure 4. In the driver software TanBoDrive an Excellon file can be imported and the orientation of the circuit board can be set: ready to go!

There is only one thing left to do: click on START. The drilling machine moves the first tool arm into the drill change position and displays an information box with the required diameter of drill. Fit the appropriate drill in the chuck and click on OK. The machine will now run, drilling at each point requiring the first drill size. If a second arm is fitted, it will meanwhile move to the drill change position.

Further requests follow, depending on how many different sizes of hole are to be made or if multiple identical circuit boards are to be drilled. It is unnecessary to describe all these here, since the software is very user friendly and practically self-explanatory. If something should go wrong, or if any difficulties are encountered, a platform where you can exchange experiences with other users and praise or berate the developers of the machine is provided at the Radix homepage.

(010024-5)

Note:
Kits and individual parts for the PCB Drilling Machine are supplied by the author via the Radix GmbH company in Germany. At the time of writing, negotiations are under way with C-I Electronics to handle international distribution of the mechanical kits, possibly including the Elektor Electronics driver board. Further information from C-I Electronics, P.O. Box 5514, NL-3008-AM, Rotterdam, The Netherlands. Fax (+31) 10 4861592, email dil@euronet.nl. Website www.dil-dos.nl

Souping up CD Burners

higher speed with different firmware

By Harry Baggen

Not only can you soup up cars, scooters and mopeds, you can also soup up computer equipment. Running CPUs at clock rates higher than what they were originally intended to use has become a popular sport. However, not everyone realises that various types of CD burners can also be 'upgraded' to better or faster versions.

Nowadays, a CD writer (or 'burner', as it has come to be commonly known) is practically a standard component of a modern computer. CD-R disks are dirt cheap, and a CD burner is the ideal means for transferring large volumes of data between computers, making backups of programs and files, making audio CDs with compilations of your favourite tunes and generating MP-3 CDs.

The pace of development with CD burners has been so fast in recent years that a new model with even higher speed appears every few months. Among owners of older models, this gives rise to the urge to acquire a faster model.

Clever computer users have figured out that just as with processors, it's quite possible to fiddle around with CD burners in order to extract a bit more speed. With some older models, such as certain Yamaha units, this involves making some changes to the printed circuit board. However, nowadays the majority of the modifications involve making adaptations to the firmware, which you could regard as the 'BIOS' of the CD burner. If you use different firmware, the CD burner will act like a different model. This is possible because there are only a limited number of manufacturers of mechanisms and electronics for CD burners. The internals of different types of CD burners are sometimes so similar that a simple firmware change is all that is needed to obtain a different speed. For example, an 8¥ Sony unit (CRX-140E) can be upgraded to a 10¥ unit by 'flashing' it with the firmware from an HP burner (which has a Sony drive on the inside).

There are two sites that have collected a number of modifications for CD burners: **CD Media World [1] and CDR-Info [2]**.

It is certainly interesting to experiment with such modifications. However, we must warn you of possible damage to the burner. It may happen that after you have flashed your burner with different firmware, it will no longer respond at all and cannot be restored to its original state. Consequently, you should preferably try this trick with an old burner that you have already written off. That way, if anything goes wrong, you can just buy a new one!

(015081-1)

Internet-Addresses:

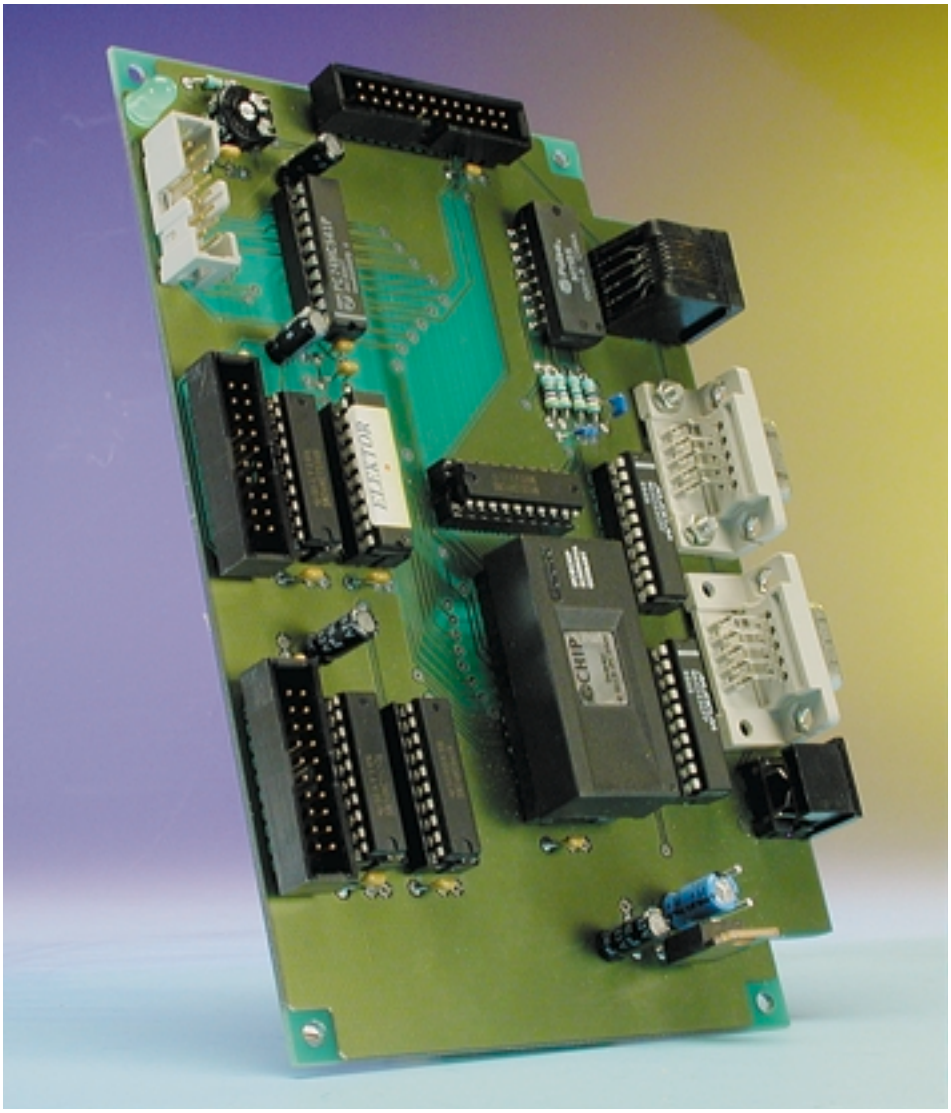
[1] CD Media World:
http://www.cdmediaworld.com/hardware/cdrom/cd_modifications.shtml

[2] CDR-Info:
http://www.cdr-info.com/tips/hardware_oc.shtml

Personal Mini Webserver (1)

a development board with an embedded PC

An embedded PC in a DIL package forms the core of the Mini Webserver. In this first instalment, we introduce an evaluation board for this controller and show how to use it. Following this, you will be in a position to control any equipment that you wish via the Internet or an Ethernet network.



Sometimes, a passionate electronic hobbyist can get the impression that modern-day electronics can no longer be implemented in DIY form. The microcontroller has too many pins, loading software into the DSP requires professional programming equipment with a matching price, the passive components are so small that you need a loupe to see what you're working with and soldering them with a normal iron is hopeless, and finally, not everyone is equipped to make multilayer circuit boards.

However, things do not have to always be this difficult, and the Mini Webserver presented here is proof of this. This small, easily built circuit, which allows you to communicate with a piece of electronic equipment remotely (as far as the Internet reaches…), is the very latest thing and (not yet) a mass-produced industrial product that you can purchase for a few Euros in your neighbourhood telephone boutique or electronics shop.

In the two previous issues of *Elektor Electronics* (May 2001 and July/August 2001) we gave our attention to the fairly complicated technologies behind communications via a local or Internet network. The controller of the Mini Webserver must be prepared to deal with a whole series of communications protocols. Imple-

Features of the Personal Mini Webserver

– 16-bit 80186 CPU with 20 MHz clock
– 512 kB RAM, 515 kB flash memory
– RTOS with Flash data system
– Software download via RS232 or Ethernet
– Protocols: TCP/IP, PPP, HTTP, FTP, Telnet, POP3, SMTP, ICMP and DHCP
– Modem connection: PPP via 9-way Sub-D connector
– Ethernet: 10BaseT with PHY via RJ45 socket
– Terminal connection: fast serial port with RS323 (RXD, TXD, CTS, RTS) via 9-way Sub-D connector
– I²C bus (master) via 6-way mini-DIN socket
– Watchdog
– 2 timer outputs, 2 timer inputs
– Power fail detection (NMI) with data saving
– Intel-compatible multifunctional A/D bus
– 16 digital inputs
– 16 digital outputs
– Direct LCD connection
– Addressable expansion connection (A/D bus, I²C bus, interrupt, ALE and RD/WR)

CPU Functions

– Multiplexed address/data bus (8 bits)
– 6 programmable chip selects
– 6 interrupt-capable inputs
– An external interrupt controller can be connected
– Two-stage hardware watchdog
– 3 timers
– 2 fast counter inputs
– 2 fast counter outputs
– 2 DMA controllers
– 10BaseT interface
– 2 asynchronous serial interfaces
– 1 serial synchronous interface (SSI)
– I²C bus-master capable
– Up to 14 programmable I/O pins
– Reset/Power Fail (NMI) generator

menting TCI/IP, HTTP, FTP, SMPT, an operating system and so on in a microcontroller is not a trivial task. Fortunately, a few commercially available microcontrollers have the necessary functions already 'on chip' or at least packaged into a small supplementary module, so we do not have to worry about the network side of the server and can concentrate fully on the application circuit.
This is precisely the purpose and objective of the hardware presented here. It has primarily been designed as a development system, so it provides access to all available interfaces and ports of the microcontroller that is employed via sockets and headers. Naturally, the development board can also be used as a prototype by installing only the necessary components, if enough space is available in the application device.
There is a connector on the circuit board that can be used to attach the application device or an extension for the development board, such as D/A and A/D converters or additional interface components.

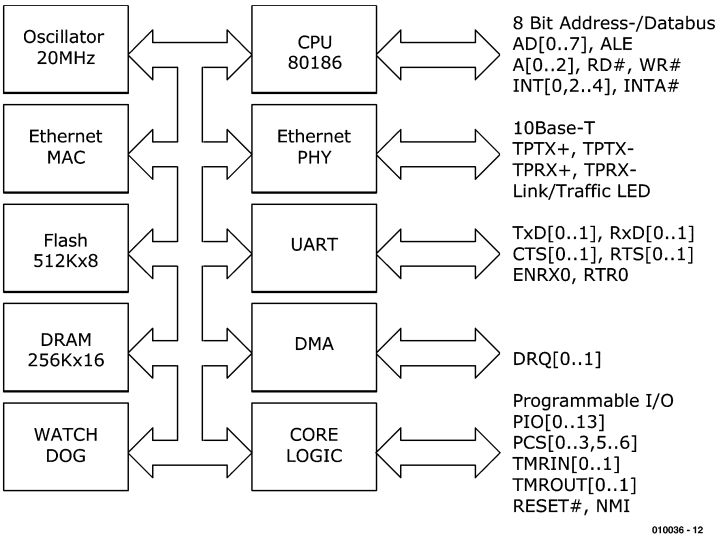


Figure 1. Block diagram of the IPC@CHIP SC12 single-chip PC.

Internet PC on a chip

One of the few 'embedded' microcontrollers that is available to normal mortals is the IPC@CHIP SC12, which is made by the Beck company in Wetzlar, Germany. The SC12 is housed in a 32-pin DIL package. As can be seen from the block diagram in **Figure 1**, it is based on an 80186 processor core clocked at 20 MHz and has 512 kB each of DRAM and flash memory. This basic configuration is complemented by service functions such as serial communications via a UART, an I²C interface, a direct bus interface and 64 kB of memory. The SC12 supports Ethernet communications (10BaseT) via twisted-pair cable using Medial Access Controls (MAC) per IEEE 802.3.
The SC12 conforms to an ISA architecture that is modified in a few points, with no ISA bus and no video interface. The programming of the serial ports, DMA, PIC and timer is also different from that of a standard IBM PC.
The most important CPU functions are listed in the box. In addition to this hardware, there is an integrated operating system called RTOS, whose similarity to DOS gives it a familiar appearance.
Beck justifiably calls the result a 'single-chip PC' that is compatible with all development tools and applications for 80C186 microcontrollers. This means that anyone who has already programmed such a microcontroller

in C, Pascal or assembly language will have no problems with the SC12. That is very encouraging, since development tools for the 80186 and a DOS-compatible operating system are like sand on the seashore – inexpensive and massively available.
Should you nonetheless not succeed in getting a functional program up and running, you can obtain help from numerous sample applications and programs (with source code) for the basic functions of the SC12 that are available from the websites of Beck and *Elektor Electronics*.
Naturally, such a high-tech component doesn't exactly come cheap. The price of the SC12 (May 2001) is 66 Euros, and a Borland C compiler can be obtained for 49 Euros if purchased together with the controller. Finally, you will need an additional 25 Euros (approx. £16) for a single-user runtime license for the RTOS operating system with multitasking DOS, TCP/IP stack, Webserver with CGI interface, FTP server, Telnet server and an API for programming in C, although the last of these is only needed for professional applications. In light of the relatively small amount of hardware and software that you have to provide yourself and the possibilities that such a high-performance mini webserver offers, this is certainly an acceptable price for the user. The product range of Beck also includes a small development board (part designation DK40), which however only has TTL interfaces (with RJ12/45 sockets) and eight buffered I/O connections.

Busses and interfaces

The SC12 is not inseparably linked to the hardware of the development board. You are not absolutely required to configure a serial port as a terminal port (TxD0, RxD0) for debugging purposes. You can also do without a modem connection if your application will exclusively employ the Ethernet. In the latter case, the second serial interface also becomes free.
On the application side, there are many port connections of the SC12 that have multiple functions. If you do not need an I²C bus, for example, you can freely program the corresponding port leads or use them as interrupt sources.
All of these considerations are arguments for taking a closer look at the functions of the individual pins. **Figure 2** shows the pin assignments of the SC12, with numerous double and triple functions.

Address/Data bus

The first thing that catches our attention is the synchronous, level-sensitive multiplexed address and data bus (AD0–AD7). The lower eight bits of the address appear on this bus in the first period of the bus cycle, followed by three periods of data. The address phase can be suppressed.
The address bus is completed by the non-multiplexed, synchronous address outputs A0–A2. These address bits are valid one half of a clock period prior to AD0–AD7.
A rising edge on the synchronous ALE (Address Latch Enable) output indicates a valid address on the combined address/data bus, WR and RD enable write and read cycles.

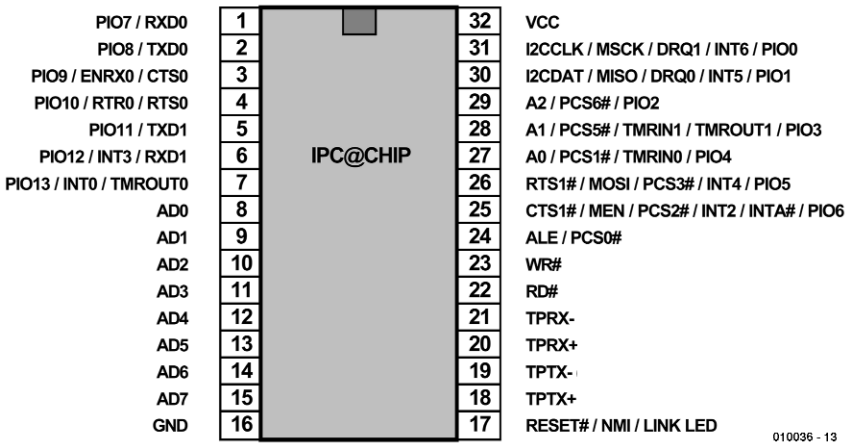


Figure 2. Many of the pins of the SC12 are multifunctional.

Programmable I/O pins

The 14 pins PIO0–PIO13 are synchronous open-drain port lines, whose function, direction and active level attributes are programmable. Certain settings are predefined following the power-on reset, and must be reconfigured afterwards by software as necessary to obtain the desired settings.

Programmable chip selects

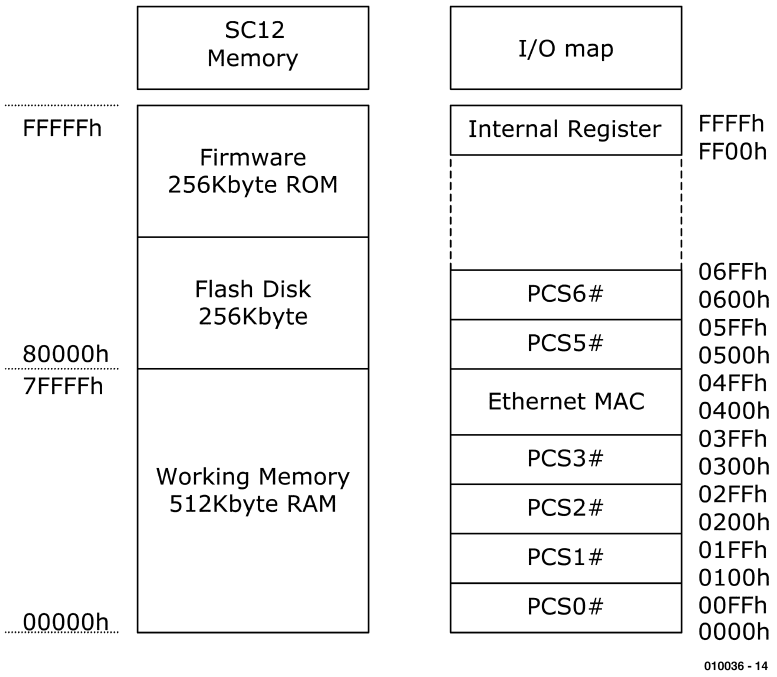


Figure 3. The memory configuration and memory-mapped I/O base addresses.

The synchronous selection outputs PCS0–PCS6 steer I/O accesses to the peripheral memory ICs. The PCS outputs work together with the multiplexed A/D address bus.

Interrupts and timers

Pins INT0, INT2, INT3 and INT4 represent maskable asynchronous interrupt inputs. If an interrupt signal appears at an unmasked input, the program is interrupted in favour of the routine that is pointed to by the corresponding interrupt vector. Interrupts are synchronised internally and can be level or edge triggered. INT1 works for the Ethernet MAC and is not available for external use. If INT0 is working in cascade mode, which means that several interrupt sources are connected via a priority encoder, the INT2 input changes into the synchronous INTA output. If INTA is active, the hardware must place an 8-bit value on the data bus so that the controller can identify the source of the interrupt.
The microcontroller has two internal timers that can be externally accessed via the TMRIN0 and TMRIN1 inputs. Following internal synchronisation, each positive edge on TMRIN increments the corresponding counter. If it is not used, TMRIN must be externally connected to a High level.
The timer outputs TMROUT0 and TMROUT1 provide either single pulses or a continuous pulse train with a programmable duty cycle.
The latter type of signal results from an internal combination of two interrupt and timer inputs in PWD (Pulse Width Demodulator) mode. PWD drives TMRIN0 and INT2 and the inverted TMRIN1 and INT4. If INT2 and INT4 are enabled and the timer is properly configured, the duty cycle corresponds to the ratio between the two timer values. In the PWD mode, TMRIN0, TMRIN1 and INT4 can be used as PIOs. If they are not used, they are ignored internally.

10BaseT Ethernet interface

The SC12 includes a complete Ethernet controller/transceiver that is accessible via the TPRX+ and TPRX– inputs and the TPTX+ and TPTX– outputs. The transceiver sends and receives signals via an output transformer and a twisted-pair connection. A LED connected to LINK LED blinks dimly if an Ethernet connection is present and brightly when data traffic is present.

Asynchronous serial ports

The SC12 supports the full-duplex transfer of two asynchronous serial data streams with 7–9 bit data at transmission rates up to 115,200 baud via pins TxD0, TxD1, RxD0 and RxD1. Freely selectable handshaking is provided for each port by CTS0, CTS1, RTS0, RTS1, ENRX0 and RTR0.

Direct Memory Access

Via the DMA Request inputs DRQ0 and DRQ1, an external component can announce a data transfer via one of the two DMA channels. DRQ0 is edge triggered and internally synchronised, while DRQ1 is level sensitive. DRQ0 is not latched and must remain active during the transfer.

Inter-IC Bus

The SC12 has an I²C interface at pins I2CCLK and I2CDAT. The controller can drive up to 127 external slaves, but it is always the master.

Reset, Power Fail generator

A level of less than 0.8 V on the asynchronous RESET pin immediately interrupts all current activities, resets the internal memory and causes the CPU to jump to the reset address FFFF0h. If the supply voltage for the system drops below 4.5 V or the watchdog times out, the Reset pin is forced to ground.
The synchronous, edge-sensitive NMI (Non-Maskable Interrupt) input is the interrupt source with the highest priority. It is completely independent of the interrupt controller for INT. A NMI request interrupts program execution, which jumps to the program location indicated by the NMI vector. NMI is used in combination with the internal watchdog.

We can complete our description of the connections with a summary of the memory configuration and the I/O base addresses, which are shown in **Figure 3**.

The hardware platform

The schematic diagram of the Mini Webserver in **Figure 4** now lets us

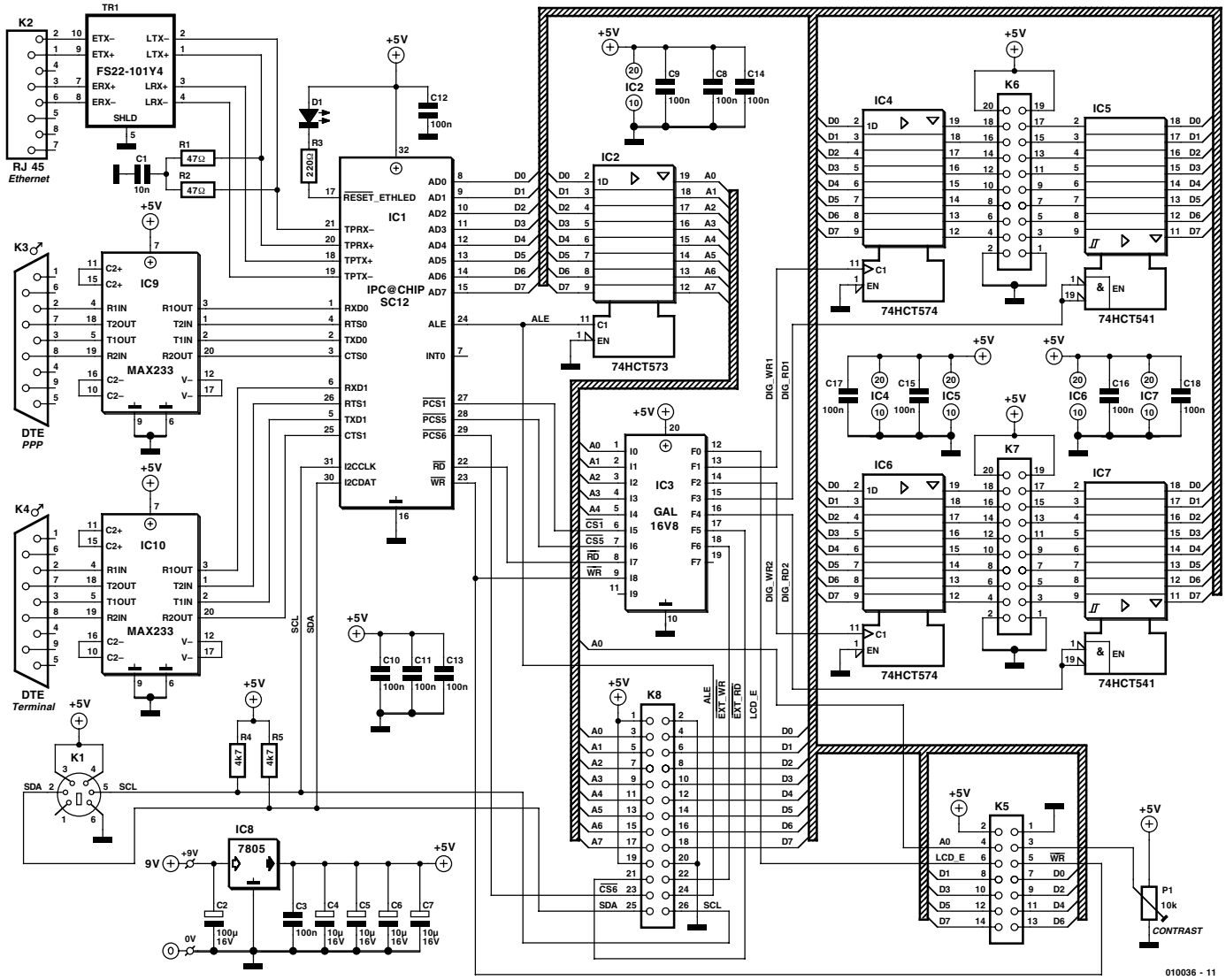


Figure 4. The circuit diagram of the development board for the SC12.

see which functions of the highly multifunctional pins of the SC12 are implemented on the development board. The combined multiplexed data/address bus AD is demultiplexed by the address latch IC2. Address bits A0–A4 are fed to an address decoder in the form of a GAL. Together with the chip select lines PCS1 and PCS5 and the read/write signals, the programmed logic selected the recipient of the data that comes after the address, assuming that it is present: 16 digital outputs buffered by IC4 and IC6, 16 digital inputs isolated by IC5 and IC7 and a two-line LC display. The digital inputs and outputs are connected to the two headers K6 and K7, with the connections for each port pin always located opposite each other.

The demultiplexed address A0–A7 and data D0–D7 are present on an expansion connector, along with the I²C bus (SDA and SCL), interrupt source 0 (INT0) and the ALE sig-

COMPONENTS LIST

Resistors:
R1,R2 =47Ω
R3 =220Ω
R4,R5 =4kΩ7
P1 = 10kΩ preset

Capacitors:
C1 = 10nF
C3,C8.-C18 = 100nF
C4-C7 = 10µF 16V radial
C2 = 100µF 16V radial

Semiconductors:
D1 = LED, green
IC2 = 74HCT573
IC4,IC6 = 74HCT574
IC3 = GAL16V8 (programmed, order code **010036-31**)
IC5,IC7 = 74HCT541
IC8 =7805
IC9,IC10 =MAX233CPP
IC1 = IPC@CHIP SC12 (order from Beck, http://www.bcl-online.de)

Miscellaneous:
PC1,PC2 = solder pins
TR1 = FS22-101Y4 (order from Beck, http://www.bcl-online.de)
K1 = 6-way Mini-DIN socket (female), pins at 240 degrees, PCB mount
K2 = RJ45 socket (female), PCB mount
K3,K4 = 9-way Sub-D plug (male), PCB mount, angled pins
K5 = 14-way boxheader or pinheader
K6,K7 = 20-way boxheader or pinheader
K8 = 26-way boxheader or pinheader
Enclosure:
PCB, order code **010036-1**

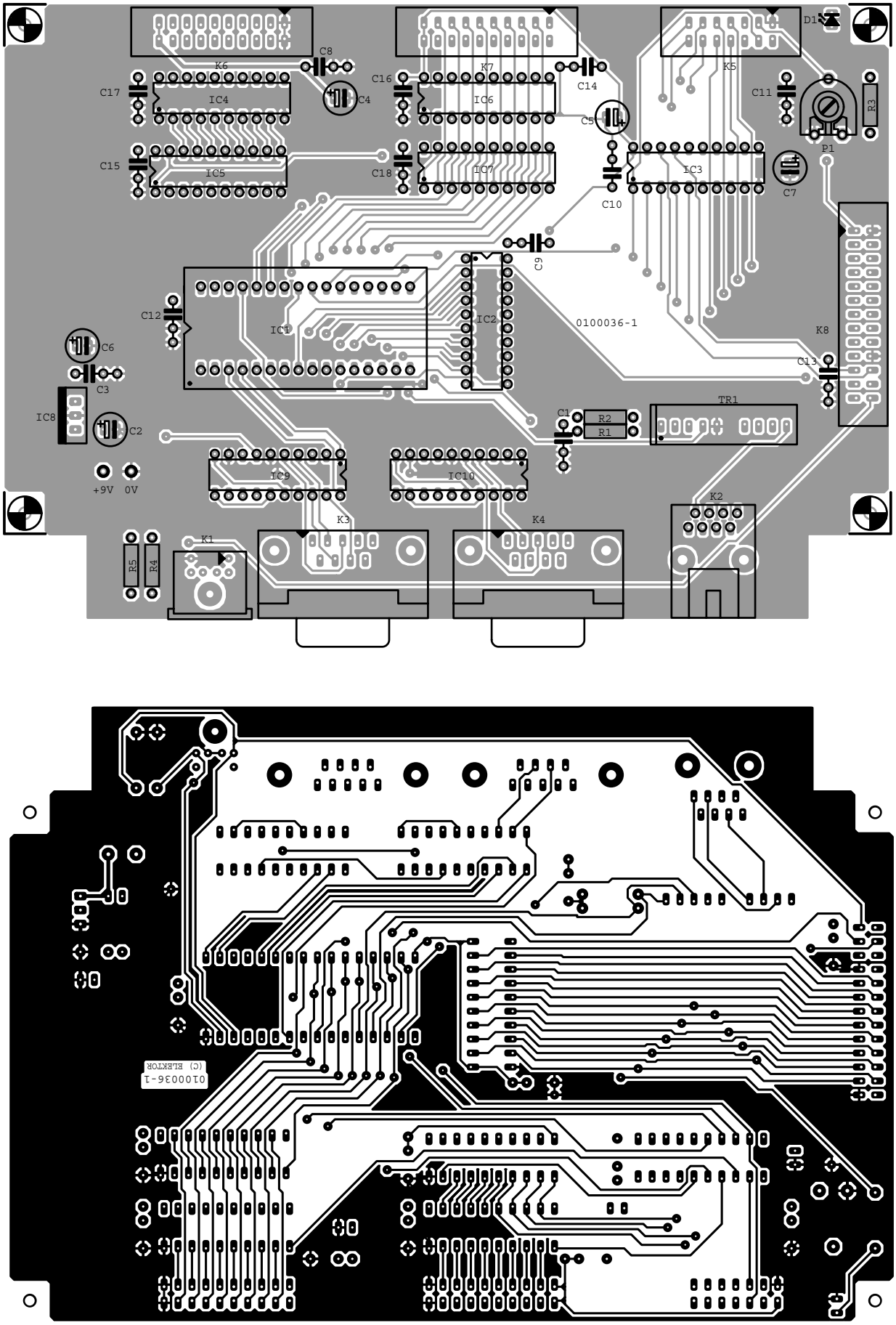
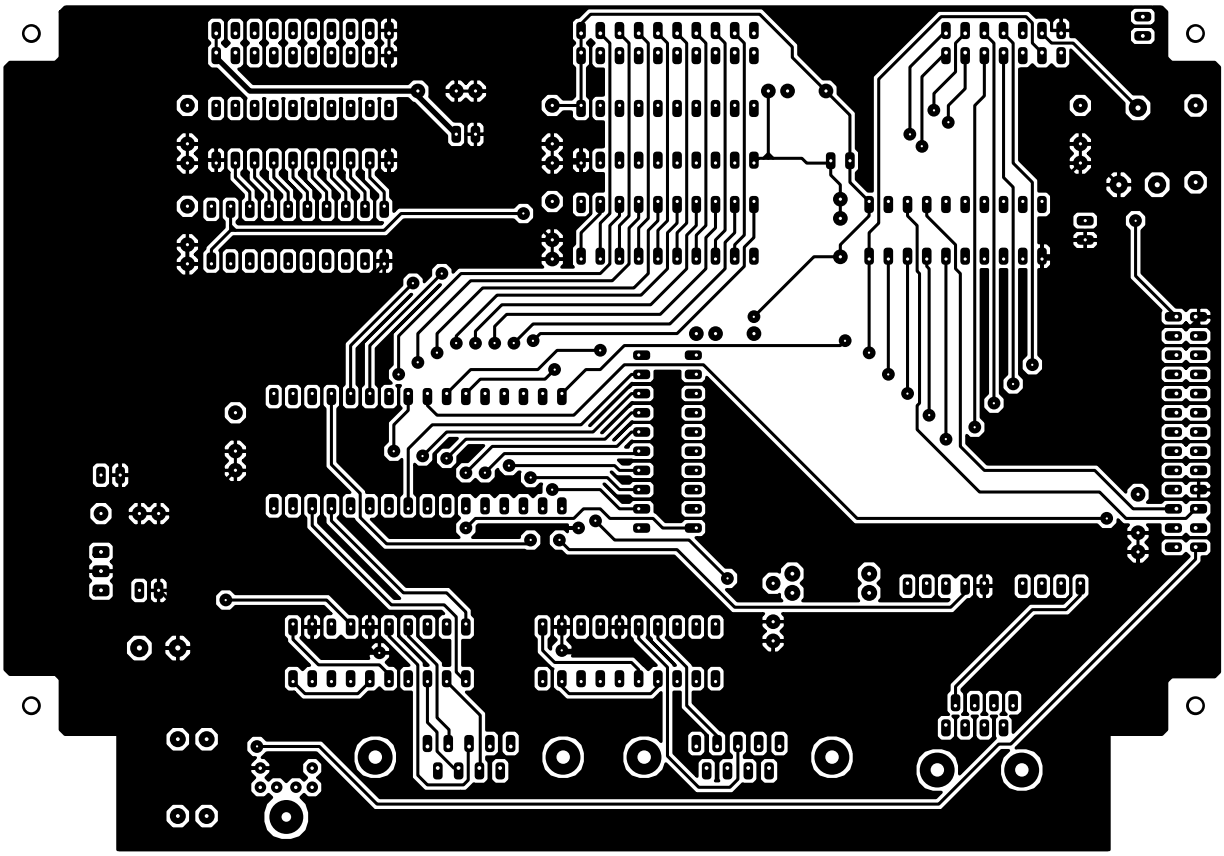


Figure 5. The double-sided printed circuit board is designed for ease of use.



nal. Selection signals for the expansion logic are also provided by the GAL, with separate signals for write (EXT WR) and read (EXT RD) operations. In addition, the +5 V supply voltage is also brought out to the expansion connector.

On the other side of the SC12, everything is standard: the two asynchronous serial interfaces for the modem and terminal are used as intended. Since the controller naturally provides only TTL levels and can only handle such levels, two MAX233 interface converters (similar to the MAX232 but without external capacitors) are provided in the form of IC9 and IC10. These allow signals with RS232 levels to be received or transmitted via the 9-way sub-D connectors. Note that both of these interfaces are DTEs, so you must use a null-modem cable for the PC and a regular 1:1 cable for the modem.

The connection between the Ethernet controller/transceiver in the microcontroller and the network requires good termination of the differential signals and electrical isolation. These requirements are satisfied by R1, R2, C1 and the special matching transformer Tr1. The common pin for Reset, NMI and LINK LED is used to drive a LED that indicates the traffic on the Ethernet.

Furthermore, we have kept in mind that there are a lot of I²C fans among the readers of *Elektor Electronics* and have thus fitted the I²C interface with pull-up resistors and a mini-DIN socket (K1), so that existing *Elektor Electronics* application circuits can easily be plugged in.

The supply voltage is adequately stabilised by a simple +5-V voltage regulator and a few decoupling capacitors. A mains adapter can be used as the power source as long as it can supply at least 400 mA at 9 V. If the connected application circuit must also be powered, its current consumption must also be figured in.

The double-sided printed circuit board is laid out as befits a development board. Everything is easy to find and not crowded too closely together, and all the connectors are arranged at the edge of the board. You will look in vain for wire jumpers, SMD components and similar soldering specialities.

This means that stuffing the board is also not a problem. All ICs can be mounted in sockets, as long as you pay attention to the polarisation (as with the electrolytic capacitors).

Once you have finished the soldering work with a good visual inspection, you are ready to wait in eager anticipation for the following issue of *Elektor Electronics*, which is where the Mini webserver will see action for the first time. In the following instalment, you will learn how to work with the terminal software and how to set up an Ethernet connection, an HTTP link or an FTP server. In addition, we well present a few sample application circuits that show how to solve certain hardware and software tasks.

However, if your interest in the Mini Webserver is so great that it prevents you from enjoying your holidays on the beach or in the mountains (or on your balcony), you should be able to find relief in the nearest Internet café.

At http://www.bcl-online.de you can view a vast number of documents related to this topic and the SC12 controller.

(010036-1)